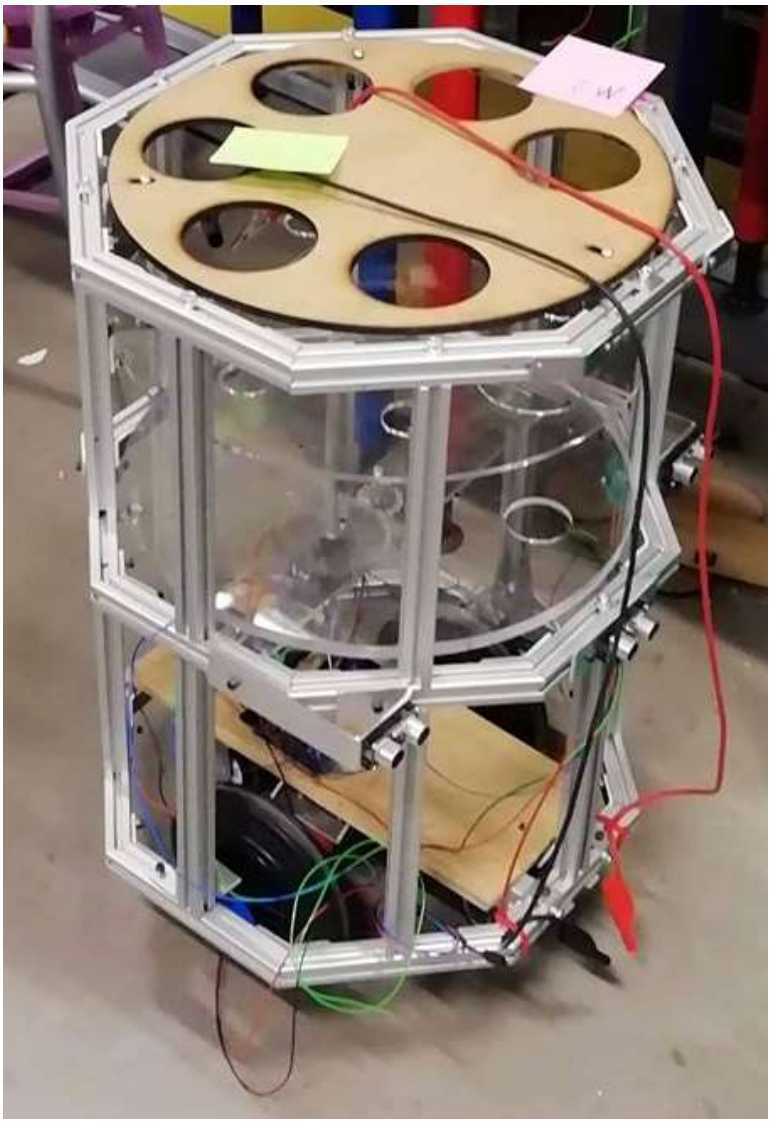
Bar Operating Bot

B.O.B.



Mechatronics project semester 3, Group 9

Table of Contents

Signatures:

Wiktoria Aleksandra Kos: 

Morten Milton Schou: 

Nick Henderson: 

Niels Tscherning Andersen: 

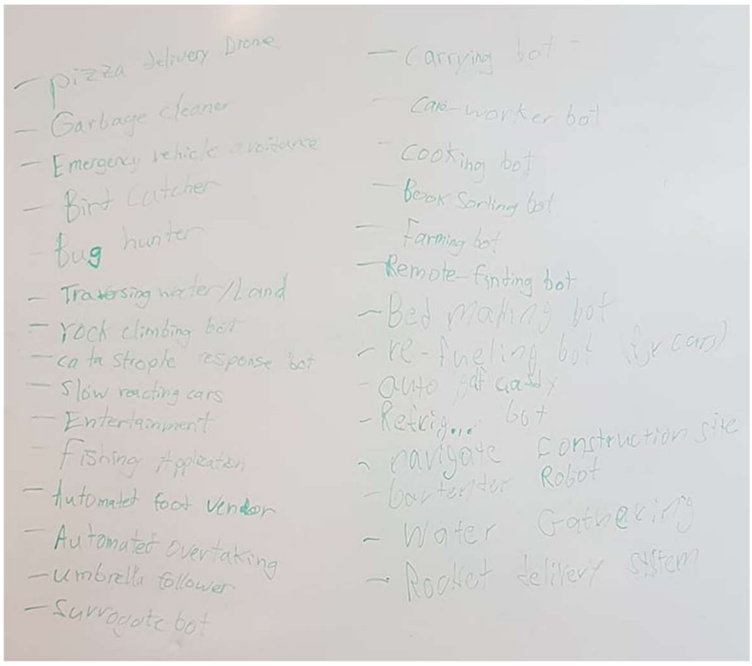
Patrick Ma: 

1. Introduction

The task given was to build an autonomous vehicle. This means that the device should be able to drive and adapt to specified environments by use of advanced sensors and control solutions. As is evident, this creates a complex problem that will need to combine multiple areas such as mechanics, hardware and software in order to achieve its goal.
It is also a broad topic that can be implemented in many areas. Therefore, it was necessary to first find the problem that may be solved by an autonomous vehicle and then develop the product based on that.

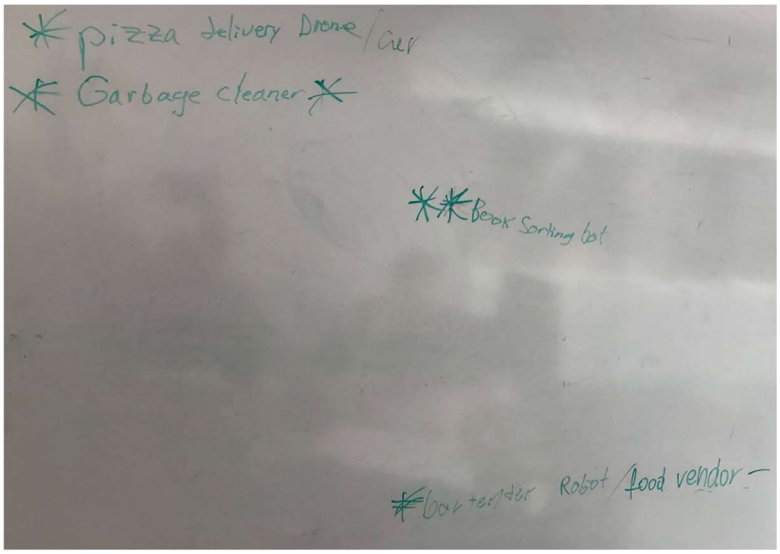
2. Project Background

2.1. Idea Creation



Brainstorm:

The brainstorm process was started by writing down all the ideas the group had come up with for a self-driving robot. Everything from a pizza-delivering drone to an umbrella-carrying robot was suggested. When the group could no longer come up with more ideas a vote was started to eliminate ideas until only the five best ones, as perceived by the group, where left.



Once the five most popular ideas were chosen, a more in-depth voting system was taken into use. This voting system used five different categories each weighted differently. They were weighted differently by applying a multiplier to each category based on importance. The categories are Simplicity, Creativity, Interest, Application and Relevance.

Simplicity: Describes how simple the group thinks the idea is to realize, is it too advanced?

Creativity: Describes how unique the idea seems compared to what can be found in real life.

Interest: Describes the overall interest the group members have in the idea.

Application: The ability to be applied to a real-world situation.

Relevance: How relevant the idea is to the project description.

The category that was ranked most important was creativity and had a multiplier of 3, on the other hand Relevance was ranked lowest with a multiplier of 1.

Problem Vote:	Simplicity	Creativity	Interest	Application	Relevance	Total score	
Pizza Delivery ATV Bot	6	3	5	7	7	44.75	#3
Garbage Street Cleaner	3.5	4	4	8	6	42	#4
Book Sorting Bot	6	7	7	6.5	5.5	57	#2
Bartender Bot	5	7.5	8	6	7.5	59	#1
Multiplier	2	3	1.25	1.5	1		

Each group member graded each category on a scale from one to ten. The average of each result was used as the overall grade for the idea in that particular category. In the final result the multipliers were applied and all the grades added together. In the end, the idea with the most points was the bartender bot with 59 points.

2.2. Survey

See survey results in APPENDIX H

As can be seen by the results of the survey, many people feel like they stand in queue for too long when ordering drinks and most people think that the drinks are too expensive. On the other hand, only a few people have trouble with spilling their drinks when moving away from the bar and even less people get the wrong order from the bartenders. Having human contact with the bartender is considered important to most people but there are a significant number of people who does not feel it is important. Most people also think they get their drink mixed right but there are still many customers who does not get the right mix. Most people sometimes have problems getting the attention of the bartender although a large amount of people rarely has that problem.

Overall, the robot would be able to help in the areas costumers have complaints about, and in the process, it will solve some issue that are less prevalent but some people still encounter when in bars.

2.3. State of the art

Beerbots:

If we look at the issue of delivering drinks to people the project by MIT called “Beerbots” does this for office buildings. It works by having two “turtlebots”, which are autonomously driven cooler boxes. These cooler boxes drive to a robotic bartender that supplies them with the beer that they need to deliver. The bartender bot cannot mix drinks. It can only place a beer can into the container on the turtlebot.

The turtle bots drive into rooms in the office building and ask the people in there if they want a beer, the office worker then presses a button on the turtlebot so that it knows to deliver a beer to that room. The turtlebots do not intercommunicate when they are too far away from each other, therefore they do not serve rooms as effectively as they could if they had constant communication. Like with our project it is also important that the delivering robots avoid obstacles and people.

Beerbots differs from our project in that they go around asking people if they want a drink, where our project needs to respond to a button press and then move to that location. Also, Beerbots cannot mix drinks, which our project needs to do because it is made for a bar and not an office building.

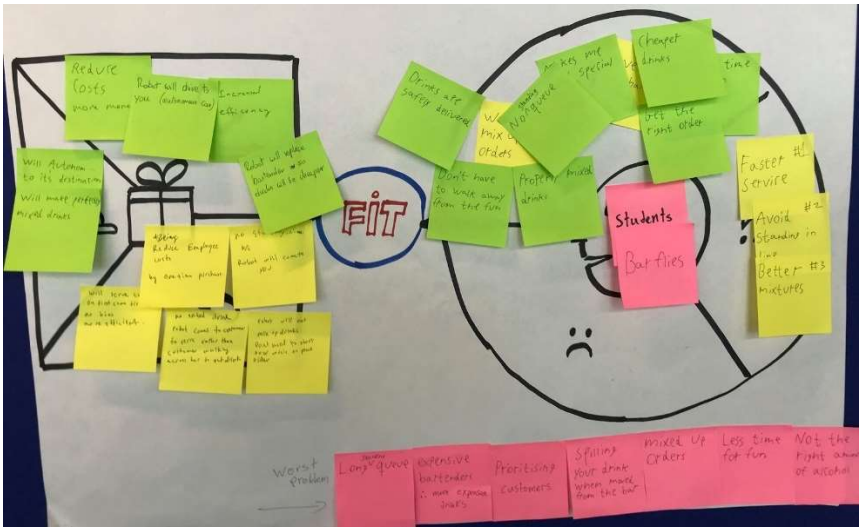
source: https://www.huffingtonpost.com/entry/beerbots-robot-bartender_us_55cca7ede4b064d5910a985d

Barbot:

Barbot is an Arduino based product that can mix cocktails. It does so by moving a glass along a conveyor belt under the different spirits and liquors that need to be mixed together. A valve is then opened so that the liquor may pour into the glass and is then closed when enough is poured in to achieve the desired cocktail mix.

The specific cocktail is ordered by the use of your phone where mix ratios and desired cocktail can be chosen to match individual preference.

source: https://thenextweb.com/shareables/2017/08/29/barbot-robot-cocktail-bartender/



Customers: They are students and barflies (people who frequent bars)

Customer jobs from most important to least important: Faster service, avoiding standing in line and better mixed drinks.

Pains from greatest to weakest: Standing in long queues, Expensive drinks due to expensive bartenders, bartenders prioritizing certain individuals, spilling your drink as you move away from the bar to another area, getting the wrong order, less time for other club activities and drinks being mixed at incorrect levels.

Gains: Drinks are safely delivered without spillage, no longer have to leave more interesting activities, drinks are mixed correctly, no need to stand in long queues at the bar, orders no longer being mixed up, no one is being prioritized and drinks are cheaper.

Products and service: Will automatically move to its destination and will make properly mixed drinks.

Pain relievers: No standing in line as the robot will come to you, reduce employee costs by a one-time purchase, will serve customers by a first come first serve basis with no bias, no spilled drinks as the robot will come to the customer and serve the drink instead of the customer walking across the bar with it, the robot will not mix up orders and no need for customers to shout over the loud music at the bar.

Gain creators: Reduced cost, robot will drive to you and increased efficiency.

3. Requirements

3.1.1. The robot must be able to accelerate to walking speed within 5 seconds.
3.1.2. Must be able to avoid collision with a detected stationary object when travelling at walking speed.
3.1.3. Must be able to navigate past a detected object.
3.1.4. The dimensions of the robot must be able to fit inside a rectangle of $4m^3$.
3.1.5. The mass range must be less or equal to 12 kg.
3.1.6. The robot must be able to be operated by a customer with little to no technical expertise.
3.1.7. The robot must be able to drive for at least 3 minutes without stopping.

3.2. **Limitations**

3.2.1. The project is limited to a budget of 1000 kr.
3.2.2. The project is limited to the period between the start and end of semester.
3.2.3. The project is limited to attainable resources.
3.2.4. The project is limited by the group's experience/skills.

3.3. **Delimitations**

3.3.1. The project will be limited to bars.
3.3.2. Technical drawings will only be made for the chosen idea.
3.3.3. Parts will be manufactured on campus, or purchased via the given budget.
3.3.4. The robot will follow an already set path, no AI.
3.3.5. The robot will only drive on flat surfaces.
3.3.6. There will be no time limit for the operation of the robot.

3.4. **Desirables**

Although there was no specific size requirement, Ideally the robot would be at a height at which one could easily reach their drink; about a meter tall. Unfortunately, this was deemed an unreasonable goal as the material requirement and labour involved to assemble it would be too great. To fit more viable dimensions, the initial design was scaled to about half the original size, excluding the width as this was dependent on the bottle sizes of the drinks.

Also, the group initially had the idea of designing the exterior of the robot to resemble a cocktail shaker. This was not of top priority, however, and in the end a more practical design was adopted.

4. Concept design

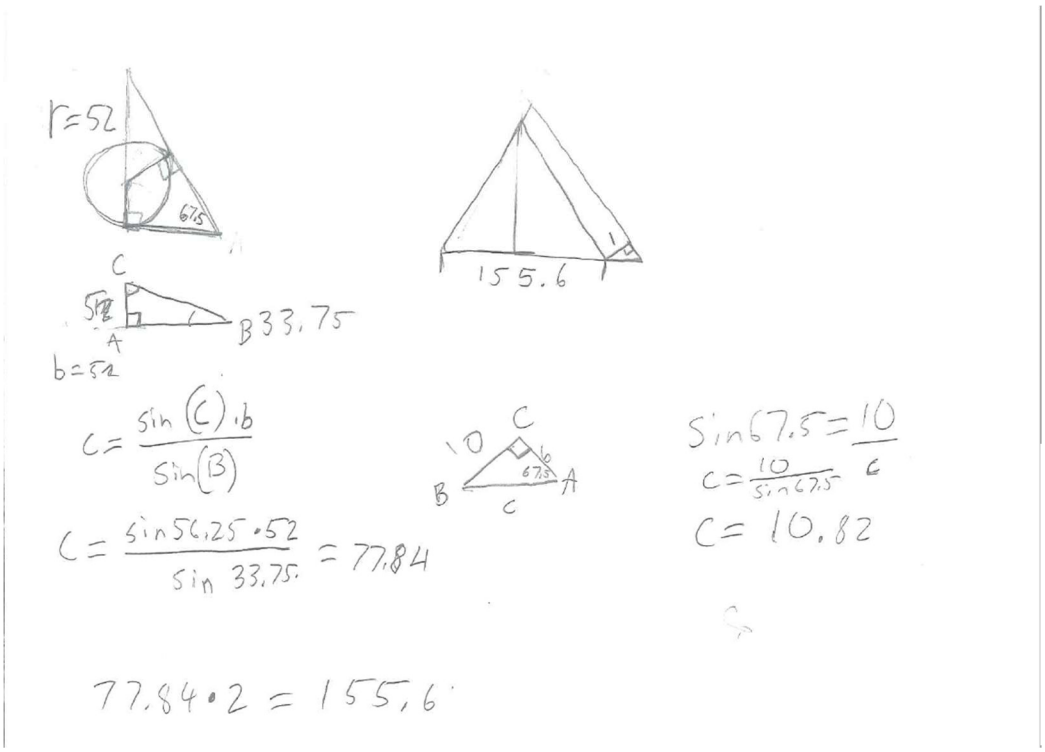
There were several variations of BOB as the design process progressed. Each variant had differences in structure, style and usability. During the design process, the team discussed the strength and weaknesses of certain design aspects. At the beginning, the design was a full-scale ideal project. From there we would scale down the size and limit functionality to what was required and what could be implemented in the prototyping time frame. Working backwards from a master design kept the project true to its original intent. As an autonomous bar robot, the primary functions encompass the ability to navigate a bar, to serve drinks and to be as/or more convenient than a traditional waiter. Some development changes as an example: were that it would have 2 driven wheels and 1 castor, in a tricycle configuration, while in the end it would be a system of 2 drive wheels with the wheels centered around the vertical pole of the robot. With 2 castor wheels, front and back. Additionally, the sensor arrangement went through several configurations. One iteration would have a lidar sensor on a spinning pole. It was changed to having multiple ultrasonic sensors mounted around the body. It was changed because the ultrasonic sensors would have better coverage and be more robust. Ultimately the final design was chosen for its simplicity and functionality.

As it stands, the robot is comprised of an octagon spaceframe fabricated from aluminum t slot. Separated into two sections, upper and lower, drink serving parts and drive unit respectively. Sensors and cover plating are mounted on the outside. Going up from the ground, the wheel, motors, batteries, and electronics are installed inside lower compartment.

5. Mechanical Design

5.1. Frame Design

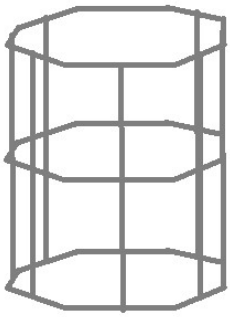
After some discussion, it was concluded by the group that the ideal shape of frame to use in order to maximize space for the drinks was an octagon. The dimensions of this octagonal frame were decided based on the average dimensions of a 2L soda bottle, which were found online. The goal was to design a frame that could house six bottles vertically within it, with some extra space. This extra space was decided to be 10 mm. the dimensions of the frame could then be defined using these parameters.

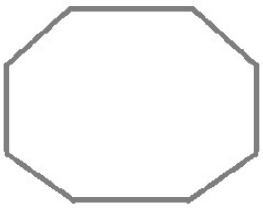


The octagonal frame is made up of 8 equally sized triangles, with dimensions dictated by these calculations

The frame would then be comprised of three octagonal frames, attached vertically at equal spacing by lengths of T-slot. The bottom frame would be for the hardware, the center frame for the drinks tray and the top frame would provide access to the drinks.

Early frame concept

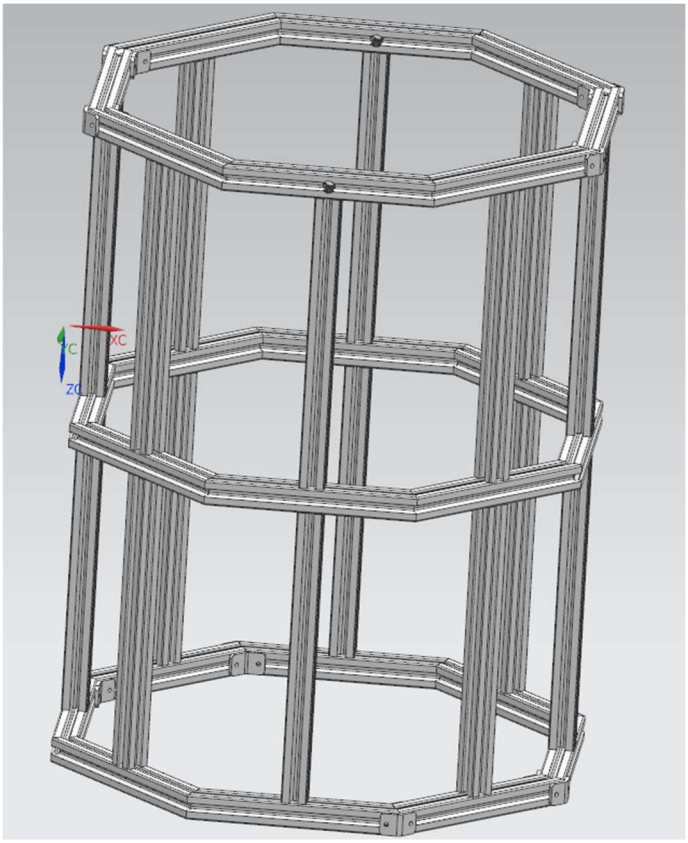




The height of the frame was decided to be 530 mm to keep proportions with the width. This would make the vertical lengths of profile 237.4mm, since they are 20 mm in width.

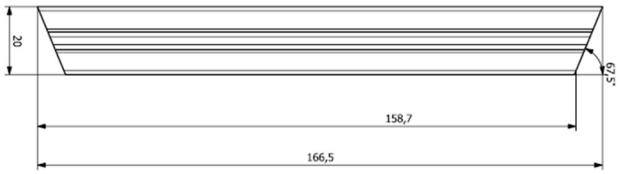
Knowing there would be a fair amount of weight for this frame to sustain, it was decided that aluminum would be the best material to use. Specifically, 20mm x 20mm aluminum T-slot profile, as this was ideal for construction of frames.

NX model of finalized frame design

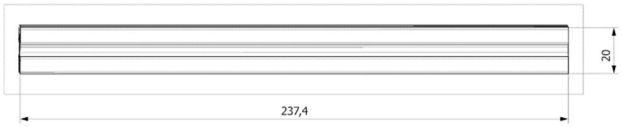


Then began the long process of manufacturing the many components of the frame, which meant a lot of tedious cutting. In total, 24 octagonal segments and 16 column segments would have to be manufactured.

