Final Report 2nd Year Project

Authors:

Francien Barkhof
Sander van den Bent
Oskar Bosgraaf
Sjoerd Gunneweg

Supervisors:

Arnoud Visser
Mitchell Verhaar

June 2022

Contents

1 Introduction

In this project our team has participated in the Hackathon - "Battle of Institutions" of the European Robotics Forum (ERF) in the Ahoy in Rotterdam. Together with 8 teams of universities in Europe, a robotics challenge was handed out to see which team was able to apply the best and most efficient solution to this challenge. The challenge contained a basic robotics operation, split up in multiple features and abilities that the robot had to fulfill, each feature being worth a certain amount of points. The companies associated with the ERF who were participating in handing out challenges were Lely, Franka and Robo House. These companies handed out their own unique challenge to the participants of the Hackathon. Every team worked on a challenge from one of these companies. Arnoud Visser, our initial client and supervisor, had given these specific assignments that were suited for sufficient preparation for the challenge. This knowledge was useful because it was known that ROS eventually had to be used to control the robot that had to perform the challenge. Also, to get familiar with working with a robot and ROS, a Mover5 robot was obtained from Braincreators. The Mover5 was only for practicing purposes however; for the ERF challenge itself a robot would be supplied that was suited for the corresponding challenge.

1.1 Client description

Our client was Lely Industries N.V. (Lely), the company that was selected to do the challenge for. Lely is a machine manufacturer for agricultural purposes, helping farmers making their work more easy and their life more pleasant using the Lely Juno robot which can be seen in Figure 1. The Lely Juno robot is an automatic feeding robot specifically designed to push back feed to cows without disturbing them. Initially the Juno 150 was designed, after which an advanced and smaller version, the Juno 100, was introduced. They have various built-in sensors which lets them drive various routes around barns.

Throughout the challenge multiple Q&A meetups were had with Lely where in-depth questions could be asked about the features and how they could be implemented. There were two of these meetings; one with all the teams of the universities gathered and one personally for the UvA team. Our robotics supervisor was Arnoud Visser during the entire project.

1.2 Problem description

The challenge consisted of a robotics operation that essentially had to fulfill 9 tasks, although they were all optional for a set amount of points and it was not mandatory that the robot had to satisfy all tasks.

The central problem of our challenge consisted of cows being fed hay inside a barn, and while they were eating the hay would be partially pushed away. The

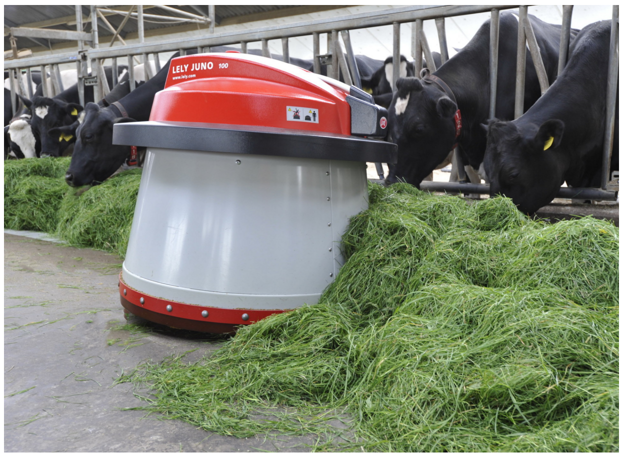


Figure 1: The Lely Juno robot

task of the robot was to push back this hay, and take into accounts other factors that could be happening inside the barn. 2 robots had to navigate through the barn, both pushing back hay. This is a multi-agent system problem, where the robots have to communicate with each other correctly.

1.3 Challenge description

The environment in which the robot had to navigate was a barn in which cows stand behind fences and were being fed hay. 2 Junos would navigate around, and respond to each other's presence. An important note is that inside the actual environment in which the robot was tested, there was not actually any hay to be pushed back; only the other features that the robot had to take into account were being tested. The environment resembled something like the image below.

The challenge consisted mainly out of 9 goals, each being worth a certain amount of points. These goals were certain functions that the robot had to be able to do. The goals consisted of:

- When the Juno passes inside a narrow area that is too small for 2 Junos to pass simultaneously, it must inform the farmer that it is inside this passage.
- When it encounters a plastic curtain strip, it must pass through it. The plastic curtain is placed on a random spot inside the barn.
- Be able to deal with holes and missing parts in the fence and keep on following it. This one contains two subgoals: a farmer should be able to cross the barn without the Juno colliding into him, and the Juno should communicate with him that he can cross the barn.