Octagonal frame segment

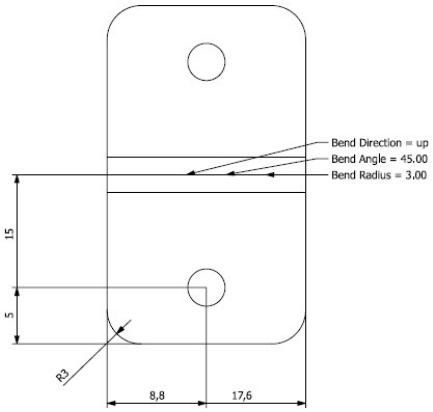


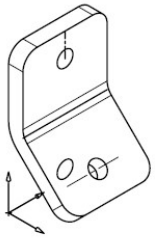
Vertical Beam frame segment



To join the octagonal frame segments, a number of small brackets would have to be made. Each of these brackets would subtend to a 45 degree angle and join each corner of the octagons.

brackets to join the octagonal frames





This was a convenient solution for the Aluminum profile used as they could be attached externally by slotting a nut into the profile and screwing the brackets into position. Initially, it was planned that both the inside and outside of each octagonal vertex would be joined with this method, which would require 48 brackets. This was a task completed in majority by the water jet cutter to save time. In the end though, it was decided that using only external brackets would suffice, the extras where convenient for a number of other tasks, however.

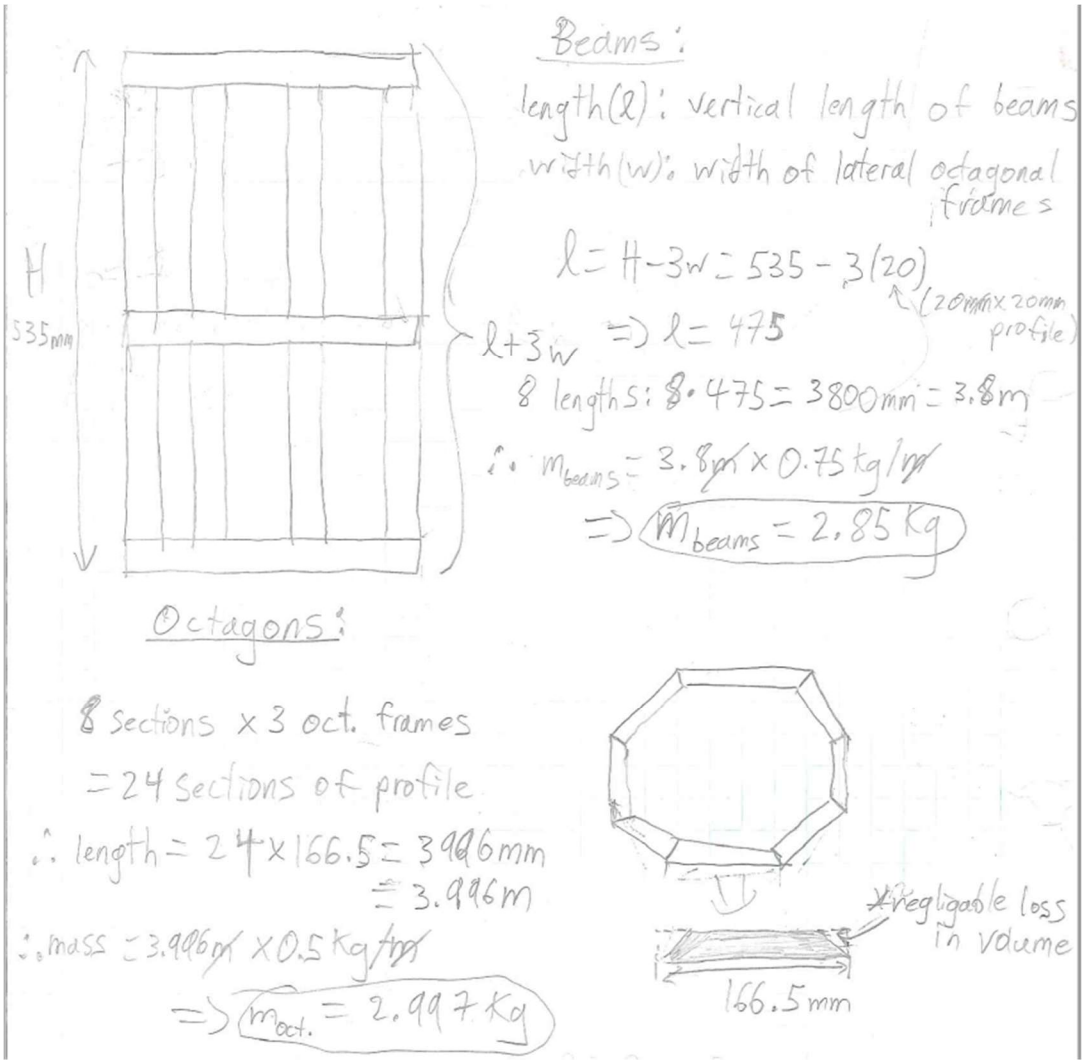
Although, the group quickly agreed on an octagonal frame design, there was also the option to construct a far more simple cubic frame. Although this would have been a lot easier and less time consuming, it certainly would not have been ideal for storing the round bottles for the drinks.

5.2. Wheel Assembly Design

The first step in mobilizing B.O.B required the group to select which type of motors should be used. It likely wouldn't matter which motors where chosen as the group were fairly confident that most of the motors in the parts list would be have been able to drive the robot, with varying degrees of torque of course. Rather than choosing at random however, the group formulated a design requirement that BOB should be able to move at least as fast as the average human walks, Online sources (Wikipedia) stated this speed to be approximately 1.4 m/s (*"Visual flow influences gait transition speed and preferred walking speed",* William B. Thompson*)*. As well as that, the robot should be able to achieve this speed in under 5 seconds, therefore an acceleration of at least 0.28 m/s^2 was required.

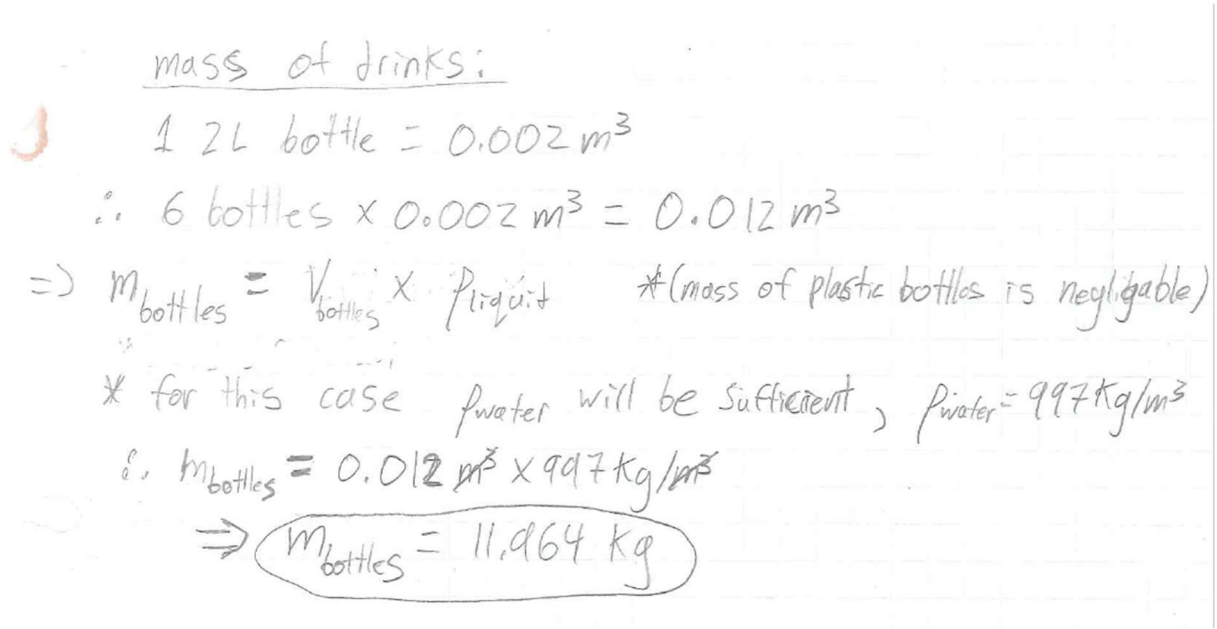
In order to calculate the device's acceleration, there were a few necessary parameters to identify; the wheel size and the approximate weight of the device. Determining the wheel size was a straight forward task as only two sizes of wheel where available, in the end a wheel of 170 mm in diameter was chosen. Approximating the device's mass would prove a more trying task. As the dimensions of the frame were known, its approximate mass could be estimated by using an online source which provided the mass per length of the aluminum profile being used (http://www.aluminium-profile.co.uk). For a similar 20mm x 20mm profile to the one being used, the site claimed an approximate mass of 0.75 kilogram per meter of length.

Frame mass calculations



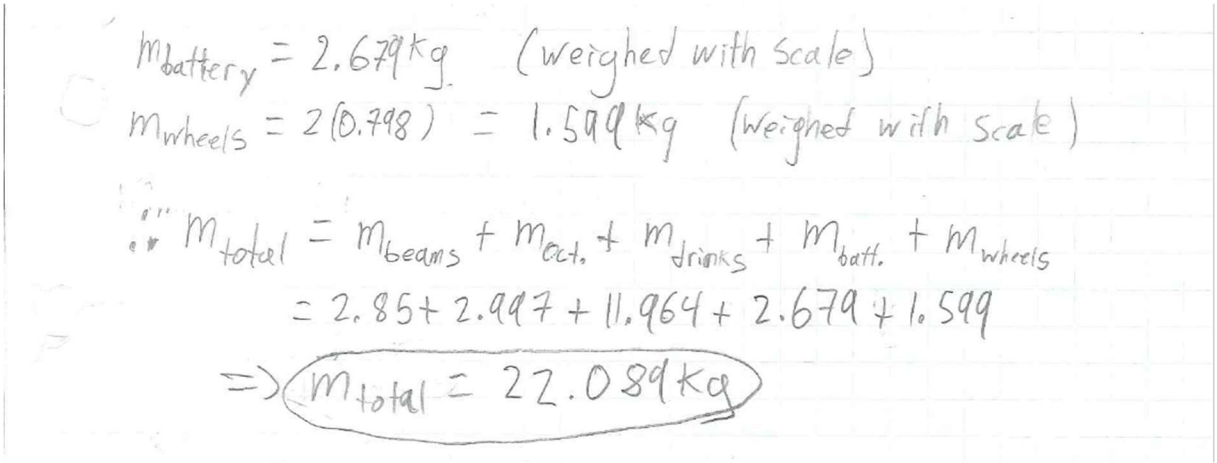
As well as that, it seemed the another considerately massive component would be the full bottles of beverages. They were roughly estimated based on their volume (2L) and the density of the liquid inside them, using the density of water as a substitute. It was agreed that the bottles themselves had a negligible mass.

Bottle mass calculations



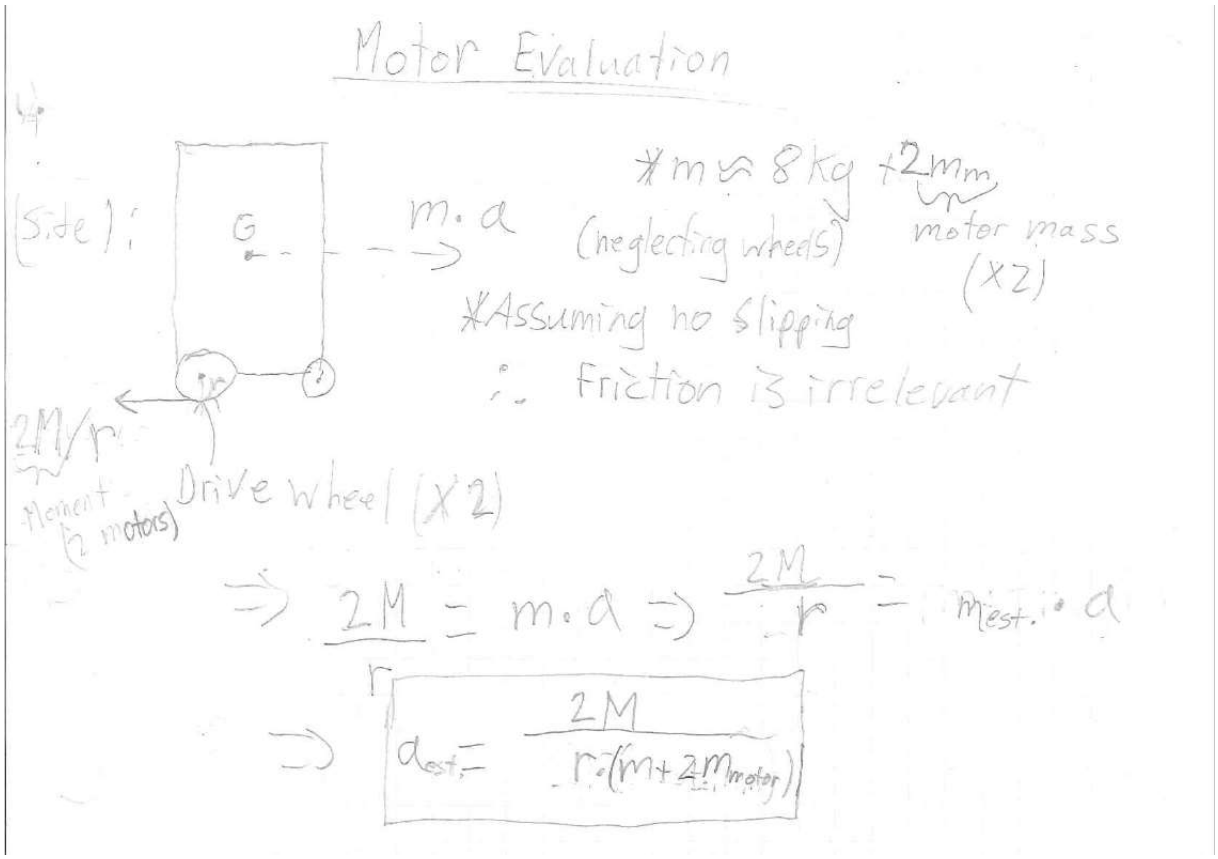
Finally, there were a few other components that seemed important to include in the estimate

Total mass calculations



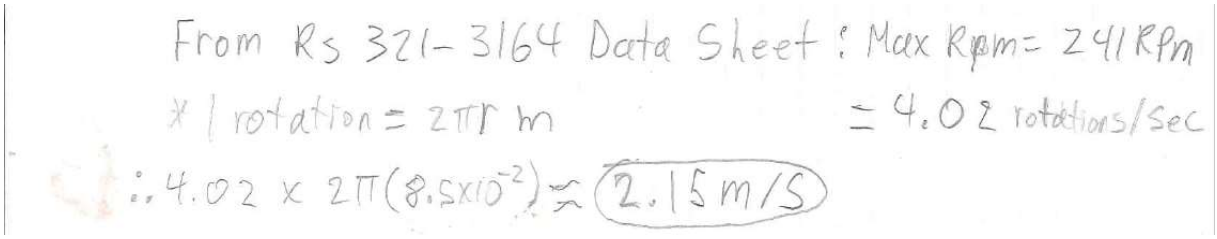
As the max torque for each motor was listed online, the maximum acceleration of the device could be analyzed in each case. This was done with a standard formula based on the setup of the robot. The decided upon design incorporated a two motor, skid steer design with two other caster wheels that stabilized it as well as granted the ability to "turn on a dime." Not that it made a huge difference, but it is worth mentioning a small amount of inaccuracy in the calculations since the moment of inertia of the wheels were not taken into account.

Standard Acceleration formula



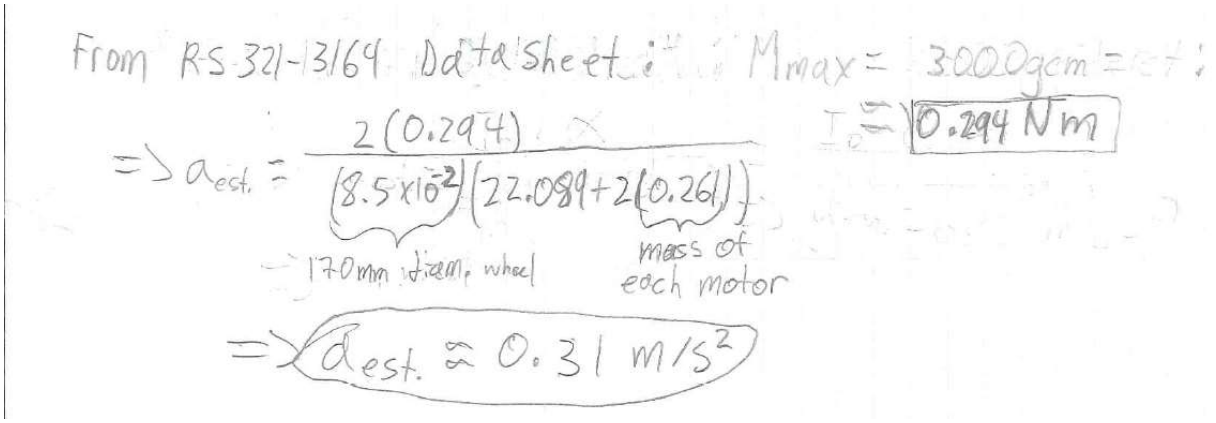
After applying this formula to each motor, the RS 321-3164 motor seemed like a good fit as it was solid enough to support the weight of the robot and met the necessary requirements. Primarily, its maximum output RPM provided a speed similar to a person walking.

Speed calculations



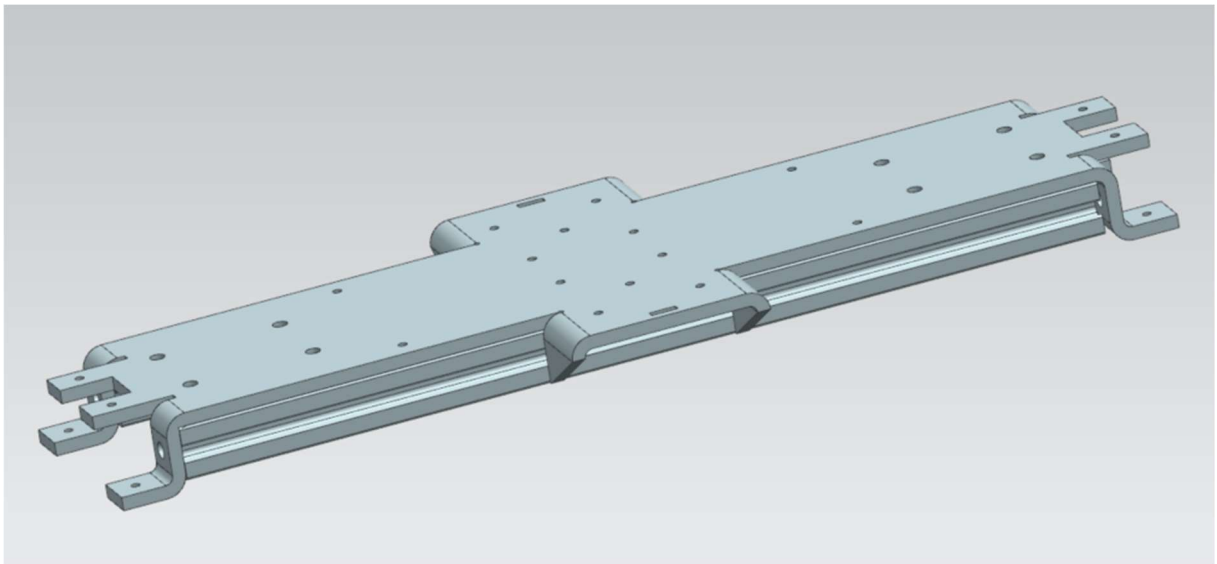
To verify its listed max torque, a short test was conducted. In this crude test, a wheel with a certain radius was attached to the motor axle and then used to attach a mass via a string hanging tangent to the wheel. After that mass was added to the other end of the string until the motor was incapable of lifting it. A somewhat unprofessional test, but it was enough to convince the group of the validity of the listed max torque. After that, the maximum acceleration could be calculated with the previously determined standard formula.

Acceleration calculations

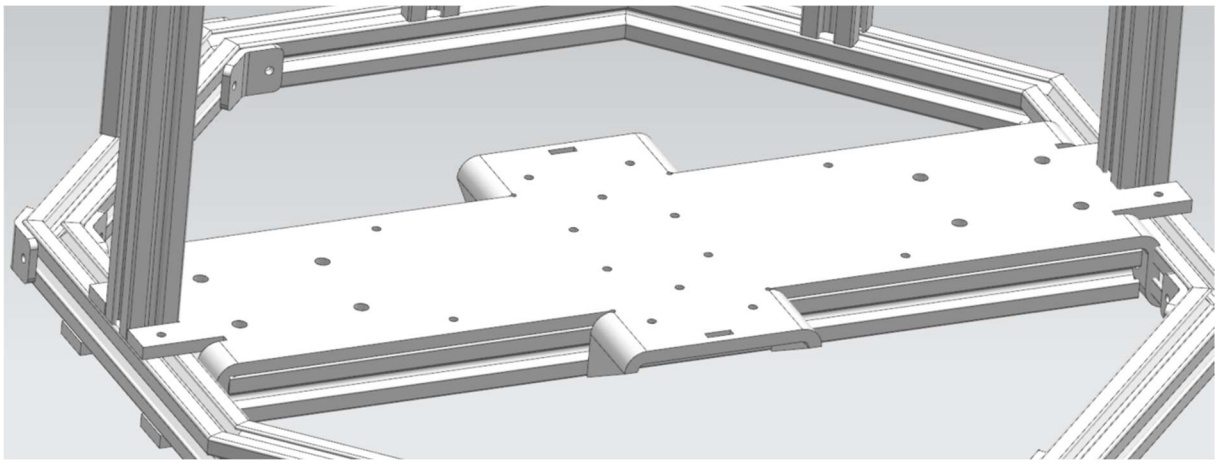


In order to now mount these motors along with their attached wheels, there was a few suggested solutions. The thought initially was to mount them to frame itself, it was soon realized there was little space to do this. It was then concluded that the best way to mount the motor/wheel system was to create a mount that spanned across the center of the device, supported on each side by opposite ends of the frame. This meant the wheels would be enclosed inside the frame, rather than sticking out the sides. As this would be the point where most of the normal force would be concentrated, it was important to the group, that it was a solid design. This was achieved by using a combination of 3mm aluminum plate and laterally supporting aluminum profile beams.

Motor/Wheel mounts NX model

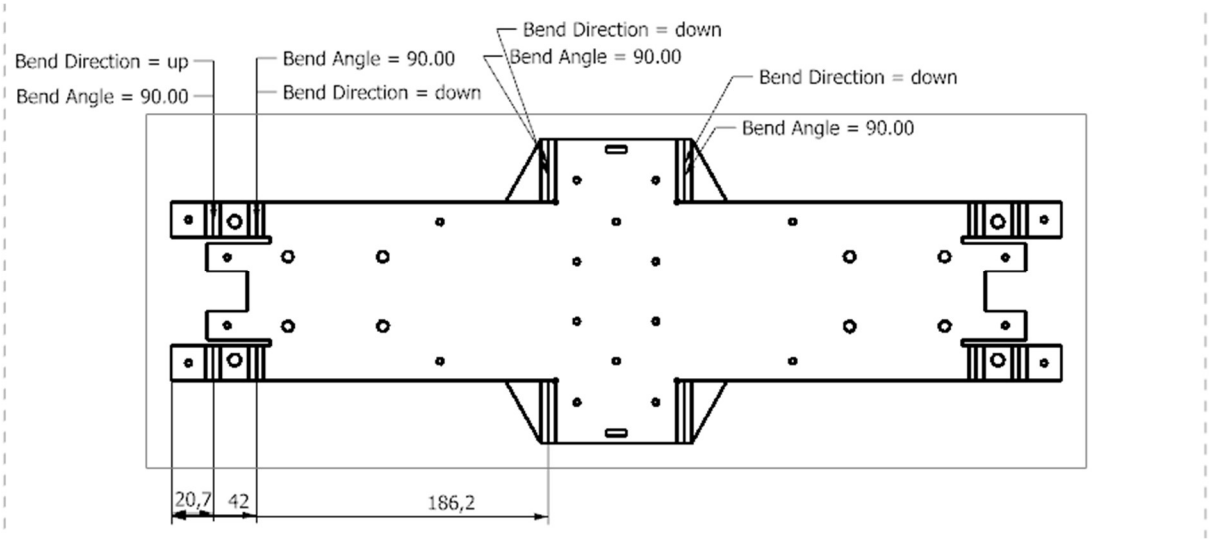


The left and right end would be slotted into the sides of the frame from the inside as shown below.



Being a fairly complex piece, it was decided it would be best to commission this piece to be cut by the waterjet cutter. This left only the bending to be done by hand. The DWG file used by the CNC system was based on a top view of the unbent sketch.

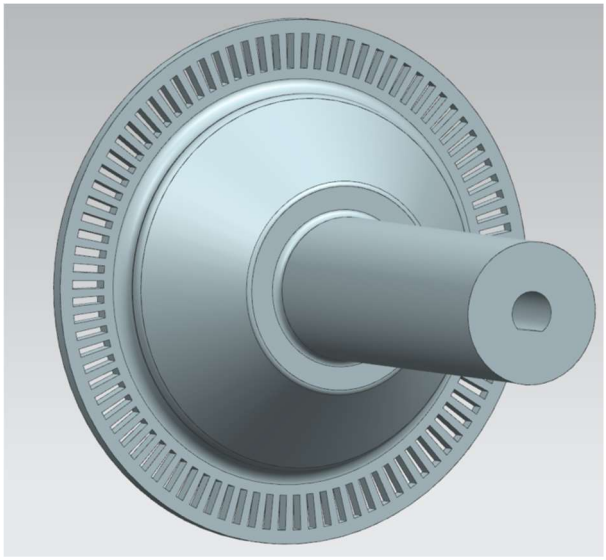
Motor mount NX draft



The purpose of the slots on the edge of the motor mounting area were to secure a cable tie across the bottom beams in hope of increasing rigidity. This feature was incorporated out of fear that the plate would bend due to the upward moment caused by the contact of the wheels to the ground, which were attached to the motor, attached to the plate. In the end it seemed to make more sense simply to attach the entire mount upside down, which the design allowed for.

In order to attach the wheels to the motor shaft a sort of axle adapter was required, as the wheel itself had a large hole through it. The best way to fabricate such a part was to 3d print it. That way the part had the functionality to lock the motor shaft and wheel using a D-lock and a locking screw. Additionally, the rotary encoder wheel was designed onto the axle as one part; saving time and improving strength. Lastly, the axle had another hole, opposed axially from motor, to press fit a steel shaft to mount a bearing. This supported the wheel from both sides.

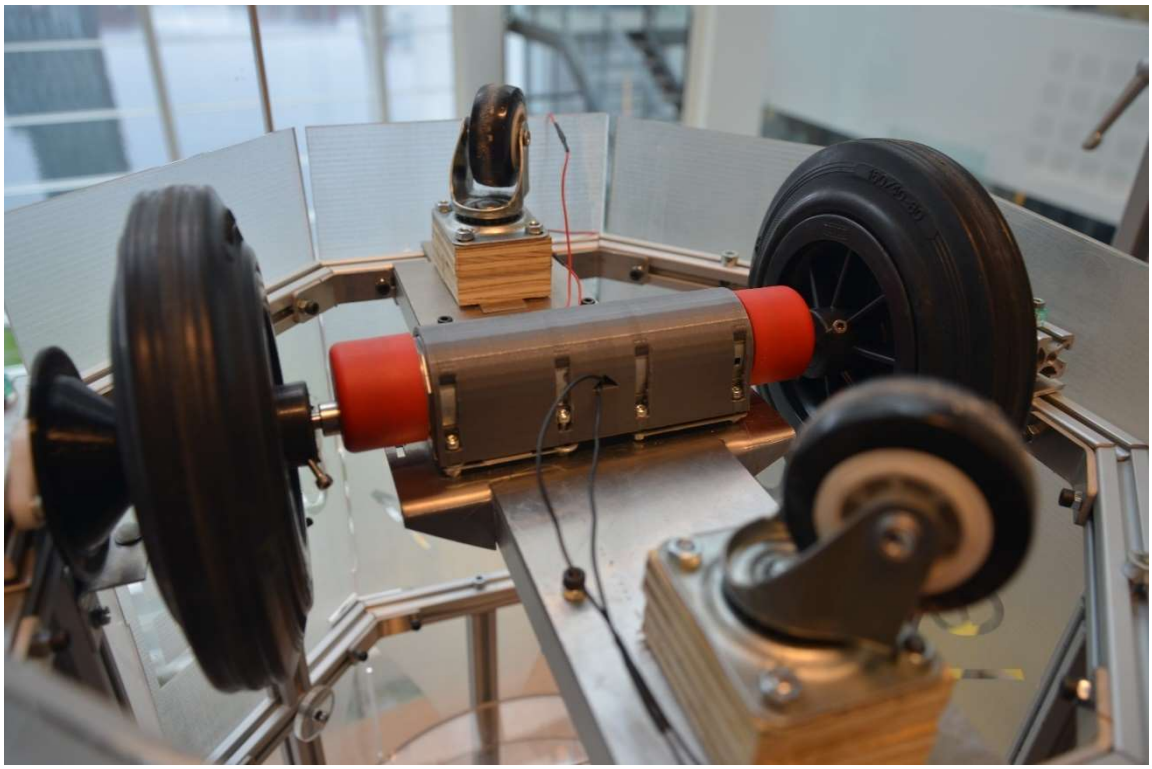
Axle adapter



The smaller diameter of this adapter could then fit tightly enough through the wheel hole that a friction fit was created.

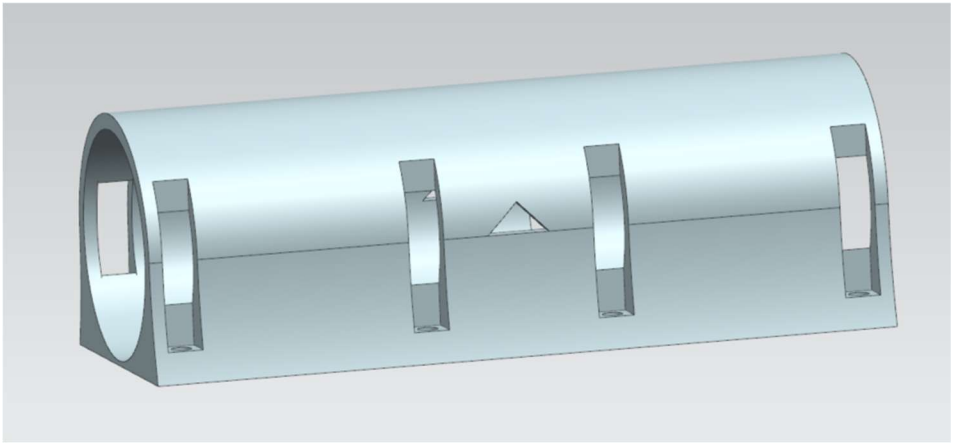
After the motor mount, motors, axle adapters and wheels were secured, the wheel assembly was complete.

Finished underside picture



A number of other modifications can be seen in this image as the perpendicular mounting brackets that came attached to the motor were incredibly flimsy. To remedy this, a motor housing was designed which connected the two motors together as well as fasten them parallel to the motor mount.

Motor Housing NX model



This was then printed in 25% density infill to ensure it remained intact.

As well as this, another measure was taken to ensure the axles were straight and rigid. A makeshift solution secures the outside ends of the axle to the inside of the frame. This was done with the unorthodox method of attaching sections of aluminum profile to the undersides of the frame, spaced low enough for the axles using a number of small aluminum plates as spacers. Large plastic exterior bearings were then attached to the sections of profile and housed the ends of each axle.

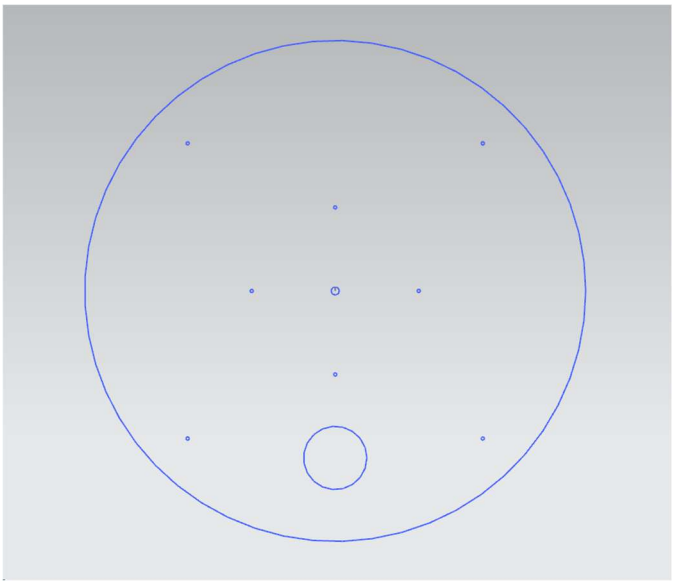
This last-minute improvisation seemed to cause some issues though. The rather inaccurate method of lining up the bearings to the axles meant that one of the axles was apparently straighter than the other. This created varying degrees of friction, which caused one wheel to rotate faster than the other. The hope was that the code would be able to compensate for these uneven rotational speeds by counting the rotations and adjusting. It did cause some frustration though.

5.3. Bottle Plate

5.3.1. Design

The plate that holds the bottle was designed in Siemens NX and was made by combining two circles. The outer circle is 360mm in diameter; the little hole is 60mm in diameter. The little hole is designed to hold the base of the cup. The two discs were designed to be put on top of each other and screwed together with the eight small holes, four located near the center and four others located near the edge of the disc. The top disc would have the small 60mm hole and the bottom disc would not have that hole, that way the cup does not fall through the discs.

A servo was to be placed under the middle of the disc, fastened to the frame of the robot, this servo would rotate the disc. Small wheels where also attached at four different parts of the frame under the disc these wheels hold the disc and keep it stable as it rotates.



Disc's design in NX

5.4. Construction

The finished discs.

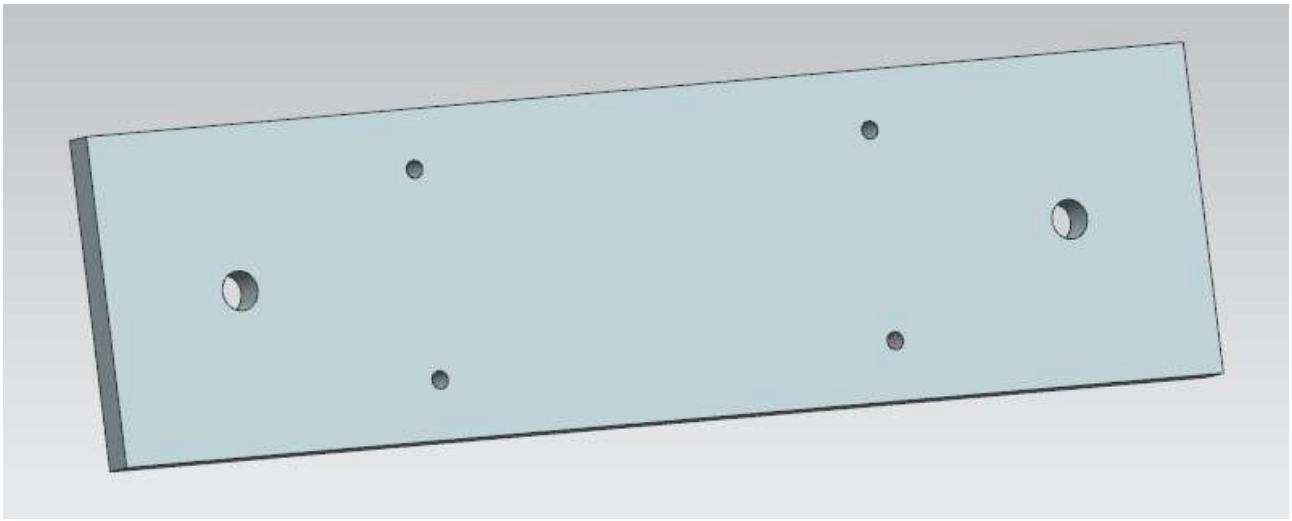
The two plates where cut out on a laser cutter and made in acrylic. It has a thickness of 5mm. The wheel mounts are the brackets used for holding the frame itself together, bend to 90degrees. The wheels are attached to those brackets by screws. The servo mounts is made in 5mm acrylic and is attached to the frame under the disc. It has been geared up to allow for more than 130degrees of rotation.

Alternatives:

An alternative to this system could have been to make the bottles rotate instead of the bottom plate. That would have made it harder to turn, as most of the weight was already in the top plate that holds all the full bottles. Another way of doing it would be to open up tubes, which lead to a single cup instead of rotating anything. However, that would require more vertical space than there actually is inside that part of the robot.

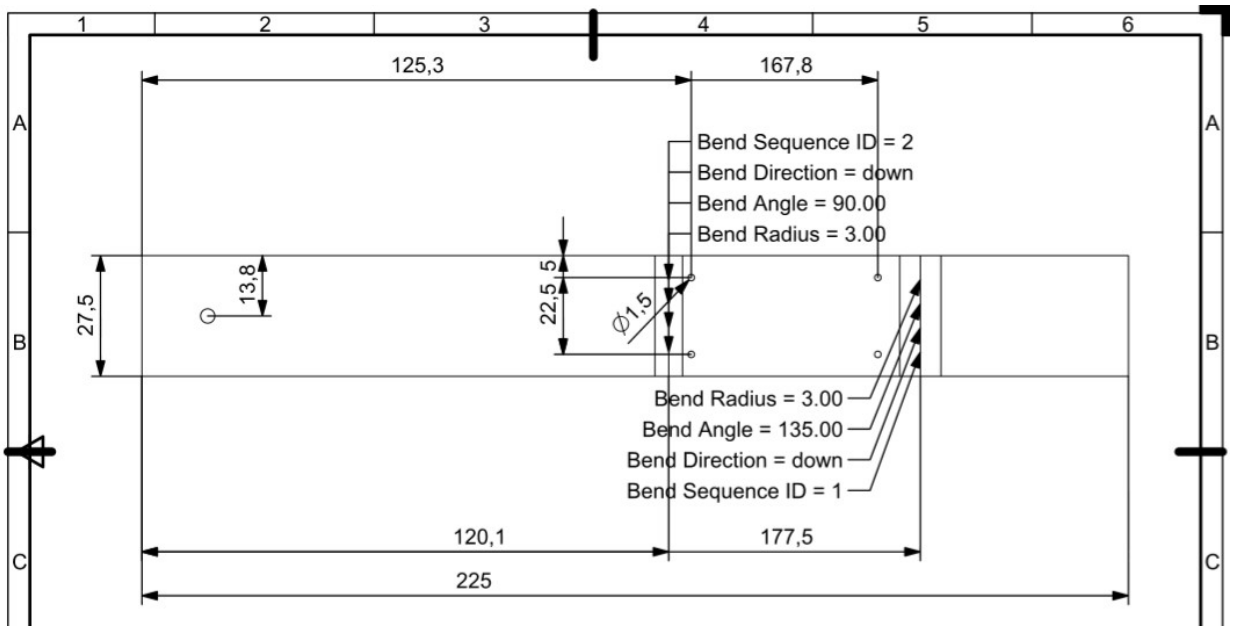
Sensor mounts:

The sensor mounts is made as a 102.5mm by 27.5mm sketch, with holes for the sensor to be screwed onto. The mount itself is made to be connected to the middle of the frame with two bolts. The sensor is mounted to the plate with plastic spacers so that the board does not touch the metal mount.



3D model of sensor mount for sides of the robot.

Two different mounts were made for the right and left front sensor. These mounts connect to the sensor the same way as the mount mentioned above. It connects to the robot differently due to the angle of the frame it is mounted on. This sensor is 125mm long and is bend in two places to allow it to be attached to the frame.



Draft of right and left front sensor mount.

Construction:

The mount is made of 3mm thick aluminum cut into the right size and the holes needed were drilled. For the left and right front sensor mount the metal plates was bend into the specific shape designed in NX.

Finished side and front left sensor mounts.

Alternatives:

Instead of making the mounts in aluminum plates, they could have been 3D printed in PLA. By making it that way, it could have been avoided to use spacers between the sensor and the mount. However making it that way would have taken longer for very little benefit.

6. Hardware design

6.1. Motor control

6.1.1. Problem background

A motor control board is needed when a larger DC motor is being used, because the current draw and back EMF can destroy a microcontroller without any kind of shield. So, a motor control board can work both as a shield to the microcontroller and the rest of the circuit but also as a H-bridge. An H-bridge is a component which can switch the polarity of a voltage applied upon the motors. This way, the motors will be able to both drive forwards and backwards (Sparkfun).

6.1.2. L298N Dual H Bridge

The L298N Dual H bridge is a combination of a motor shield board and a motor control board. As seen in Figure 1 Schematic of a L298N Dual H-Bridge, on the right side there are diodes, which function is to make sure the back EMF is not returned into the circuit but rather to +12V and then to ground.

Schematic

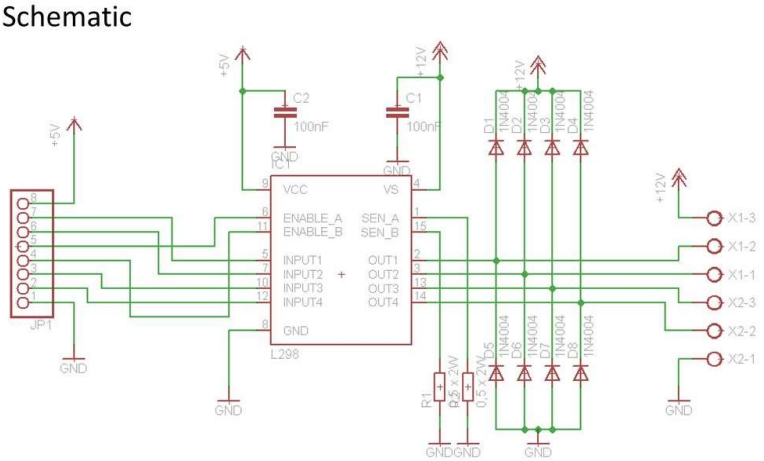
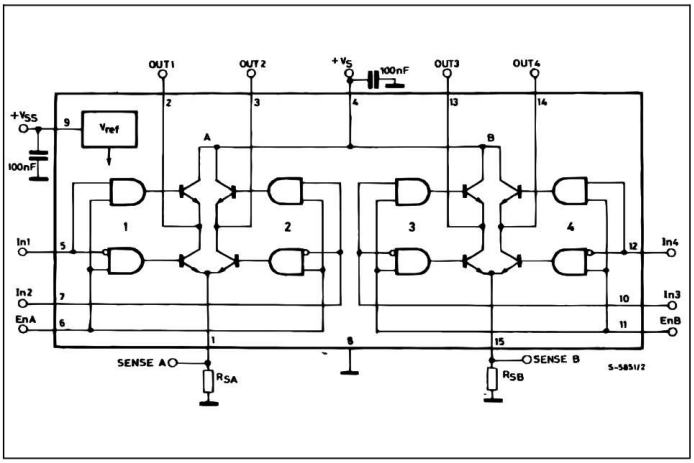


Figure 1 Schematic of a L298N Dual H-Bridge

The L298N also is a motor control board, which is controlled by logic. The block diagram of such logic can be seen in Figure 2 Block diagram and logic table in L298N motor control board. Here it can be seen that each motor will run independently of the other, which is a very useful feature because it will give the opportunity of the robot to rotate both to the left and right.
A problem with the L298N was that the group only used one for two motors. This meant a current of larger than 2 amps were flowing through the chip and into the motors, at initial acceleration. The maximum current rating of the chip is 4 amps (Sparkfun), but for many stops and starts, the chip/heatsink got very hot and therefore not able to deliver the desired current to the motors. This problem could be solved with two L298N chips instead of one.