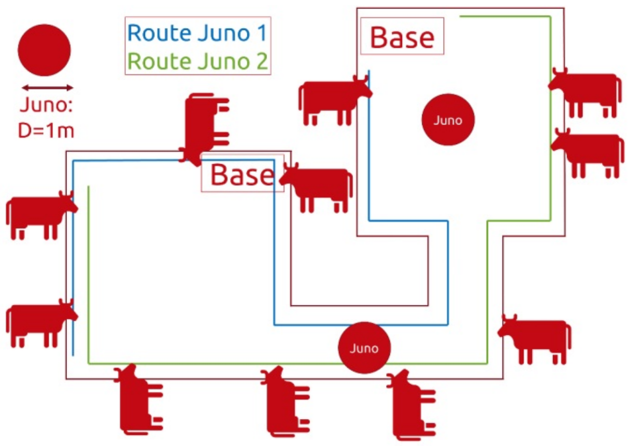


Figure 2: Example of barn environment

- Leave from and return to a home 'base' (a traffic cone) in the middle of the barn.
- Leave from and return to a home base at a random location inside the barn.
- Being able to drive around obstacles lying around in the barn; for example a wheelbarrow.
- Counting the amount of cows that are present with heat detection; the approximate temperature will be between 75 and 100 degrees Celsius.
- Turn on lights if there is another Juno in the neighborhood.

It was not possible to implement all these challenges, so a list of features had to be made that was ordered and ranked by importance.

1.4 Product vision

It was not realistic to implement all the 9 features so a selection was made of features of which were thought to be important and they were ordered by priority.
Because the robot had to navigate through a barn and do specific tasks, there were three very important elements to these operations: navigation, communication and extra features. For this reason, our product vision is mainly focused on subsections of these elements and they were expanded by specifying concrete features.

The navigation of the robot was a fundamental part because the Juno needed to move around the barn to (hypothetically) push back hay to the cows.

For communication, it was necessary to create a sufficient messaging between the Juno and the farmer, but also communication between the 2 Junos.

1.5 Previous research

Navigational robotics are not a new field in terms of research and thus a sufficient amount of knowledge about autonomous and guided navigation is available in papers like [6]. This paper gave a global overview of different navigational techniques and their strengths and weaknesses, which is helpful in the initial stages of robotic design. Promising for our product vision were autonomous methodologies with the use of vision [5] and sensor networks [2]. There is no public information available about the navigational driver code and methods used, for the existing Juno platform developed by Lely because this is a private and closed source project.
Although Lely has already implementated such a navigation robot, there is room for improvement in the area of multi-agent systems and the integration of image processing and computer vision.

2 Implementation

2.1 Proposed solution

In order to fulfill the wishes of our client a threefold solution was proposed and implemented. First of all, the robots needed to navigate through the barn while keeping the correct distance to the wall. This was done by using both a behaviour based and model based approach for each of the Junos so that only a few sensors were utilized. Secondly, the robots needed to interact with each other, communicate with the farmer and send log info to the developers. The interaction between the Junos was established by the ROS features 'client' and 'master'. An app was built to directly send information to the farmer. The ROSboard was used in order to update the developers on the current status and sensory input. Lastly, the extra features such as homing, distance detection and cow count were implemented with computer vision for which we needed the camera input.

2.2 Hardware

RoboHouse supplied every contestant of the Hackathon with hardware to test their implementation on. This hardware included a Mirte robot with integrated Orange Pi single board computer (OPI), a Raspberry Pi (RPI), an Arduino Uno R3 and multiple separate sensors. Because the Mirte robot is designed to be used in a classroom setting, equipped with an array of simple sensors for learning about the basics of robotics, multiple types of sensors could not be connected and used in combination with the Mirte's OPI.
This resulted in the choice being made to test the solution on two Robotis

Right distance	**Front distance**	**Action**
+R	+F	Adjust right
+R	-F	Turn left
R	+F	Move ahead
R	-F	Turn left
- R	+F	Adjust left
- R	-F	Turn left

Table 1: State-action scheme to follow the right wall

TurtleBot 3's, which make use of a RPI to connect all sensors. With the TurtleBot3's proprietary software and it being a well documented platform, the testing with especially the centrally mounted LiDAR scanner was almost effortless. See the Appendix for an in-depth explanation of the sensor mounting, connection and issues.