BLOCK DIAGRAM



ENA	In1	In2	Out1	Out2	Motor 1	Led1	Led2
L	X	X	Z	Z	Stop	Off	Off
H	H	L	Vin-Vdrop	L	Forward	On	Off
		H	Z	Z	Stop	Off	Off
	L	L	Z	Z	Stop	Off	Off
		H	L	Vin-Vdrop	Backward	Off	On

ENB	In3	In4	Out3	Out4	Motor 2	Led3	Led4
L	X	X	Z	Z	Stop	Off	Off
H	H	L	Vin-Vdrop	L	Forward	On	Off
		H	Z	Z	Stop	Off	Off
	L	L	Z	Z	Stop	Off	Off
		H	L	Vin-Vdrop	Backward	Off	On

Figure 2 Block diagram (Sparkfun) and logic table (Instructables) in L298N motor control board.

6.1.3. 12-bit PWM board PCA9685

The group decided to use a PWM board. This decision lies upon the fact that the code is using a lot of interrupts and therefore had problems with sending a constant signal to the L298N. With an external signal board, the signal only must be sent to the PWM board once. The PWM board will then send the constant signal of desired duty cycle to the motor control board. A PWM board is useable if the full speed of a motor is too fast and therefore wanting to decrease the speed. This is done by giving a specific duty cycle for which the motor will be running. By giving a duty cycle of for example 50%, the given voltage to the motors will be 6 volts (if the voltage source is 12 V) (Sparkfun), an example of this can be seen in Figure 3 An example of a duty cycle and how the average voltage is measured.

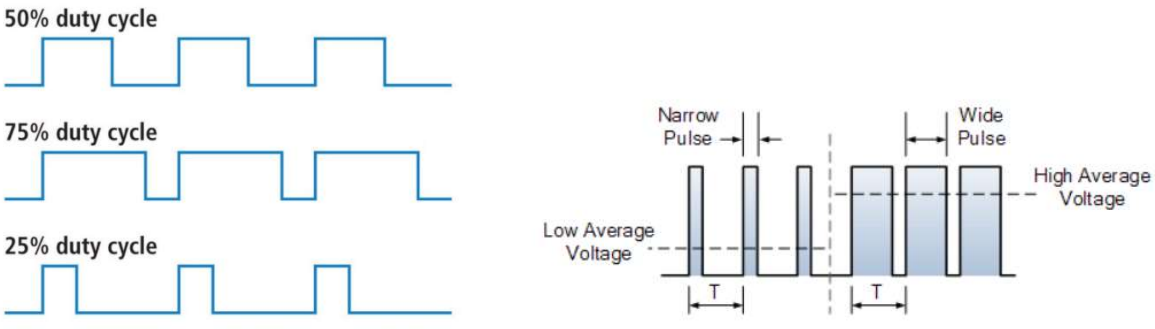


Figure 3 An example of a duty cycle and how the average voltage is measured.

The group chose to use the PCA9685 PWM board. This board is specifically made to control LEDs, but the configuration is also a great option for a motor control PWM signal.
The PCA9685 is used by connecting “Enable A” And “Enable B” from the L298N to Port 3 and port 1 respectively. This way the PWM-board will have turn off/on the motors with the desired speed.

6.1.4. Alternatives

Alternative solutions could be making a custom H-bridge. By making a custom PCB, many connections can be implemented. This would also give an opportunity to fix the problem of the escalating increase in high temperatures by implementing a larger heatsink or two L298 chips instead of one. The Figure 1 Schematic of a L298N Dual H-Bridge is a start of a PCB layout.

6.2. Sensors

6.2.1. Problem background

When the motor and motor control board is selected, sensor must be chosen. To make a robot able to sense and/or interact with the surroundings, sensors are needed. There are many forms of applications of sensors, but for this project, only two types are used, an ultrasonic and an optical sensor.

Ultrasonic sensors use sonar to measure the time for which a signal take to “hit” a surface and bounce back. The sensors will, in this project, function as an object detector, where six will be contributed around the robots’ surface.

Optical sensors work by sending a constant signal of, for example, light to a receiver and then count every time this signal is obstructed. These sensors will function as a counter for rotations of the wheels.

6.2.2. Ultrasonic sensors

6.2.2.1. Problem background:

An HC-SR04 is a specific ultrasonic sensor, which works by receiving a signal, for at least 10 μs, on the trigger pin, the module will then automatically send eight signals of 40K Hz (ELEC Freaks) and receive them again in the “echo” pin, as seen in Figure 4 Timing diagram of the HC-SR04. From the module is activated to the signal is received again is the time it will calculate the distance from.

$$Distance = \frac{High\ level\ time * Velocity\ of\ sound\ (340 \frac{m}{s})}{2}$$

Here the high-level time is the time it takes from activation to receiving the signal. The high-level time multiplied with the velocity of sound is divided by 2 because only the distance to the object is needed.
An example of how the distance effect the echo, can be seen in Figure 6 Oscilloscope of ultra-sonic sensor: Yellow is trigger and green is echo, where a hand was waved around the ultrasonic sensor. From the figure, it can be seen the distance is proportional with the width of the echo pin.

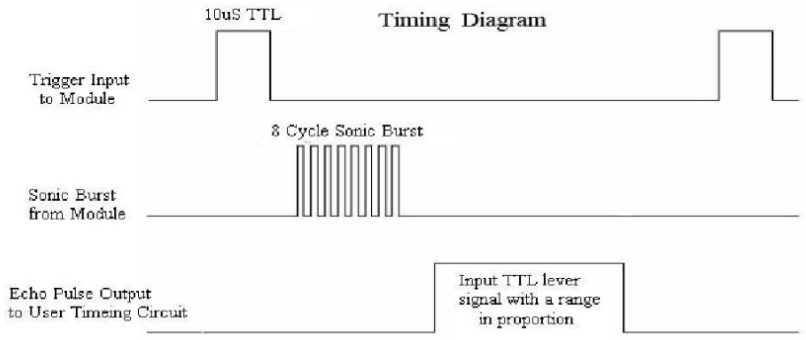


Figure 4 Timing diagram of the HC-SR04 (ELEC Freaks)

6.2.2.2. Tests

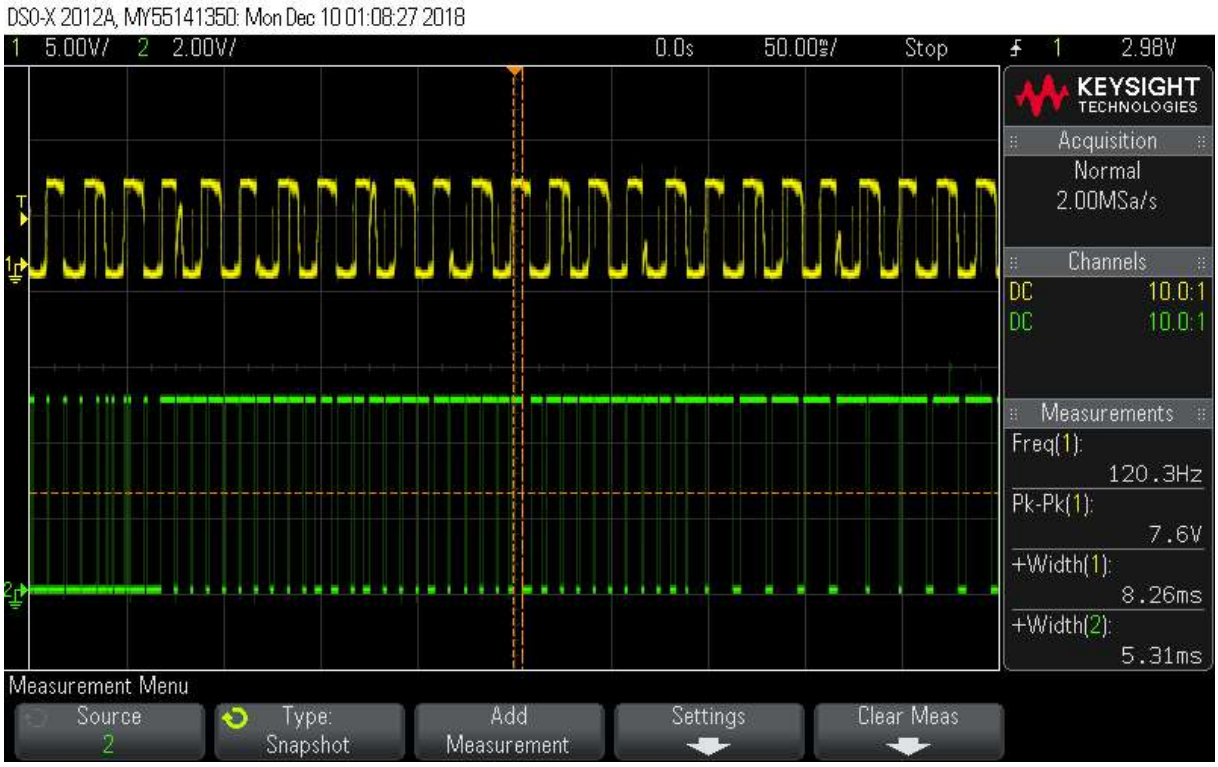


Figure 6 Oscilloscope of ultra-sonic sensor: Yellow is trigger and green is echo

	Distance (in cm)												
Nr.	6	10	11	20	21	30	40	50	60	70	80	90	100
1	4146	8495	9544	16619	19104	28225	36102	45231	54452	64064	8027	16821	26338
2	4119	8469	9516	16593	19123	28194	36078	45629	54424	64031	8003	17171	26308
3	4119	8468	9517	16592	19124	28197	36075	45586	54426	64038	7997	16792	26301
4	4119	8517	9519	16594	19125	28198	36071	45207	54426	64035	8002	17173	26305
5	4120	8943	9517	16971	19124	28195	36077	45589	54419	64035	8004	17173	26299
Median (seconds):	4119	8495	9517	16594	19124	28197	36077	45586	54426	64035	8003	17171	26305
Distance (cm)	4,42	9,11	10,20	17,79	20,50	30,22	38,67	48,86	58,34	68,64	8,58	18,41	28,20
Error in %	35,8987	9,822790631	7,8322021	12,443607	2,4463331	0,7400142	3,439266726	2,3279742	2,8491463	1,9850433	832,5949	388,99315	254,66436

Acceptable error range (E<13%)

Unacceptable error range (E>13%)

Timer's period is 1/16MHz		Wnat we actually get are the cycles	
1 cycle =	6,25E-08	time =	cycles*period(our median)

Figure 5 Testing of ultrasonic sensor with different distances and the percent error.

Some testing was done on the ultrasonic sensor to test the precision of output at different distances. In the datasheet it is mentioning to be a minimum range of 2 cm and a maximum of 4 meters (ELEC Freaks). The experiment was intended to figure the lowest and highest possible distance for which laid in between an acceptable error of less than 13%. Even though the datasheet gave a value between 2 cm and 4 m, only values between 10 cm and 70 cm had less than the acceptable error range, this can be seen in

As it can be seen in the figure, the median value for 80 cm is plainly wrong and therefore highest likely a human mistake, which means the ultrasonic sensor maybe have a higher acceptable distance.

6.2.2.3. Alternatives

An alternative to the ultrasonic sensor could be a LIDAR. This sensor works much like the ultrasonic sensor but instead of ultrasonic waves, this sensor sends light and measure the distance that way. The LIDAR that could be used is VL6180. The sensor is very precise in measurement but as seen in the datasheet (Adafruit), the measure range is between 0 and 10 cm, which is not good for a relatively fast-moving robot. The sensors schematic is as seen in Figure 7 Schematic of VL6180.

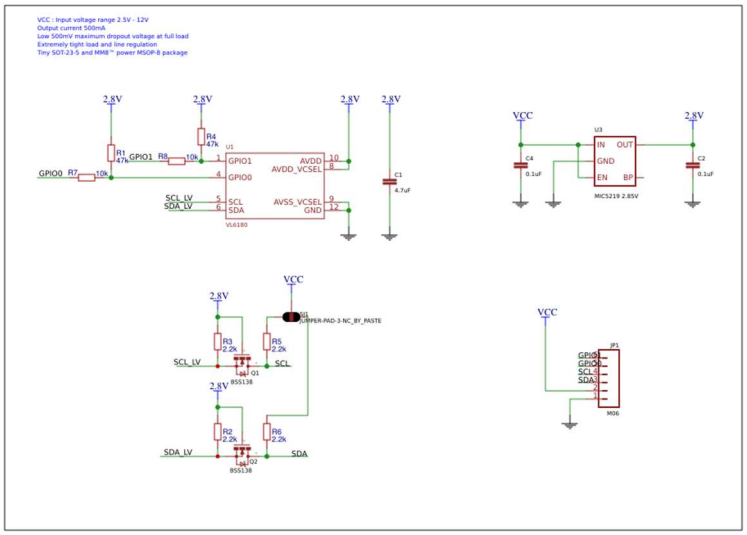


Figure 7 Schematic of VL6180

6.2.3. Optical sensors

6.2.3.1. Problem background

When talking about optical sensor, a lot of different types exist. Here the focus will lay upon an IR optocoupler, a schematic of such component can be seen in **Error! Reference source not found.**.

If an object, like a card, "*is placed in the sensor slot, the beam of infrared light from the LED to the phototransistor is blocked, turning off the phototransistor. Its emitter terminal, which has been at a high voltage of about 4.8V because the phototransistor has been in saturation and therefore conducting heavily, turns off and the emitter voltage falls to a low value of around 0.8V.*

This turns off the 2N3904, which causes its emitter to fall to around 0.3V. This is seen as a logic 0 … Removing the object from the slotted switch allows the infrared light to reach the base of the phototransistor causing it to conduct and saturate again, which now has a voltage of nearly 5V at its emitter. This switches on the 2N3904 causing a logic 1 (a voltage greater than 2V).”(Learnabout electronics).

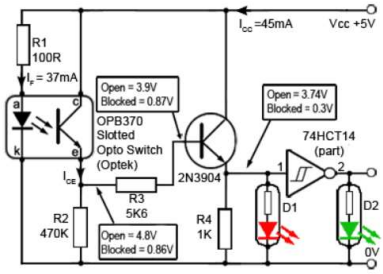


Figure 8 Slotted optical switch circuit

The optical sensor will function as a counter of the motor adapter, which can be seen in mechanical part. The concept behind counting the holes is to measure how long the robot have travelled, because a full rotation of the wheels is 90 holes. This also give the opportunity to rotate by a certain angle, which is crucial if a coordinate system must be followed.

6.2.3.2. Tests

Some testing was made on the optical sensors, to see the difference between analog and digital output.
The difference can be seen that with the digital pin a high-low output, which can be seen as 1 and 0 respectively, is given.
The analog gives instead a steady square wave output, these differences can be seen in Figure 9 Analog vs digital output of optical sensor. The yellow (1) is analog output and green (2) is digital output.

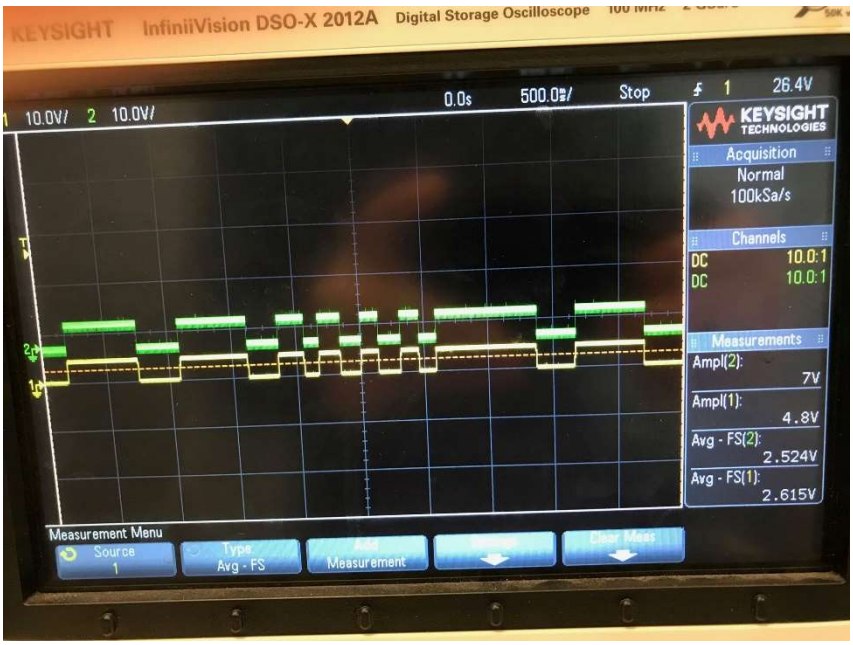


Figure 9 Analog vs digital output of optical sensor. The yellow (1) is analog output and green (2) is digital output.

6.2.4.Alternatives

An alternative to an optical sensor is difficult to find, instead it would be a great alternative to replace the DC motors with a pair of stepper motors. This replacement would mean the optical sensor is not needed, because the a stepper motor knows, by design, how much of a rotation is done and therefore easy to calculate the angle/distance it have travelled.

6.3. Buttons

Buttons are the main tool to initialize the device. This is an answer for one problem that was encountered during development process – it was the start and stop commands. These should be independent and work even if the device is already doing some task.

Also, in further design the buttons would be used as a user interface in order to select the drink.

6.3.1. Selection of components and alternatives

For this purpose, the microswitch with long arm was selected.

Alternatives:

In the early stage of the concept design there was an idea of using Bluetooth as a remote control. But since there was no person with experience in this field the idea was neglected.

Another alternative was to make a device start after some predeclared time interval for instance 10 seconds. But it would make the device uncontrollable since the only way to start/stop it would be the manual power supply alteration – connection and disconnection respectively.

6.4. Microcontroller

6.4.1. Background

When all the electrical components are selected, it is time to choose a microcontroller which have the required memory, speed and pins for project.

Initially a single Arduino Nano was used, as seen in Figure 10 Schematic of circuit with one Arduino Nano, this quickly figured to be a too small microcontroller, as it had too few pins.

Therefore, the idea was to use two Arduino Nanos, which would work as master and slave, the schematic can be seen in **Error! Reference source not found.**.

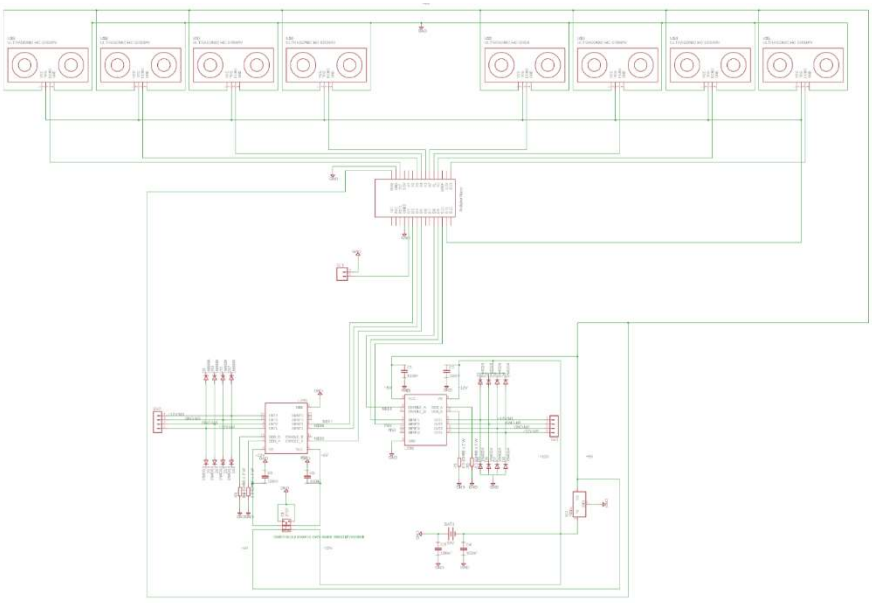


Figure 10 Schematic of circuit with one Arduino Nano

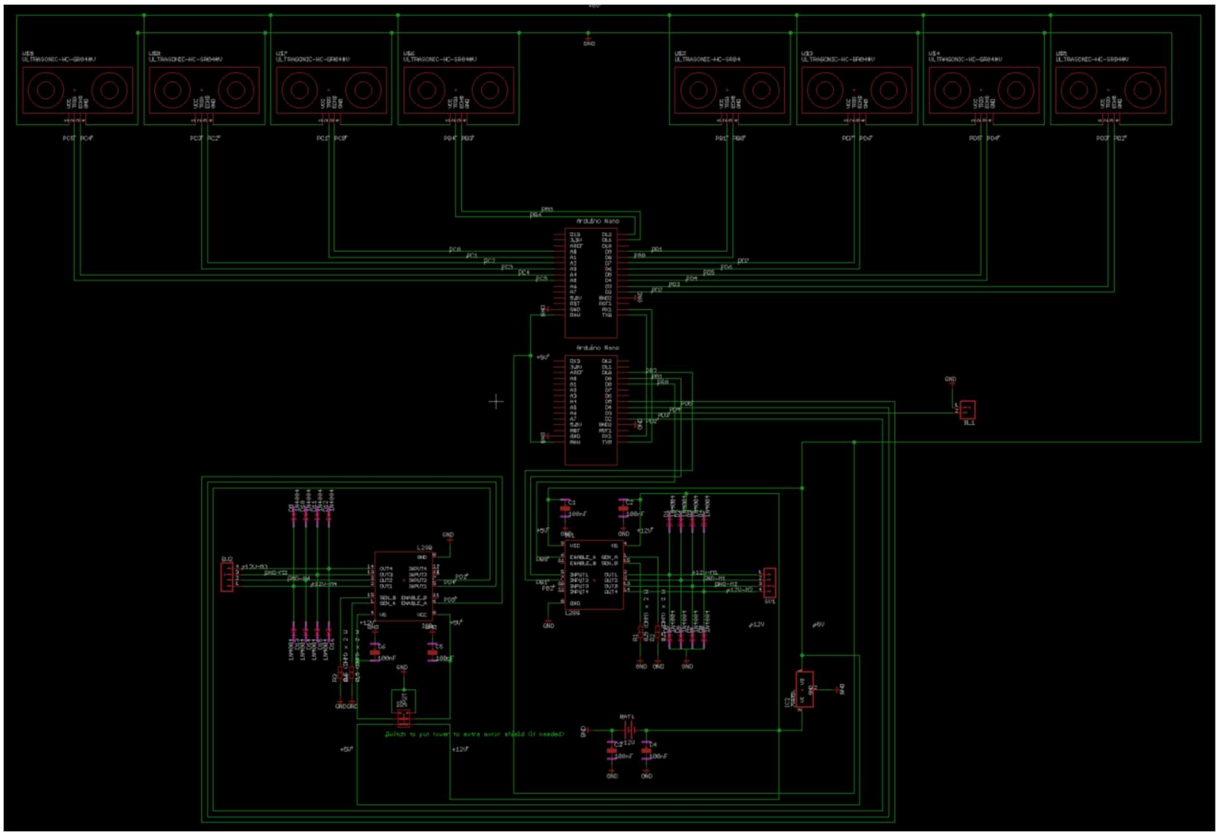


Figure 11 Schematic of circuit with two Arduino Nanos

For both the convenience and learning experience the chose was laid upon the Arduino Mega 2560. The difference between ATmega328P (Arduino Nano) and the ATmega2560 can be seen in Figure 12 ATMEGA 328P and ATmega2560. Beside these, one of the more important differences is the number of interrupt pins; the ATmega 328P only have 2 interrupt pins whereas the ATmega 2560 have 5 interrupt pins.

Features	Arduino Uno	Arduino Mega (1280)	Arduino Mega 2560
Microcontroller (MCU)	ATmega328P (Datasheet)	ATmega1280	ATmega2560 (Datasheet)
Operating Voltage of the Microcontroller	5 V	5 V	5 V
Typical Supply Voltage for the board	7 V – 12V	7 V – 12 V	7 V – 12 V
Digital I/O Pins	**14 (includes 6 PWM outputs)**	**54 (includes 15 PWM outputs)**	**54 (includes 15 PWM outputs)**
PWM outputs	**6**	**15**	**15**
Analogue Input Pins	**6**	**16**	**16**
Max DC Current per I/O Pin	20 mA	20 mA	20 mA
Max DC Current for 3.3V Pin	50 mA	50 mA	50 mA
Flash memory of MCU	**32 KB**	**128 KB**	**256 KB**

Figure 12 ATMEGA 328P and ATmega2560

A microcontroller is the brain behind an electronic circuit and it is crucial that it has the capability to handle the program that is needed for the specific task.

This project will handle six ultrasonic sensors, two optical sensors, one/two motor control board and one PWM board. This will require a lot of code, which means a high amount of memory and many pins. The ATmega 2560 have 54 pins, whereas 15 are PWM PINS, and 256 KB of flash memory.

The circuit for which an ATmega 2560 is used, can be seen in Figure 13 Schematic of circuit with an ATmega 2560Figure 13 Schematic of circuit with an ATmega 2560 and it's pin out in Figure 14: Table of connections.

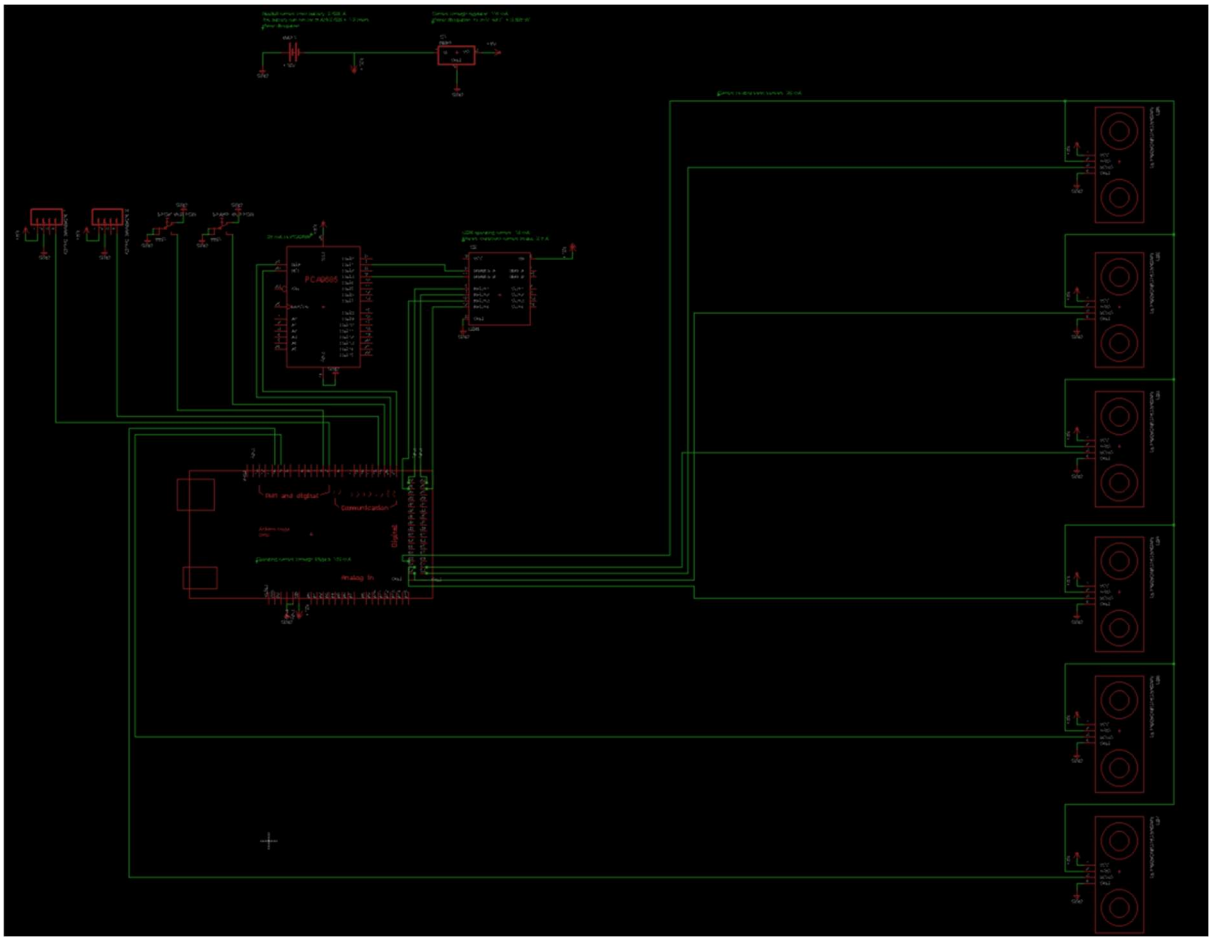


Figure 13 Schematic of circuit with an ATmega 2560

6.4.2. Alternatives

An alternative to the ATmega 2560 could, as mentioned earlier, be two Arduino Nanos, one slave and one master. By having a slave Arduino all the interrupts could be programmed into it and only send the necessary data to the master Arduino, the schematic for such circuit can be seen in Figure 11 Schematic of circuit with two Arduino Nanos.

6.5. Battery selection

6.5.1. Problem background

For a moving robot, a stationary voltage source is not recommended and therefore batteries are the best solution to this. Batteries work in many ways. The group focused on lead-acid batteries, because they are inexpensive and easy to recharge (not dangerous), compared to, for example, a lithium battery.

To know how big of a capacitance the battery need, the following equation can be used:

$$C = x * T * 0.8$$

C: $Capacity\ in\ Ah$
x: $Current\ drawn\ in\ amps$
T: $Time\ in\ hours$
0.8: Constant for which the battery is safely fully discharged (A lead acid battery begin to degrade at that point)

6.5.2. Calculations

The equation above can be manipulated so the time of which a certain battery can supply the circuit is. The given battery has a 9Ah, with a steady +12V output, the current drawn from the circuit is 2.6A, if the robot is travelling at full speed. As for the current drawn is only when travelling at full speed, it will be considered a constant current consumption, this could be considered a safety factor. So the maximum hours the battery can supply the circuit is:

$$T = \frac{C}{x * 0.8} = \frac{9}{2.6 * 0.8} = 2.7\ hours$$

This means the robot should be able to run for 2.7 hours non-stop, which is more than enough for the project.

7. Embedded design

Embedded design is a stand-alone part of the overall control of the device electronics. This phase is dedicated to handling the hardware elements such as sensors, motors and various boards all together.

7.1. Embedded structure

Embedded structure follows the design of pin connection for all the hardware elements. This is necessary for the system as it enables the control and further development of the system. With these connections the priority and the final function of the system was decided.

7.1.1. Connection design

There were various elements to take care of, therefore the list of them is as follows:

ARDUINO MEGA

Elements to consider:

- 2 buttons for start and stop
- 6 ultrasonic sensors
- 2 optical sensors
- 2 motors

Possible extras:

- 4-8 buttons to order a drink
- 1 extra motor/servo
- 1 extra optical sensor

Electronics circuits used:

- 12-bit PWM board – PCA9685
- Double H-bridge board – L298N
- Specially designed PCB

Having all these elements which yields 17 direct pin connections to the Arduino Mega board and 12 Ground (GND) pins and 9 connections to +5V and 1 connection to +12V.

The function of each element was considered.

For that reason, it was concluded that the stop button should have a highest priority after optical sensors, as it is the tool to stop device when necessary. In comparison the start button is not of the same priority - it is just meant to initialize the process and after that all the other elements take the responsibility of the behavior of the robot.

Ultrasonic 1	
Trigger	DIGITAL PIN 48 / PINL1
Echo 1	DIGITAL PIN 53 / PINB0 (PCINT0)
GND	Common GND
VCC	+5 V
Ultrasonic 2	
Trigger	DIGITAL PIN 48 / PINL1
Echo 2	DIGITAL PIN 52 / PINB1 (PCINT1)
GND	Common GND
VCC	+5 V
Ultrasonic 3	
Trigger	DIGITAL PIN 48 / PINL1
Echo 3	DIGITAL PIN 51 / PINB2 (PCINT2)
GND	Common GND
VCC	+5 V
Ultrasonic 4	
Trigger	DIGITAL PIN 48 / PINL1
Echo 4	DIGITAL PIN 50 / PINB3 (PCINT3)
GND	Common GND
VCC	+5 V
Ultrasonic 5	
Trigger	DIGITAL PIN 48 / PINL1
Echo 5	DIGITAL PIN 10 / PINB4 (PCINT4)
GND	Common GND
VCC	+5 V
Ultrasonic 6	
Trigger	DIGITAL PIN 48 / PINL1
Echo 6	DIGITAL PIN 11 / PINB5 (PCINT5)
GND	Common GND
VCC	+5 V
Optical sensor 1	
Analog pin 1	DIGITAL PIN 19 / PIND2 (INT2)
Digital pin 1	Not used
GND	Common GND
VCC	+5 V
Optical sensor 2	
Analog pin 2	DIGITAL PIN 18 / PIND3 (INT3)
Digital pin 2	Not used
GND	Common GND
VCC	+5 V

Connection	PINS
PWM Board	
SCL	DIGITAL PIN 21 / PD0 / SCL
SDA	DIGITAL PIN 20 / PD1 / SDA
GND	Common GND
VCC	+5 V
V+	Not used
OE	Not used
Port 3	Enable A (M1) H-bridge
Port 1	Enable B (M2) H-bridge
MOTOR CONTROL H-bridge	
Input pin 1 (M2)	DIGITAL PIN 22 / A0
Input pin 2 (M2)	DIGITAL PIN 23 /A1
Input pin 3 (M1)	DIGITAL PIN 24 / A2
Input pin 4 (M1)	DIGITAL PIN 25 / A3
+12 V	+12V supply
GND	Common GND
+5V	Not used
Enable A (M1)	Port 3 on PWM board
Enable B (M2)	Port 1 on PWM board

Buttons	
Stop button	
C	Common GND
NC	DIGITAL PIN 2 / PINE4 (INT4)
Start button	
C	Common GND
NC	DIGITAL PIN 3 / PINE5 (INT5)

Figure 14: Table of connections

As it can be seen in Table of connections the optical sensors were assigned to the highest accessible interrupts – External Interrupt 2 and 3 (Since 0 and 1 are taken because of i2c protocol connection to PWM board). Since these sensors are the main tool of measuring the location of BOB it is clearly reasonable to assign them as the highest priority – calculating the rotations of the wheels should be as accurate as possible.

Then ultrasonic sensors were attached to Pin change interrupt as the control of them will include reading form only one element at the time and so only form one pin. The time and sensor will be set in the software and for this reason this interrupt was chosen.

7.1.2. Alternatives

As it comes to alternatives the alteration in pins could be make – change the priority of elements. For example, ultrasonic sensors could be handled as external interrupts but such high priority was not necessary to these sensors.

Another possibility was to use two Arduino nano instead of Mega. Then the sensors would be split up and the highest priority elements – buttons, sensors and maybe motors would be connected to mother board. The ultrasonic sensors, due to large number if pins, would be connected to slave Arduino nano.

7.2. PCB design

7.2.1. Background

A PCB board (printed circuit board) uses conductive tracks and pads of copper luminated unto a non-conductive substrate (Wikipedia). By using a PCB board, it is possible to customize the circuit of which the robot will run upon. PCB board in this project is made in EAGLE, here the circuit is first made in a schematic and afterwards designed in a board from which the PCB will be made.

7.2.2. Design

A big flaw of the schematic seen in Figure 13 Schematic of circuit with an ATmega 2560, is cable connection between each board and component, the result of this, compared with bad cable management, can be seen in Figure 15 Cable result of circuit without PCB board.

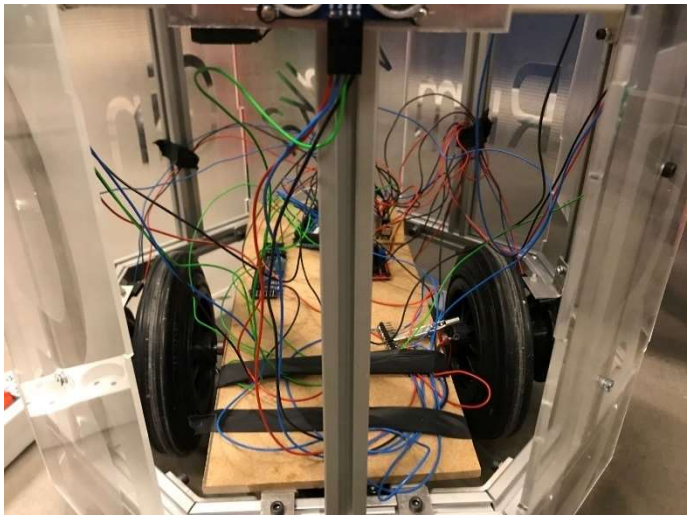


Figure 15 Cable result of circuit without PCB board

To work around the problem, the idea was to use a PCB as a “Motherboard” where all possible cable connections should be upon. The schematic and board of the first design version can be seen in Figure 16 Schematic and board layout of motherboard V.1.

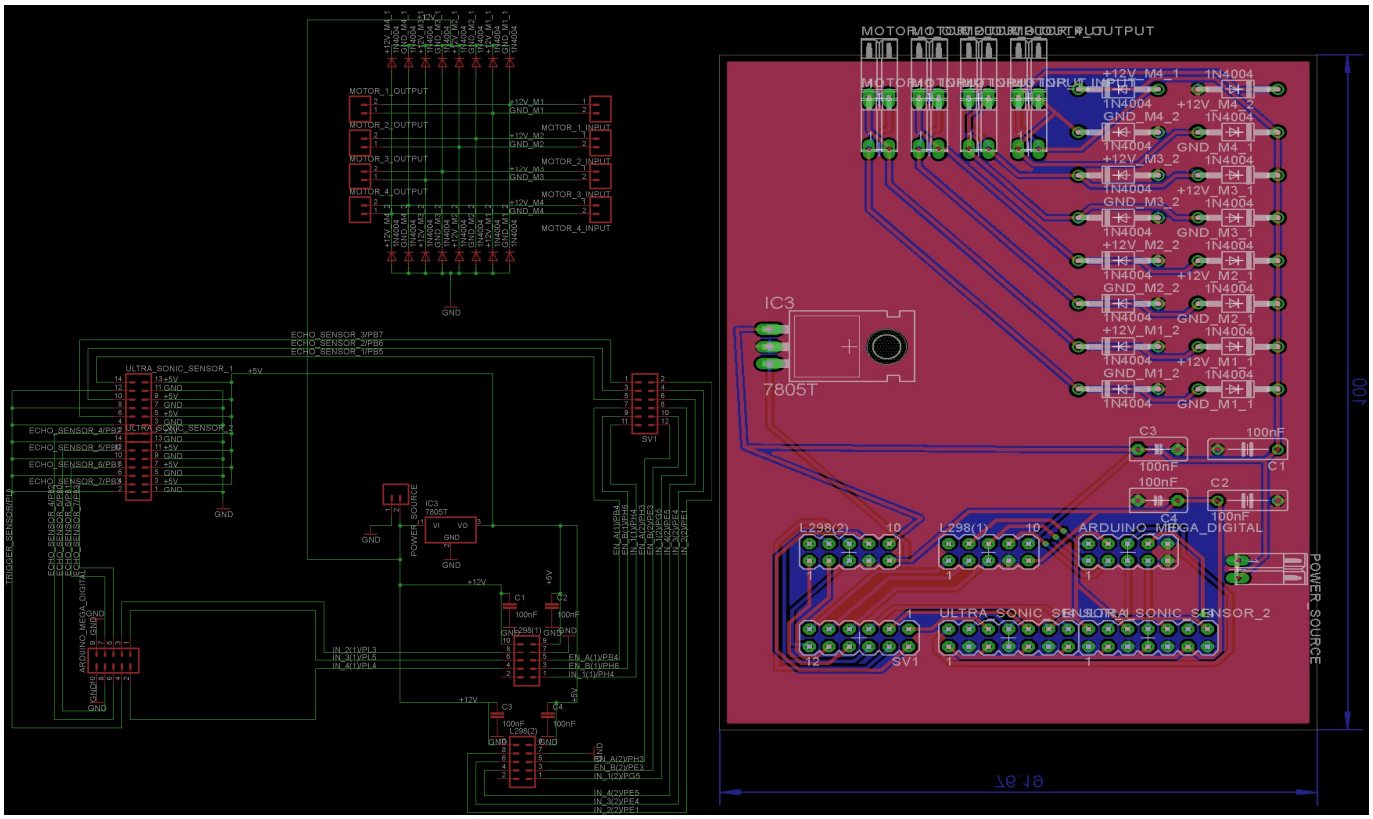


Figure 16 Schematic and board layout of motherboard V.1

The manufactured result of Motherboard version 1, can be seen in Figure 17 PCB Motherboard V.1 front side and back side respectively

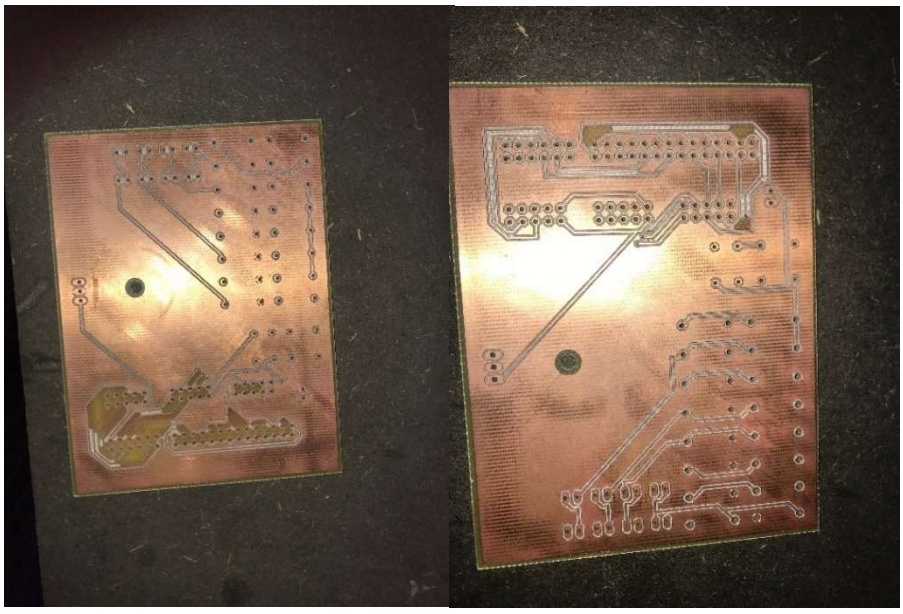


Figure 17 PCB Motherboard V.1 front side and back side respectively

Version 1 had some design flaws; The hole for the terminal blocks, which can be seen in the left corner on the back side, was too narrow and therefore not possible to solder the terminal blocks into it. Another problem was the PCB were designed with a pair of diodes for the motor control board (which the L298N already have), this mistake was a result of misreading of the datasheet. The last problem will be addressed in the next design.

After acknowledging these design flaws, a newer version was designed: Motherboard V. 2.
The schematic and board layout can be seen in Figure 18 Schematic and board of motherboard V. 2.
The resulting manufactured board can be seen in Figure 19 PCB of motherboard V. 2.