2.3 Navigation

2.3.1 Behaviour based

Behaviour based robots are able to switch behaviours according to its current state of the environment measured by sensory input. The chosen behaviour based approach to navigate one robot is the implementation of a wall following algorithm based on sonar distance inputs. A scheme of the different states of the environment, such as being in a corner or following a wall, has been linked to different behaviours which was then translated to code and adjusted by testing. To make the robot as cheap and sustainable as possible, only three sonar distance sensors were attached to the robot. This resulted in only having three measuring points; to the left, to the front and to the right. When following the right wall it is important that the robot is in the correct distance range from the wall while making sure there is nothing in front of it, creating the scheme in Table 1 in which the distances in Figure 3 are used.

A behaviour based approach has the advantage of being simple and therefore easy to implement, understand and adjust. However, because it chooses its behaviour based upon very basic requirements, it does not acquire any actual knowledge about the environment. Therefore, it is not able to deal with fluctuation in the environment such as a missing part of the wall.

2.3.2 Model based

Model based robots do acquire knowledge about the environment which they try to capture in a model. Based upon this created model they are able to behave and interact with the environment. The chosen behaviour based approach to navigate the other robot is the creation of a cognitive map with a LiDAR scanner [7] in which it follows the corners of the map.

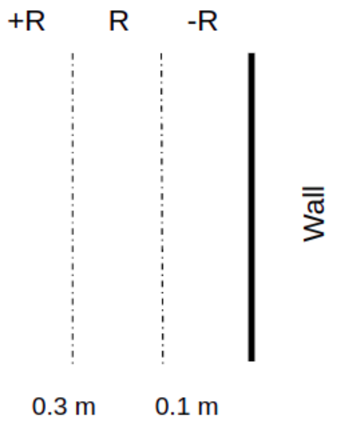


Figure 3: Visual definition of the terms used in Table one. R corresponds to being in the perfect distance to the wall, -R is close to the wall and +R to far.

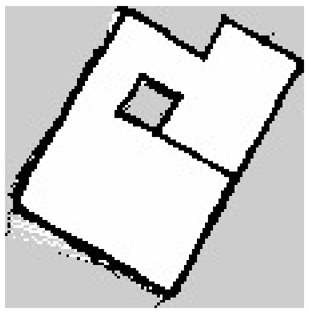


Figure 4: Map of the testing environment created by the SLAM algorithm

To create a map of the environment a LiDAR scanner has been used. LiDAR stands for Light Detection and Ranging. It is a technique with which the environment can be scanned to calculate distances to the walls and objects. With the use of the LiDAR scanner input, the SLAM (Simultaneous Localization and Mapping) algorithm [1] has been implemented. In this algorithm, localization and mapping is done simultaneously, which means that it is able to map the area whilst keeping track of the location of the robot within that area. In Figure 4 an example of such a SLAM map of our test environment can be seen.

When the map and localization is accurate, a path within this map needs to be planned. For the path planning, two solutions were proposed.

First of all, multiple points with the coordinates of the corners were manu-

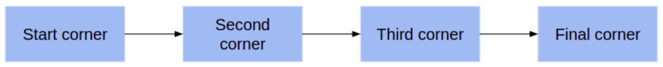


Figure 5: Visualisation of a hypothetical FSM for path planning

ally selected on the map. With these points a finite state machine (FSM) [4] was created of which each of the states represents a corner, as seen in Figure 5. By moving from one state to the other in the FSM a different corner was set as goal position for the robot. In this way it follows the path along the corners while keeping the correct distance.

Secondly, multiple points with the coordinates of the corners were found by the implementation of the Harris corner detection algorithm [3]. This algorithm takes an image of the map created by slam as input. In this image the algorithm determines the junction of two edges, where an edge is a sudden change in image brightness, and returns the corner's coordinates on the given image. The coordinates of the corners were then also used to automatically create the FSM. From this point on wards the path planning is the same as the first proposed solution.

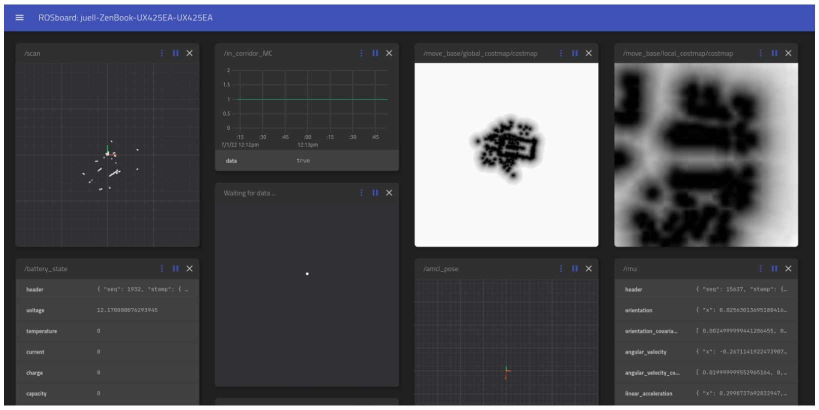
The advantage of the model based approach in comparison to the behaviour based approach is the fact that the robot is able to deal with a fluctuating environment as described above while following it's route more accurately. In order to make the robot as autonomous and automatic as possible the second solution of the model based is preferred to first proposed solution. However, in the tests the first semi-manual solution performed much better. Therefore, both solutions were proposed but only the first model based solution has been actually implemented.