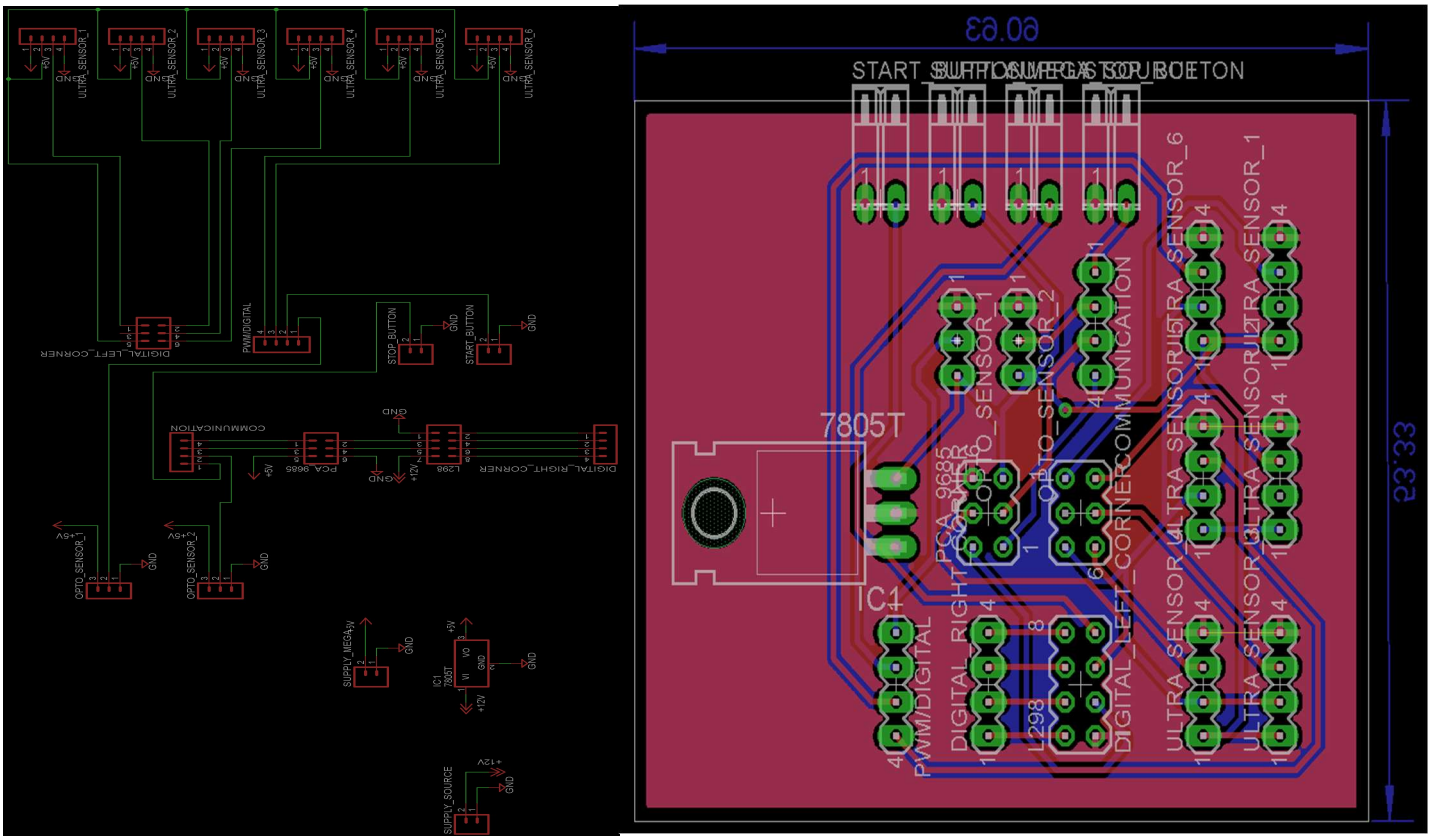


Figure 18 Schematic and board of motherboard V. 2

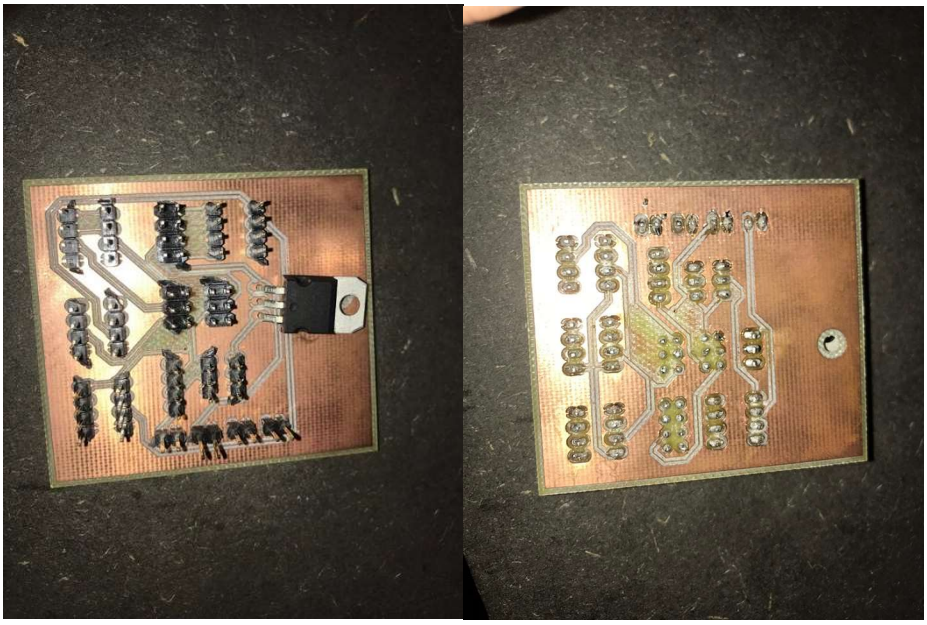


Figure 19 PCB of motherboard V. 2

As for version 1 came with design flaws, version 2 was no exception; Because the PCB was designed on a two layered board, the connection between each layer is very poor, this resulted in a very unstable voltage source and was therefore declared useless.

“There is no success without failure”, this seemed to also be the case for this project. The design flaws accompanied by the former versions was accounted for when version 3 was designed. Motherboard version 3 is a single layered PCB board, which seen in Figure 20 Schematic and board of Motherboard V. 3.

Designing a single layered board was the initial idea but have not been possible for the “Autorouter” in EAGLE to do this, therefore some manual labor had to be used. The resulting manufactured board can be seen in Figure 21 PCB of Motherboard V. 3.

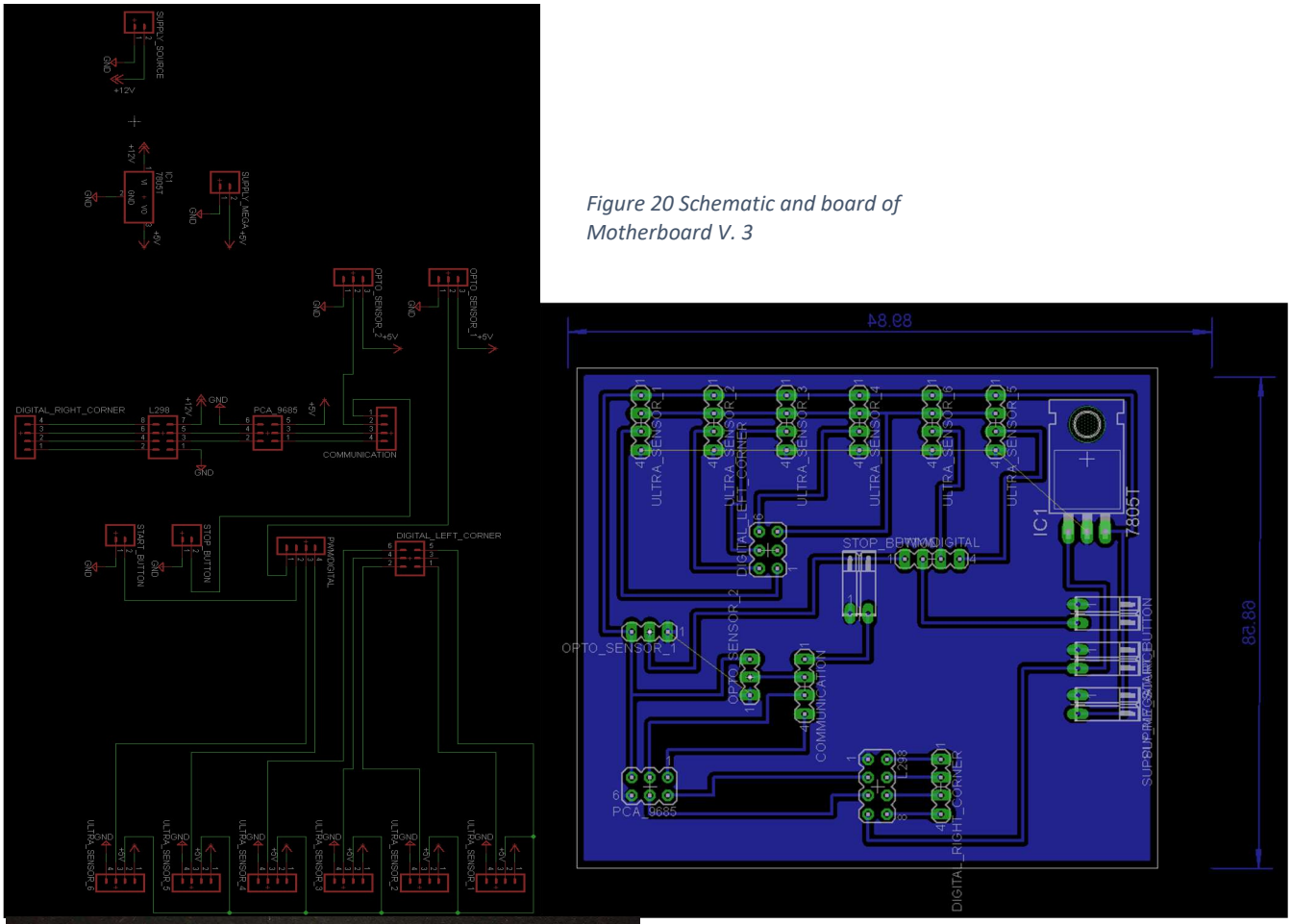


Figure 20 Schematic and board of Motherboard V. 3

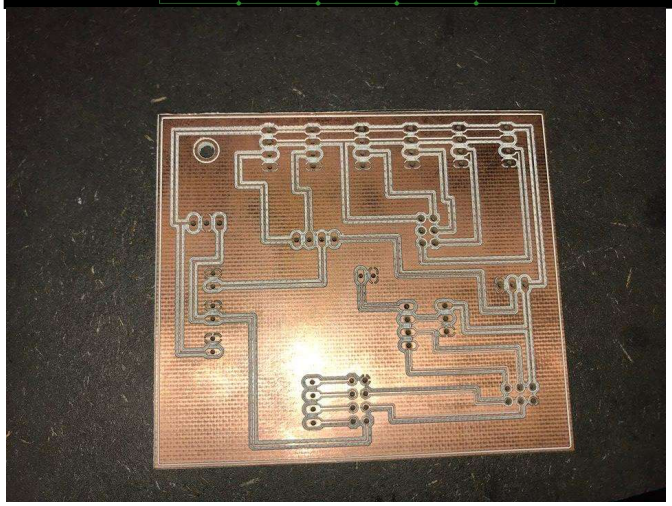


Figure 21 PCB of Motherboard V. 3

8. Software design

After designing the embedded thus hardware elements, connection etc. the software phase could begin. In this part of the design, the main code, functions and logic was created. These elements would be responsible for using and reading from sensors and for algorithms which should be executed to make the robot behaved as desired.

8.1. Block diagram

At first the theoretical algorithm of the robot was made. It is the schematic of how BOB can possibly move. This algorithm is also the prototype and mostly meant for tests if all the most important subparts responsible for autonomous behavior- initialization, sensor readings and reactions to them and movement.

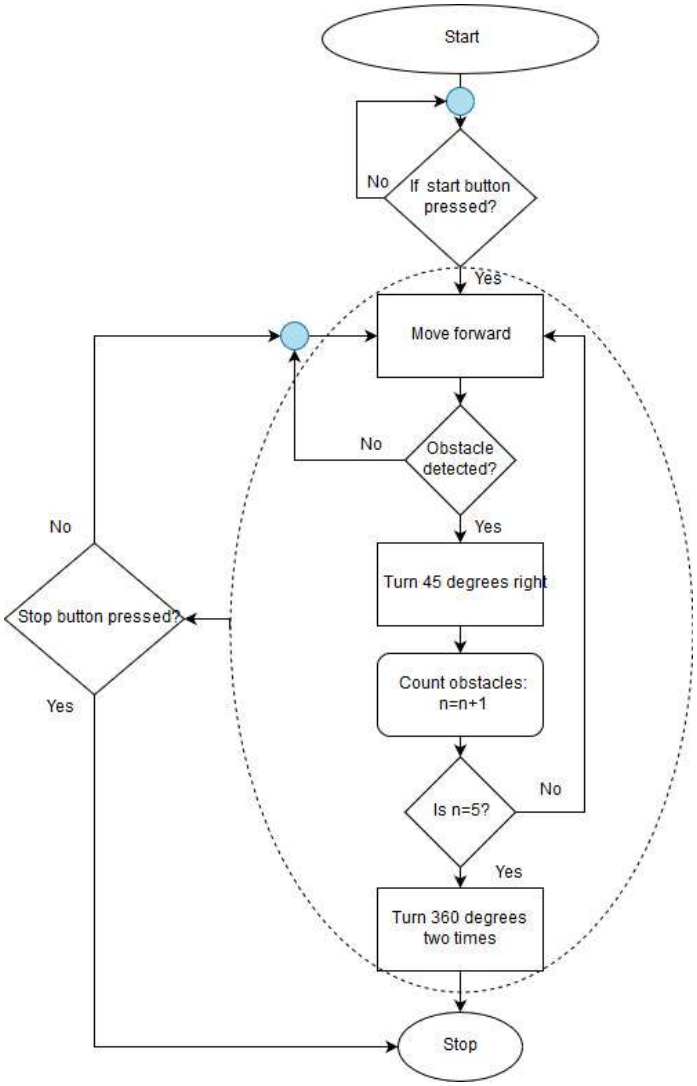


Figure 22: BOB- block diagram

8.2. Functions

Having the overall software concept in mind, the real designing had started. In this part the most important function will be presented and discussed, as they are responsible for correct behavior of the robot.

8.2.1. Mapping

Navigation code was broken down into a few general functions. First the robot starts at a home location. This would be its zero point. Cartesian coordinates would work as the primary coordinate system. This allows simplicity in adding new bar stops for the user. Then the robot would take those Cartesian points and convert them to relative points in polar coordinates, relative to the robot itself. A few benefits of this system are that it can know how far and how much to turn to get to a certain point. Self-location is done in two ways, the idea of dead reckoning is used to keep track of the robot's location i.e. plotting and updating location as it moves. As this results in position drift as more time goes by, therefor a self-correction method is needed. After a few trips the robot would return to its home station, a corner piece that makes sure that it would reset the robot's real position back to zero. When encountering an obstacle, guidance stops, then the robot navigates around an obstacle. Guidance is then recalculated with the robot's new position. The functions: Distancetodelivery; DirectionTurn; are functions that found the distance using trig functions courtesy of the math.h header file. The Cartesian coordinates of the serving point are compared to the robot's position to get a relative coordinate distance. Then the x, and y distances are put into the Distancetodelivery function which converts the cartesian coordinates into polar coordinates. As the Theta angle is the angle the target point is at relative to x axis, the direction turn function subtracts the robot's angle from the angle it needs to be facing. From this we can determine whether going left or right would be faster and how much to turn. Lastly Obstacle avoidance logic is that for each sensor there is a limit of how close an object can be before the robot will stop. Then depending what sensor is blocked it will choose the appropriate direction to turn to go around the obstacle.

8.2.2. Ultrasonic sensors

The ultrasonic functions are based on the principle of working of these sensors. For more detailed information please look in **Appendix A**

8.2.2.1. Triggering the sensor

Ultrasonic_trigger is simply, as the name suggests, setting the trigger pin HIGH for necessary amount of time which is 10 microseconds to initialize the sensor.
The argument of the function is the sensor number. It is an indication of which sensor is supposed to be read later.

```
void ultrasonic_trigger(char which_sensor)
{
    //all ultrasonics are connected to one trigger pin
    //trigger
    //We disable all echoes so that we can focus on one only
    PCMSK0 &= ~(1<< PCINT0);
    PCMSK0 &= ~(1<< PCINT1);
    PCMSK0 &= ~(1<< PCINT2);
    PCMSK0 &= ~(1<< PCINT3);
    PCMSK0 &= ~(1<< PCINT4);
    PCMSK0 &= ~(1<< PCINT5);
```

Figure 23: Ultrasonic Trigger function part 1

Since ultrasonic sensors work as an PCINT interrupt it is necessary to disable all of the echo pins except one. This will make sure that other pins will not interfere and distort the desired sensor reading.

```
switch(which_sensor)
{
    case 1:
    {
        PCMSK0 |= (1<<PCINT0);
        break;
    }
    case 2:
    {
        PCMSK0 |= (1<<PCINT1);
        break;
    }
    case 3:
    {
        PCMSK0 |= (1<<PCINT2);
        break;
    }
    case 4:
    {
        PCMSK0 |= (1<<PCINT3);
        break;
    }
    case 5:
    {
        PCMSK0 |= (1<<PCINT4);
        break;
    }
    case 6:
    {
        PCMSK0 |= (1<<PCINT5);
        break;
    }
}
//actual trigger
PORTL |= (1<<PINL1);
_delay_us(15);
PORTL &= ~(1<<PINL1);

}
```

Figure 24: Ultrasonic trigger function part 2

8.2.2.2. Measuring and data handling

This function is a reading function. It measures 5 values, saves them in an array sorts them and then finds the median.

The reason for this process is that during ultrasonic tests the small error was discovered. Statistically one measurement out of five was a deviation. Therefore, the software compensation was made – finding the median value.

The function takes one argument- sensor number. It defines from which sensors the data should be measured. The in the for loop the sensor is triggered and the measured time value is saved in an array. This process is reapeated 5 times.

```
void sort_and_measure(char which_sensor)
{
    for (int i=0;i<5;i++)//making 5 measurement for one sensor
    {
        ultrasonic_trigger(which_sensor);
        switch (which_sensor)//saving value in the array
        {
            case 1:
            {
                table_of_measurements[i]=wave_time_1;
                printf("%d   ",wave_time_1);
                break;
            }
            case 2:
            {
                table_of_measurements[i]=wave_time_2;
                //printf("%d   ", wave_time_2);
                break;
            }
            case 3:
            {
                table_of_measurements[i]=wave_time_3;
                //printf("%d   ",wave_time_3);
                break;
            }
            case 4:
            {
                table_of_measurements[i]=wave_time_4;
                //printf("%d   ",wave_time_4);
                break;
            }
            case 5:
            {
                table_of_measurements[i]=wave_time_5;
                //printf("%d   ",wave_time_5);
                break;
            }
            case 6:
            {
                table_of_measurements[i]=wave_time_6;
                //printf("%d   ",wave_time_6);
                break;
            }
        }
    }
```

Figure 25: Sorting and data handling function for ultrasonic sensors

Then, the sorting is made by bubble sort method:

```
unsigned int temporary_value;
for (int j=1;j<5;j++)//sorting the measurements in the array so that we can
  get the median
{
    for (int i=0;i<5;i++)
    {
        if (table_of_measurements[i]<=table_of_measurements[j])
        {
            temporary_value=table_of_measurements[i];
            table_of_measurements[i]=table_of_measurements[j];
            table_of_measurements[j]=temporary_value;
        }
    }
}
table_of_medians[which_sensor-1]=table_of_measurements[2];//saving the
  medain for n-sensor
}
```

Figure 26: Sorting function

After that the median is found which is the third value in the sorted array.
Note that the numeration in C language starts form zero, therefore the value we save as medin is denoted by [2]. And because of that it is necessary to substract one form our sensor number – the numbering in this case begins from 1 and needs to be transfomred into C standards.

The real readigs and measuremtns are made in PCINT interrupt routine.When the trigger signal is made, this interrupt routine is executed. In this the sensor number is checked to ensure that the right sensor echo pin is read.

Then, if the pin went HIGH the timer begins. And when the echo pins go LOW the timer stops and the measured time is saved – this is the value the sound wave travelled both ways.

```
ISR (PCINT0_vect) // for ultrasonics sensors
{
    if (sensor_num==1)
    {
        if( (PINB & (1 << PINB0)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_1=TCNT1; //save time
            TCCR1B = 0; //stop counter
            TCNT1 = 0; //reset stored counter value
        }

    }
    if (sensor_num==2)
    {
        if( (PINB & (1 << PINB1)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_2=TCNT1; //save time
            TCCR1B = 0; //stop counter
            TCNT1 = 0; //reset stored counter value
        }
    }
    if (sensor_num==3)
    {
        if( (PINB & (1 << PINB2)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_3=TCNT1; //save time
            TCCR1B = 0; //stop counter
            TCNT1 = 0; //reset stored counter value
        }
    }
```

Figure 27: Ultrasonic interrupt routine part 1

```
    if (sensor_num==4)
    {
        if( (PINB & (1 << PINB3)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_4=TCNT1; //save time
            TCCR1B = 0; //stop counter
        }
    }
    if (sensor_num==5)
    {
        if( (PINB & (1 << PINB4)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_5=TCNT1; //save time
            TCCR1B = 0; //stop counter
            TCNT1 = 0; //reset stored counter value
        }
    }
    if (sensor_num==6)
    {
        if( (PINB & (1 << PINB5)))
        {
            // LOW to HIGH pin change
            TCCR1B |= 1<<CS12 ;  //when 1st change on the pin appears - start
              the counter //prescaler 256
        }
        else
        {
            // HIGH to LOW pin change
            wave_time_6=TCNT1; //save time
            TCCR1B = 0; //stop counter
            TCNT1 = 0; //reset stored counter value
        }
    }
}
```

Figure 28: Ultrasonic interrupt routine part 2

Alternatives:

As for the alternatives, it is always possible to handle the sensors directly in the main code as a function which would work on the same principle as interrupt. But since the main code should be as readible as possible, the functions and interrupts make it much easier.

Also the error in readings- it could be possible to handle directly in the main code as well. For example, it could be checked if at least 3 consecutive readings were the same with predetermined precison (which should be then calculated). But again it would add extra lines and make it less readibe and also it may give some false output if the enviroment is changing very fast. Also the timing would need to be considered.

8.2.3. Optical sensor

Optical sensor is the very important part of the design. The software control is based upon the principle the sensor works. For more detailed information pleaske look in **Appendix B**

8.2.3.1. *Interrupt routines*

The wheel position is measured by optical sensor and it is handled in the interrupt as follows:

```
ISR (INT2_vect) // count pos_wheel_2
{
    pos_wheel_2++;
    count_wheel_2++;
}
ISR (INT3_vect) // count pos_wheel_1
{
    pos_wheel_1++;
    count_wheel_1++;
}
```

Figure 29: Optical sensors interrupt routines

The interrupt is set in such manner that it is triggered by an rising edge (look for pin setup in **Appendix G**). For that reason, the area between the holes- the solid object makes an output signal with the rising edge – the pin goes HIGH. The code counts this signals whenever they occur.

Alternatives:

As for the alternatives in the code it could be done in tha main again – the wheels would rotate and the manual check of the optiacl sensor pins would be made with *if* and *while loop* statements. This is a very impractical way since there are other sensors – ultrasonic which also need to be checked. In the main there is only one thing that can be executed. This may result in unprecise readings and give some extra error in the position variables.

8.2.4. Motor control

Since BOB is an autonomous vehicle it should be able to move around on its own. This means that such elements as motors are required to make it move. Also the control of them should be considered. How fast should the robot move? In which direction? This problems are handled in the motor control part of the design. For better understanding the code, it is important to know how H-bridge and PWM signal work. For detailed explanation please look in **Appendix C.**

8.2.4.1. Speed control function:

Based on discussed H-bridge principle the function *set_motor_speed* was made. It has two input arguments- first one defines which motor is addressed and second is which "state" it should be in, this includes: clockwise rotation, counter clockwise rotation and stop command.

Each motor has 2 pins. By switching them on and off the desired state is achieved:

```
void set_motor_state(char which_motor, char state)
{
    switch (state)
    {
        case CW:
    {switch(which_motor)
        {
            case M2:
            PORTA |= (1<<PINA1);
            PORTA &=  ~(1<<PINA0);
            break;
            case M1:
            PORTA |= (1<<PINA2);
            PORTA &=  ~(1<<PINA3);
        }
        break;
    }

            case CCW:
            {
                switch(which_motor)
                {
                    case M2:
                    PORTA |= (1<<PINA0);
                    PORTA &=  ~(1<<PINA1);
                    break;
                    case M1:
                    PORTA |= (1<<PINA3);
                    PORTA &=  ~(1<<PINA2);
                }
                break;
            }
            case STOP:
            {
                switch(which_motor)
            {
                case M2:
                PORTA &= ~(1<<PINA0);
                PORTA &=  ~(1<<PINA1);
                break;
                case M1:
                PORTA &= ~(1<<PINA2);
                PORTA &= ~(1<<PINA3);
            }
            break;
            }
        }
    }
```

Figure 30: Motor control function

The motor speed is controlled via PWM board. Since it used i2c protocol, the functions from a premade library were used (with some small changes).

One function initializes the communication with PWM board at a specified frequency. Since it is used to control motors, the highest possible frequency was chosen – 200Hz:

```
PCA9685_init_pwm(PWM_FREQUENCY_200);
```

Figure 31: Function to initialize the PWM board frequency

Another function is to control the speed. It is done with giving an argument to the function of specified quantity - since it is the 12-bit board (max. value 4095) the 100% duty cycle corresponds to 4095. By giving the numeric value the duty cycle is controlled and thus the speed of the motor:

```
PCA9685_set_pwm(M1_PWM,0,motor_speed_1);
```

Figure 32: Function to control the speed of specified motor

As it can be seen the function has three arguments:

- Argument 1: The pin that should be changed – the pin was assigned with name so that is clearer to understand and use
- Argument 2: It is the initial ON value – in that case it is 0
- Argument 3: The OFF value – after this value the pin goes LOW

Alternatives:

It could be possible to generate the PWM signal from Arduino but since it is not independent and needs to be combined with other elements such as sensors the additional board was the best solution for this project.

Another possibility is to neglect the PWM signal and just turn motors ON and OFF. Then the speed control is neglected but the relatively straight movement is achieved. It would be very dependent on the optical sensors. If the count of one wheel is higher than another it would stop and turn ON if they are equal.

8.2.5. Buttons

Buttons are the tool to turn the device ON and OFF. Since the component was selected its function needed to be defined in the software.

Principle of working:

The working principle of a microswitch is simple. When button is pressed it sends the high signal at keeps it until it is unpressed.

8.2.5.1. Interrupts routines

The buttons are handled in the interrupts routines where stop button has the highest priority.

Let's look at the stop button first:

```
 ISR (INT4_vect) //stop button
{
    set_motor_state(M1,STOP);
    set_motor_state(M2,STOP);
    PCA9685_set_pwm(M2_PWM,0,0);
    PCA9685_set_pwm(M1_PWM,0,0);
    flag_stop=1;
}//stop button
```

Figure 33: Interrupt routine for stop button

Here when the stop button is pressed the program "jumps" and executes this interrupt routine. It can be seen that inside this routine the previously discussed functions are used (see motor control chapter). The first two stop the motor with the use of H-bridge and the next two set their speed to zero. The last element namely *flag_stop* is an indicator whether the button was pressed or not.

As for the start button the interrupt routine is:

```
ISR (INT5_vect)//start button
{
    flag_start=1;
}//start button
```

Figure 34: Interrupt routine for start button

As it can be seen there is only one line executed – the variable value change. If the start button was pressed it sets the flag (the variable value is changed to 1) and then it can be referred in the main code:

```
while(1)
{
    flag_start=0;
    if (flag_start==1)
    {
        flag_start=0;//clear the start flag variable
        flag_crazy=0;
        flag_stop=0;
        pos_wheel_1=0; //we clear holes counters for optical sensors
        pos_wheel_2=0;
        printf("Stops at flag_start loop" );
        break;//break the while loop
    }
```

Figure 35: Main code: Initialization loop

This is the very beginning of the main code. It enters this *while loop* and stays there until the button is pressed. If it is, the flag is set to 1 and then in the *if statement* all the necessary variables are cleared and the loop is broken (the *break* command)

Alternatives:

For the stop button, all the function could be executed in the main. The flag would be set and that would be checked in the main – if it was set then it would stop the motors.

The drawback of this is that it would be constantly checking the flag and if there is any another loop encountered then it would be necessary to check the flag value as well – it takes extra space and makes it harder to combine.

And for the stop button, it could start running motors as soon as the button is pressed. But since it is necessary to make sure that the does not execute next lines (which should be executed after the start button) some loop is required. And combining both, the start commands and the loop would save time in the interrupt and extra lines of code.

8.3. Test procedure

The software is a complicated part of the design. It can be split up into smaller parts like individual sensors or systems that should be implemented.

Testing of Software:

1. First the software design was divided into smaller, individual parts – according to sensors, components, task to do etc.
2. Then the code prototype was written for each subtask and the part was tested with that. According to its functionality the test would look differently. For example, if ultrasonic sensors were the subtask, the test would check if they are able to detect and obstacle and give reasonable data. (The functionality of each element It was discussed in particular section for this specific component). If the result was positive the next subtask could be handled.
3. If all the components were tested successfully then they could be combined together. The code can be altered in some manner but the main functions written in the testing should remain unchanged.
4. The whole system is tested. The short test code is written so that each element is tested and, if necessary, cooperating with other element's function – as information exchange etc.
5. If first test of combined system was positive then the real main code can be written according with the requirements set in the beginning.
6. Then the very final test is made. The device is put in the real-environment and should do the desired tasks. If it is positive then the device design is finished.

In case of any failed tests, the solution should be found and then tested again. If problem is very difficult to solve then it may need an assistance or in worst scenario the different method should be implemented.

8.4. Main Code

The main code is the combination of all the functions written for particular elements. It is the implementation of the block diagram discussed before.

```
int main(void)
{
    uart_init(); // open the communication to the microcontroller
    io_redirect(); // redirect input and output to the uart
    i2c_init();
    set_pins();
    flag_start=0;
    flag_stop=0; pos_wheel_1=0;pos_wheel_2=0;
    sensor_num=1;
    motor_speed_1=2250;
    motor_speed_2=3000;
    while(1)//the controller runs this forever
    {       //motor 1 is on the right
         //motor 2 is on the left from the front
        while(1)
        {
            flag_start=0;
            if (flag_start==1)
            {
                flag_start=0;//clear the start flag variable
                flag_crazy=0;
                flag_stop=0;
                pos_wheel_1=0; //we clear holes counters for optical sensors
                pos_wheel_2=0;
                printf("Stops at flag_start loop" );
                break;//break the while loop
            }

         }
         while(1)
         {

                 if(flag_crazy==1)
```

Figure 36: Main code part 1

```
            {
                flag_crazy=0;
                break;
            }
        if (flag_stop==1)
        {
            flag_stop=0;
            break;
        }


        //start moving "forward"
                        printf("start loop \n");
                        PCA9685_set_pwm(M1_PWM,0,motor_speed_1);
            set_motor_state(M1,CCW);
                        PCA9685_set_pwm(M2_PWM,0,motor_speed_2);
            set_motor_state(M2,CCW);
                        _delay_ms(5);
                        //checking for obstacle
                        for (int i=1;i<4;i++)
                        {
                            sort_and_measure(i);
                            printf("median: %d \n",table_of_medians
            [sensor_num-1]);
                            if((table_of_medians[i-1]<350) &&
            (table_of_medians[i-1]>30))
                            {
                                //stop BOB
                                set_motor_state(M1,STOP);
            PCA9685_set_pwm(M1_PWM,0,0);
                                set_motor_state(M2,STOP);
            PCA9685_set_pwm(M2_PWM,0,0);
                                _delay_ms(1000);

                                theta=degrees_to_radians(40);
                                //turn 40 degrees right
                                printf("turn right \n");
                                real_turn(right);
                                _delay_ms(1000);
                                obstacle_counter++;
                                break;
                            }
                        }
                        if (obstacle_counter==5)
                        {
                            for (int i=0;i<2;i++)
                            {
                                theta=degrees_to_radians(360);

                                     real_turn(right);
                                     obstacle_counter=0;
                                 }
                                 PCA9685_set_pwm(M1_PWM,0,0); set_motor_state
                     (M1,STOP);
                                 PCA9685_set_pwm(M2_PWM,0,0); set_motor_state
                     (M2,STOP);
                                 flag_crazy=1;
                                 }
                             }
    }
}
```

Figure 37: Main code part 2

9. User test

The user tests were made during TEK EXPO. There was specially designated area were BOB was put in. The simple carton boxes were put there as the obstacles. The user could enter the area and initialize the robot which did not make a problem. They were also allowed to change the place of the boxes. The received feedback was mostly positive with some small suggestions of how the device should be changed in order to be more user-friendly.

Feedback:

First of all, the robot design was appreciated. They claimed it was original and liked that it could fit and work in normal everyday life (like small rooms in a house etc.). They felt relatively safe though they were surprised by the acceleration and the pace of the robot. It was discovered that BOB was moving too fast and its speed needed to be decreased so that it would match to walking speed of the user. The user suggested to make the BOB one meter taller and one meter wider, so people could feel what is coming towards them (bartender bot) because now it is too small to reach for the drink easily.

10. Ethical aspects

It could be argued that it is not ethical to automate serving drinks to people as the rate of young alcoholics is already increasing.

However, this robot will not serve drinks to people outside of clubs and parties were people will be drinking regardless of how fast it is served, so it should not contribute to any problems with alcoholism.

It could be a problem that bartenders lose their jobs.

This would not be the case as this robot is designed to help with the huge amount of people around the bar and spread out people more in the club. So, the Bartenders are still needed for the people still at the bar but now there are not too many people standing in line at the bar. In addition, the bar bot needs to be refilled occasionally and a bartender could do that.

11. Technical review

One spectator suggested that a custom motor control board was to be manufactured to avoid overheating.

Making a custom motor control board would have allowed for a more efficient heatsink to be made, that way overheating would have been avoided

Another spectator suggested that navigation via GPS would work better than light sensors.

Navigating via GPS would allow for B.O.B to accurately find the customer, without the hassle of using light sensor for the wheels.

We also got a suggestion to us a stepper motor instead of a normal motor to get rid of the optical sensors.

A stepper motor would have allowed for the robot to know where it is with the motor itself, instead of using external light sensors.

Questions we got from different spectators. Why can it not move for more than a minute at a time? If it worked how it should, how would it navigate between obstacles? Another question was why it could not run straight?

12. Conclusion

It was the complicated process but in the end the robot was built and working as expected. Having requirements in mind BOB was designed and made in order to fulfill them. As it can be seen throughout the process the requirements were met.
The robot dimension and frame were designed so that it has stability but also looks modern and fits small rooms. The weight of the robot was designed so it can be lifted if necessary. BOB needs an operator to initialize the device, which makes it easy to use. Also, its movement speed is based on the average walking speed, so it can be easily supervised. It's advanced sensors and control solutions enable it to detect obstacles and avoid them.

Overall it can be said that the project was a success and fulfilled the requirements stated in the very beginning. For the list of all components used please look at **Appendix E**

In order to see the tasks distribution please look at **Appendix D** and for the time plan please look at **Appendix F**

Appendices

Appendix A

Principle of working:

The ultrasonic work as follows:

Figure 38: Ultrasonic sensor- principle of work
picture form: https://components101.com/ultrasonic-sensor-working-pinout-datasheet

First the trigger pin needs to be on for 10 microseconds.
Then the sound wave of 40HZ frequency is send
The echo pin is waiting for the reflected wave.

The echo pin stays high until it receives the signal back. By having this information, it is possible to calculate the distance it has measured by the following formula:

$$s = t * v$$

But since it is an echo wave – it has travelled the distance twice; the time has to be half of the initial value which yields:

$$s = \frac{1}{2} t * v$$

For simplification of calculation the speed of sound was considered as the constant of:

$$v = 343 \frac{m}{s}$$

So, some calculations were made to see if sensor is reliable:

	256 prescaler										
Nr	**5cm**	**10cm**	**15cm**	**20cm**	**25cm**	**30cm**	**35cm**	**40cm**	**45cm**	**50cm**	**55cm**
1.	21	38	66	85	107,5	129,5	151,5	173,5	195,5	217,5	239,5
2.	22	37	55	85	101,5	122,2	142,9	163,6	184,3	205	225,7
3.	25	37	55	86	101	121,1	141,2	161,3	181,4	201,5	221,6
4.	23	37	56	85	101,5	122	142,5	163	183,5	204	224,5
5.	20	37	56	85	103	124,4	145,8	167,2	188,6	210	231,4
Median:	22	37	56	85	101,5	122,2	142,9	163,6	184,3	205	225,7
distance [cm]:	6,037	10,153	15,366	23,324	27,852	33,532	39,212	44,892	50,572	56,252	61,932

Figure 39: Table of measurements for ultrasonic sensor

	256 prescaler									
Nr	**60cm**	**70cm**	**80cm**	**90cm**	**100cm**	**110cm**	**120cm**	**130cm**	**140cm**	**150cm**
1.	261,5	305,5	349,5	393,5	437,5	481,5	525,5	569,5	613,5	657,5
2.	246,4	287,8	329,2	370,6	412	453,4	494,8	536,2	577,6	619
3.	241,7	281,9	322,1	362,3	402,5	442,7	482,9	523,1	563,3	603,5
4.	245	286	327	368	409	450	491	532	573	614
5.	252,8	295,6	338,4	381,2	424	466,8	509,6	552,4	595,2	638
Median:	221	261	291	326	364	400	433	471	511	550
distance [cm]:	60,642	71,618	79,850	89,454	99,882	109,760	118,815	129,242	140,218	150,920

Figure 40: Table of measurements for ultrasonic sensors

Note: Though calculations were made before, during the process it was decided to change the prescaler and then the new time values needed to be measured.

As it can be seen in the tables, the sensors' output is reliable. Though it may be different for each sensor, the error is small enough to treat them as one model.

By knowing this information there were made two functions: *ultrasonic_trigger* and *sort_and_measure*

Appendix B

Principle of work:

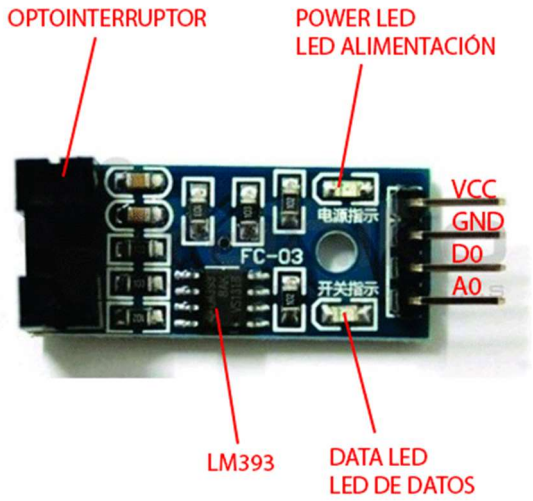


Figure 41: Optical sensor FC-03
picture from: http://androminarobot-english.blogspot.com/2017/03/encoder-and-arduinotutorial-about-ir.html

In optointerruptor there is a light emitting element and on the opossite side is the a light receiveing element. If there is no obstacle between those elements there is no signal. But if the light beam is stopped for example by the paper sheet then a HIGH signal will be sent – it will last for as long as the beam is ceased.

Therefore, it is possible to calulate the rotations of the wheel – by the special made ring with 90 holes it can be defined what is the position of the wheel.

Appendix C

Puls Width Modulation (PWM) - Principle of work :

Pulse width Modulation:

During the main algorithm development such issues were encounterd- how fast shoud the motors rotate and in which direction?

For that reasen in the hardware design the H-bridge and PWM board was selected. Former one enables to control the direction and ,thanks to the PWM board ,the speed control as well.

The speed is controlled by so called PWM signal where PWM stands for Pulse Width Modulation.
As the name suggests the width of the signal is controlled. Normally it is HIGH or LOW signal where former one is +12V and latter is 0V [considering motor supply power].

But thanks to PWM it is possible to contorl this width and achieve the different voltage. This is when term duty cycle comes in. Duty cycle is given in percentage and it describes the amount of time the signal is HIGH:

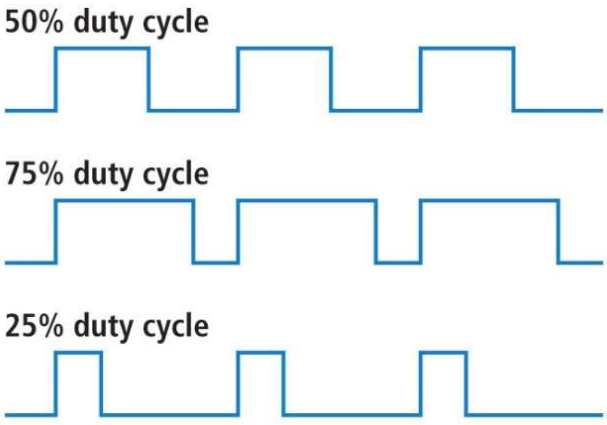


Figure 42: Duty cycle diagram
picture taken from: https://learn.sparkfun.com/tutorials/pulse-width-modulation/all

Then 100% dytu cycle is 12V. If half of it is taken so 6 V the duty cycle is also halfed: 50 %. And 25% dyty cycle coressponds to 0.25*12=3V

H-bridge: Rotation direction control:

With H-bridge it is possible to control the direction of the motor. This electrionc element consits of four transistors which act as switches. By switching them on and off the rotation direction can be changed:

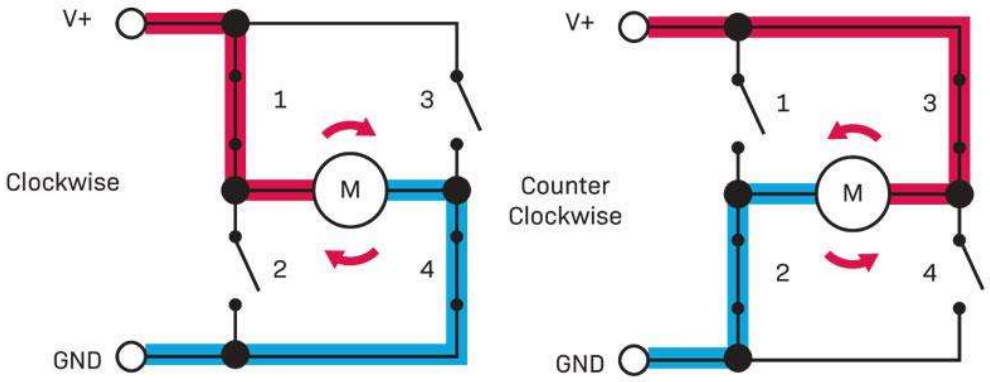


Figure 43: H-bridge – principle of working
Picture taken from: https://diyodemag.com/projects/bridging_the_gaps

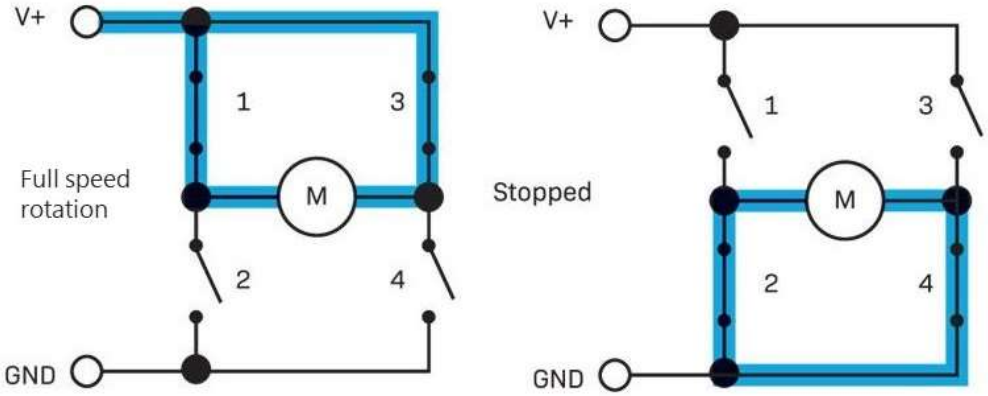


Figure 44: H-bridge principle of working
picture taken from: https://diyodemag.com/projects/bridging_the_gaps

Appendix D

The process was complicated and required the task distribution. Since they are cover in the report and written by the person who did them, then the following list will specify which team member contributed to what part of the design.

The list of the tasks done:

Wiktoria Aleksandra Kos:

- Main areas: Software development
  Tasks done:
  - Sensor functions- ultrasonic sensors, optical sensors
  - Motor control functions – speed control and direction of rotation
  - Logic of the robot
  - Pin connection design
  - Software testing and corrections
  - Component list management
- Report writing:
  - Paragraph *1. Introduction*
  - Paragraph *6.3 Buttons*
  - Paragraph *7.1 Embedded structure*
  - Paragraph *8.1 Block diagram*
  - Paragraphs: from *8.2.2 Ultrasonic sensors, 8.2.3 Optical sensors, 8.2.4 Motor Control, 8.2.5 Buttons*
  - Paragraphs *8.3 Test procedure* and *8.4 Main Code*
  - Paragraph *9. User Test*
  - Paragraph *12. Conclusion*
  - Appendices: *Appendix A,B,C, D , E* and *G*

Morten Milton Schou:

- Main areas: Electronic development
  Tasks done:
  - PCB design
  - Pin connection design
  - Circuit schematic/design
  - Motor speed function – finding the right ratio
  - Battery calculations
- Report writing:
  - Paragraph *3. Requirements*
  - Paragraph *6.1 Motor Control, 6.2 Sensors, 6.4 Microcontroller and 6.5 Battery selection*
  - Paragraph *7.2 PCB Design*
  - Appendices *Appendix I, J*

Nick Henderson:

- Main areas: Mechanical design
  Tasks done:
  - Frame design – calculation, manufacturing
  - Wheel mount design
  - Motor mount
  - Mobility calculations
  - Axles design
  - Sensor mount
- Report writing:

- Paragraph *5.1 Frame design*
- Paragraph *5.2 Wheel Assembly Design*
- Paragraph 3.3 Desirables

Niels Tscherning Andersen:

- Main areas: Mechanical design
  Tasks done:
  - Wheel mount design
  - Motor mount
  - Axils design
  - Timetable management
- Report writing:
  - Paragraph *2. Project Background*
  - Paragraph *5.3 Bottle Plate*
  - Paragraph *10. Ethical Aspects*
  - Paragraph *11. Technical Review*
  - *Appendix F, H*

Patrick Ma:

- Main areas: Mechanical and Software design
  Tasks done:
  - Concept design
  - Frame calculations
  - Axle design
  - Logic – mapping
  - Encoder wheel design (for optical sensor)
  - Valve
- Report writing:
  - Paragraph *3. Concept Design*
  - Paragraph *8.2.1. Mapping*

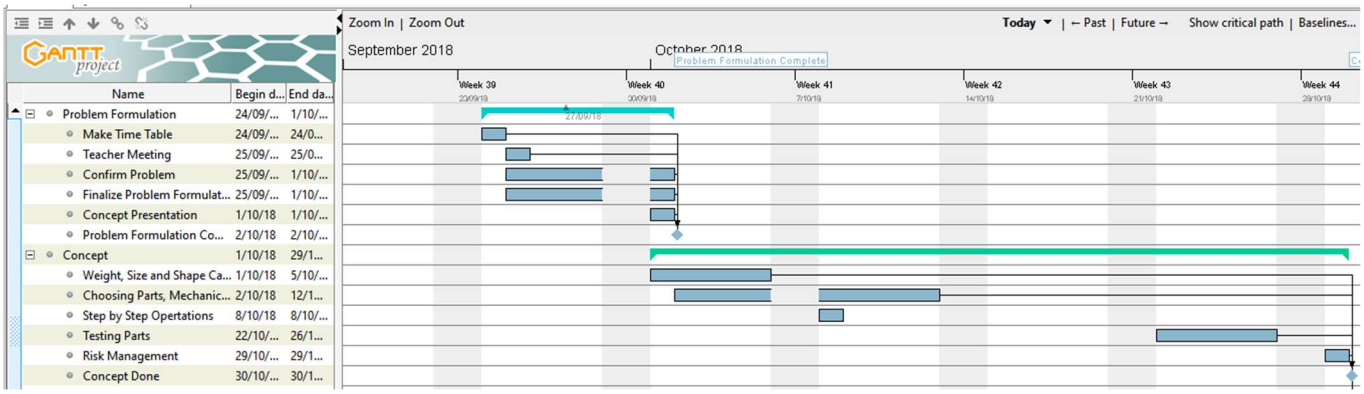
Appendix E

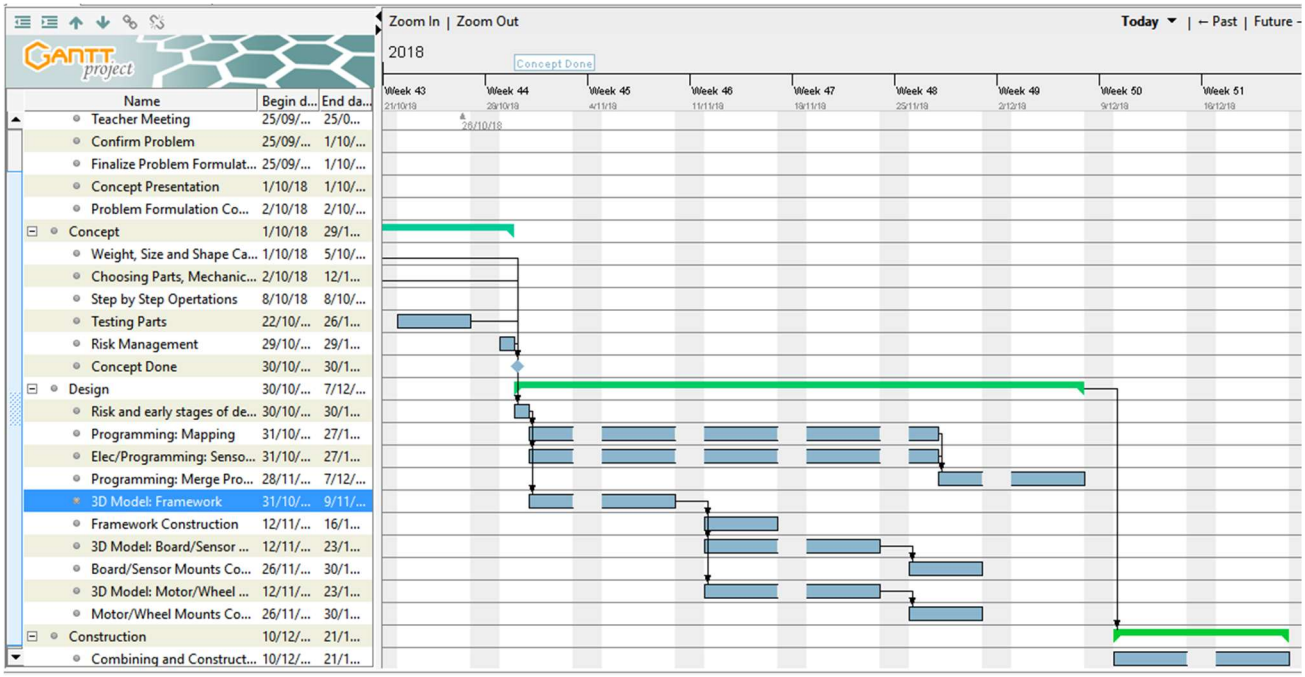
Since Bar Operating Bot is a complex and big project which required a lot of work and components in order to fulfill the requirements.
Here is the full list of components used (As no parts were outsourced, the overall cost was 0 DKK)

Date	Number of	Description	element cost	Total SDU Cost (DKK)	Remarks	Supervisor acceptance
08.10.2018	1	lidar sensor	0	0,00		Part received
01.10.2018	1	Ultrasonic distance sensor: HCSR05	0	0,00		Part received
10.10.2018	1	RS PRO 321-3164 DC	0	0,00	For testing	Part received
10.10.2018	1	AtMega 358 - Nano Module RS 696-1667	0	0,00		Part received
10.10.2018	0	SDU Nano Shield	0	0,00	Cancelled	Cancelled
05.11.2018	6	Ultrasonic distance sensor: HCSR05	0	0,00		Part received
05.11.2018	1	RS PRO 321-3164 DC	0	0,00		Part received
05.11.2018	2	LN 298 dual H-bridge	0	0,00		Part received
06.11.2018	1	Battery RS-PRO	0	0,00		Part received
07.11.2018	2	castor wheel	0	0,00		Part received
08.11.2018	2	normal wheels + mechanic model	0	0,00		Part received
12.11.2018	1	AtMega 358 - Nano Module RS 696-1667	0	0,00		Part received
12.11.2018	50	Brackets	0	0,00		Part received
13.11.2018	1	T-slots-frame	0	0,00		Part received
13.11.2018	2	Big wheels	0	0,00		Part received
12.11.2018	1	Arduino MEGA	0	0,00		Part received
19.11.2018	2	Optical sensor FC-03	0	0,00		Part received
04.12.2018	1	12-bit PWM board PCA9865	0	0,00		Part received
04.12.2018	2	3D print element for motor rotation control	0	0,00		Part received
12.12.2018	1	12-bit PWM board PCA9865	0	0,00		Part received
10.12.2018	2	Solenoid	0	0,00		Part received
10.12.2018	1	Servo	0	0,00		Part received

Appendix F

Time plan:





Appendix G

All of the used pins should be defined as either output or input. They are defined with reference to Arduino.

Output pin is when Arduino sends a data out and the Input is when the data is sent to Arduino (reads the data). Note that for the sensor the setup is the opposite – for example the ultrasonic sensor. Its echo pin sends an output which is the time the sound wave has travelled both ways. For the sensor it is an output pin but an input pin from the Arduino point of view – since it reads the data.

Therefore, the separate function named set_pins was made to set all the necessary pins:

```
void set_pins()//defines the pins-input/output ; sets the interrupts
{
    //motor PINS: A0 A1 A2 A3 (but all set high just in case)
        DDRA = 0x11111111; //output - motor input pins M2; //output - motor
          input pins M1

//  optical sensors
        DDRD &= ~(1<<PIND2); //input - wheel 1
        DDRD &= ~(1<<PIND3);// input - wheel 2
        EICRA |= (1<<ISC21) | (1<<ISC20); // set INT2 to trigger on rising edge
          - wheel 1
        EIMSK |= (1 << INT2); // Turns on interrupt for INT0
        EICRA |= (1<<ISC31) | (1<<ISC30); // set INT3 to trigger on rising edge
          - wheel 2
        EIMSK |= (1 << INT3); // Turns on interrupt for INT3
//  buttons
        //stop button
            DDRE &= ~(1<<PINE4); //STOP button input
            PORTE |= (1<<PINE4); //enable pull-up
            EICRB |= (1<<ISC40); // set INT4 to trigger on any edge
            EIMSK |= (1 << INT4); // Turns on interrupt for INT4
        //start button
        DDRE &= ~(1<<PINE5); //START button input
        PORTE |= (1<<PINE5); //enable pull-up
        EICRB |= (1<<ISC50); // set INT5 to trigger on any edge
        EIMSK |= (1 << INT5); // Turns on interrupt for INT0

//  ultrasonics
                DDRL = (1<<PINL1); //trigger pin L1
                DDRB &= ~(1<<PINB0); PORTB |= (1<<PINB0); //sensor 1 - in front
                  PCIN0
                DDRB &= ~(1<<PINB1); PORTB |= (1<<PINB1); //sensor 2 - front
                  left PCINT1
                DDRB &= ~(1<<PINB2); PORTB |= (1<<PINB2); //sensor 3 - front
                  right PCIN2
                DDRB &= ~(1<<PINB3); PORTB |= (1<<PINB3); // sensor 4 - side
                  left PCINT3
                DDRB &= ~(1<<PINB4); PORTB |= (1<<PINB4); //sensor 5 -side right
                   PCINT4
                DDRB &= ~(1<<PINB5); PORTB |= (1<<PINB5); //sensor 6 - back
                  PCINT5

                 PCICR |= (1 << PCIE0); // set PCIE0 to enable the group for
                   PCINT8...14
                 PCMSK0 |= (1<< PCINT0) | (1<<PCINT1) | (1<<PCINT2) | (1<<PCINT3)
                    | (1<<PCINT4) | (1<<PCINT5);

         sei(); // turn on interrupts for all the sensors!!!
          PCA9685_init_pwm(PWM_FREQUENCY_200);//we need to say at what frequency
            the PWM board should run at

 }
```

Appendix H

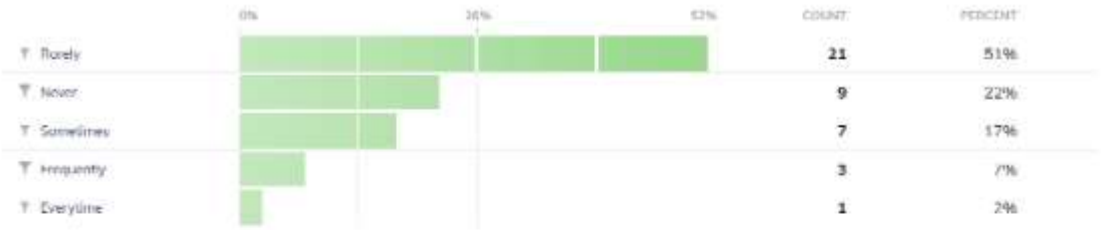
Public Survey:

Question 04

How often do you spill your drink when moving from the bar?

Answers 41 100%

Skips 0 0%



	COUNT	PERCENT
Rarely	21	51%
Never	9	22%
Sometimes	7	17%
Frequently	3	7%
Everytime	1	2%

Detailed Report

Question 05

Do you often feel the bartender gives you the wrong ratio of alcohol and mixer?

Answers 41 100%

Skips 0 0%

	COUNT	PERCENT
No	24	59%
Yes	17	41%

Detailed Report

Question 06

How often do you get the wrong order?

Answers 41 100%

Skips 0 0%

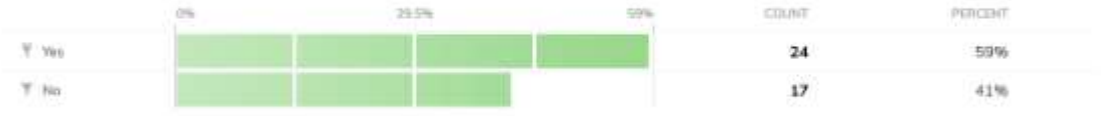
	COUNT	PERCENT
Rarely	20	49%
Never	19	46%
Sometimes	1	2%
Everytime	1	2%
Frequently	0	0%

Question 07

Is human contact important when ordering a drink?

Answers 41 100%

Skips 0 0%



	COUNT	PERCENT
Yes	24	59%
No	17	41%

Detailed Report

Appendix I

Bibliography

Adafruit. (n.d.). *Proximity and ambient light sensing (ALS) module.* Retrieved from https://cdn-learn.adafruit.com/assets/assets/000/037/608/original/VL6180X_datasheet.pdf

ELEC Freaks. (n.d.). *Ultrasonic Ranging Module HC-SR04.* Retrieved from https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf

Instructables. (n.d.). Retrieved from https://www.instructables.com/id/L298N-Motor-Driver-Controller-Board/

Learnabout electronics. (n.d.). *Slotted optical switch.* Retrieved from http://www.learnabout-electronics.org/Semiconductors/opto_54.php

Sparkfun. (n.d.). *H-bridge.* Retrieved from https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

Sparkfun. (n.d.). *Signal PWM.* Retrieved from https://learn.sparkfun.com/tutorials/pulse-width-modulation/all

TT electronics. (n.d.). *Slotted optical switch.* Retrieved from http://www.farnell.com/datasheets/9111.pdf

Wikipedia. (n.d.). *Printed circuit board.* Retrieved from https://en.wikipedia.org/wiki/Printed_circuit_board

Appendix J

Risk management:

Risk management

Group 9

October 30, 2018

1 Introduction

Risk management is a crucial part of a project for which intentions is to be permitted to sell their product to costumers.
Risk management is an risk estimation of 3 factors:

- Severity
- Probability of occurrence (Pocc)
- Probability of detectability (Pdet)

The scenario of which the risk will be estimated, is if the robot should get a motor-control board failure and therefore hit a person head-on.
If the motor-control board fail it is assumed the robot will begin moving at full speed. To estimate the severity of such case, 7 factors should be considered:

- Mass of robot
- Acceleration of the robot
- Height of robot
- Design of robot
- weight of person
- Height of person

2 Severity

As the robots mass and acceleration is known, the force of which the robot will hit the person will be:

$$\begin{aligned} Mass &= 8[Kg] \\ Acceleration &= 1.38\left[\frac{m}{s^2}\right] \\ Force &= 0.5 * 1.38 = 11.04[N] \end{aligned}$$

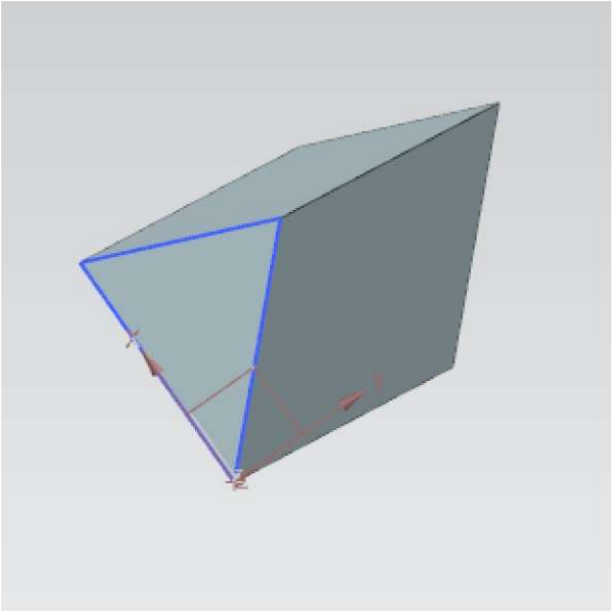


Figure 1: Model of the impact triangle

Even though the force is known, the design and the height of the robot still matters.
The design will be of a octagon, which will result in tiny edges around the robot. It is assumed that the person will solely be hit by one of these edges, which will result in maximum damage.
The design (without these edges) can be seen in appendix 1 (figure 3).
This edge can be seen as a triangle (see figure 1).
If a triangle with a force of 11.04 [N] is colliding with a person, height of the triangle and the person also have an importance, how is it gonna collide with him? Only his legs or his hole body?
The average height of a man is 1.76 meter and with a leg length of 0.8 meters. The robots height is 0.5 meters, which means the robot only will hit the legs.
Now it also have to be considered if the person is gonna fall, if the robot hits them. The maximum weight of the robot is 8 Kg while the average weight for a male of 1.78 meters is 76 Kg. From this it can be assumed the person is not going to fall over.
can be concluded that the damage will solely be distribution upon the persons legs.
Now the severity can now be compared to the official table (See figure 2).
The robot is laying between rank 2 and 3.

3 Probability of occurrence

The probability of occurrence is the likelihood of the motor-control board fail. To know the exact possibility of which the motor-control board will fail, can only be approximated from multiple tests for the solely purpose of finding flaws in the circuit. This needs industrial expertise and facility, which is not possible for this project and therefore a roughly approximation have to be made.

Ranking	Description SEVERITY
1	Failure effect is of no consequence to the patient
2	Failure effect may cause mild physician interaction but has only mind and temporare consequence to the patient
3	Slight discomfort to patient
4	Mild/Moderate safety consequence.
5	Patient severe health degradation / death

Figure 2: Official table of severity

Ranking	Description LIKELIHOOD	Likelihood of occurrence (Incidents/year out of 1000 cases)
1	Remote	Less than or equal to 1 in 1000
2	Low	>1 and < 50 in 1000
3	Moderate	>=50 and < 200 in 1000
4	High	>=200 and < 500 in 1000
5	Very High	>=500 in 1000

Figure 3: Official table of probability of occurrence

To make a roughly approximation acceptable to use, it always have to include a safety factor. It have been chosen that the probability of failure is $\frac{1}{100}$.
In the official table of likelihood of failure (see picture 3), it can seen the probability of occurrence for the project ranks as 1.

4 Probability of detectability

For the probability of detectability it is a little more difficult. To detect a failure in the motor-control board it must be connected to a Oscillator or similar, which is impossible in this project after the product is "released". Therefore the only way to figure out the detectability is from the same method as in the probability of occurrence, with multiple in external facilities.

5 Countermeasures

Even though the detectability is impossible to be measured in the project and the occurrence is only very roughly estimated, certain countermeasures can be made. If the severity is upgraded from rank 2/3 to 1/2, the detectability will have a less importance.
The harm done to the person is solely from the edges of the robot. If these edges were to be made more round, the severity would be upgraded to a ranking of 1/2.

Figure 4: Table of occurrence and severity ranking. Blue marker is before counter measures and yellow is after.

6 Conclusion

It can be concluded that the robot have a severity of rank 2/3, a probability of occurrence of $\frac{1}{100}$ which make it rank as 1 and a unknown probability of detectability.
In figure 4, the before and after countermeasures rankings can be seen.

Appendix 1:

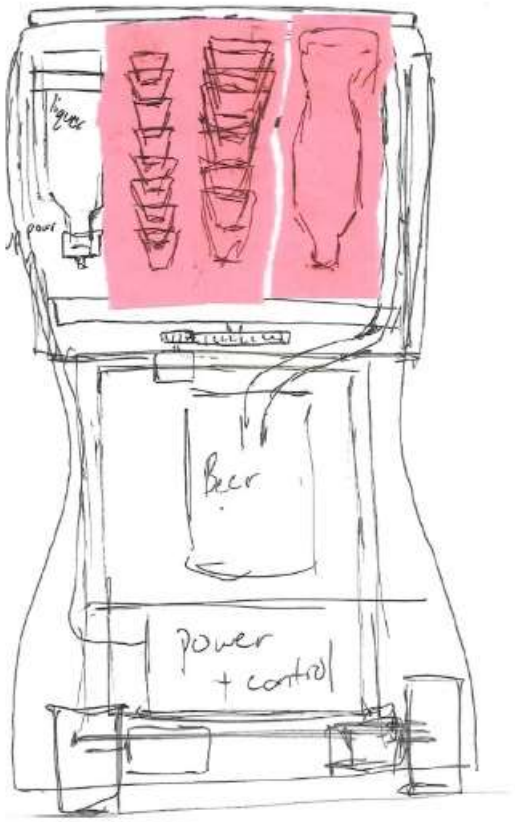


Figure 5: Hand sketch of desired design in front view