2.4 Communication

In order to ensure no collision and understandability of the robots, two types of communication needed to be implemented. First of all, the two Junos cannot simultaneously be in a narrow passage. Therefore, they need to interact with each other about moving through a narrow passage so that only one Juno at the time will pass the corridor. Secondly, the robots need to communicate with the farmer to inform him/her about their statuses such as 'following the wall' or 'being in a narrow passage'. Lastly, the developers need to be updated on the current status and sensory input

2.4.1 Junos

The communication between the Junos was implemented by the usage of the ROS features 'client' and 'master' [8]. This entails that one Juno is the master



```
data                       true
```

```
header                     { "seq": 1932, "stamp": { …
voltage                    12.170000076293945
temperature                0
current                    0
charge                     0
capacity                   0
```

```
header                     { "seq": 15637, "stamp": {…
orientation                { "x": 0.02563013695180416…
orientation_covaria…       [ 0.0024999999441206455, 0…
angular_velocity           { "x": -0.2671141922473907…
angular_velocity_co…       [ 0.019999999552965164, 0,…
linear_acceleration        { "x": 0.2998737692832947,…
```

Figure 6: ROSboard example

and the other the client. The client Juno is only allowed to pass the corridor when the master client has already passed. This implies that when the client Juno arrives in the corridor while the master client has not passed yet, it has to move backward to make space and wait until the master Juno has passed. To obtain this specific behaviour they need to inform each other about their in or out corridor status. This was done by sending boolean (in/out) messages from client to master and the other way around.

2.4.2 Developers

The developers needed to understand the current robotic behaviour, to interpret the implementation and to make debugging or improvement possible. Therefore, insight into our implementation, robot statuses and sensory input needed to be shown and visualised in a compact and comprehensible manner. In order to do so the ROSboard was used. ROSboard is a node that runs a webserver on the robot. This allows the developers to receive information of the robot's status and sensory input in the form of textual information and visualisations. A ROSboard example can be seen in Figure 6.

2.4.3 Farmer

The farmer needed to be informed and updated about the current states of the robots and in particularly about being in a narrow passage as described in one of the challenges. This was supposed to be done by the usage of an mobile application. The mobile app receives information from the Azure services which receives information from the robot. Unfortunately, due to time limits, it was not possible to implement the communication channel between the farmer.

2.5 Features

2.5.1 Homing

One of the requirements was to make the Juno autonomously move back to its home base, a processed which is called homing. The challenge of this requirement was the fact that the base could be anywhere in the barn, even outside its route and could also be relocated during the challenge. Therefore, a robust and relocation invariant solution was desired. In order to achieve this an external webcam was used and a 'blob detection' algorithm was implemented. This computer vision algorithm was specifically designed to detect orange blobs since the home base was covered by an orange cloth. The algorithm captured orange parts on the screen and returned the bounding box and coordinates of the center orange object. The center of the orange object that was detected was used to determine if the robot had to navigate to the left or to the right. If the center of the blob was on the left side of the camera, left adjustment was made and vice versa. This way the robot was able to safely go home back to its base.

2.5.2 Juno vicinity lights

Because one of the Juno robots had an orange exterior, the blob detection and its size could also be used to estimate the distance from the camera to the recognised blob, and thus could be utilised as a distance metric between the two Juno robots. To receive extra points in the jury's grading, the Internet of Things based *Shelly Plug-S* needed to be turned on whenever the two Junos were within a certain distance of each other and off again whenever object/Juno avoidance had been accomplished. Using the manufacturers own product API and the *paho-mqtt* API, messages were published from the developers laptop, to a mqtt broker on a specific topic. A Shelly Plug-S is subscribed to this topic and therefore listens to the messages, relayed by the broker. The contents of these messages are HTTP commands for changing values in the plugs power settings. The use of this API was recommended by the Hackathon organisers in the preparation stage, due to the planned setup for the Shelly plugs including the necessity for a ways of authorisation. The final setup of these devices did not include any required means of authorisation whatsoever and a simple GET request function was written to change the power setting on the two plugs.

2.6 Developed product

In the developed product at the ERF the sensor mounting differed from the proposed solution as well as the environment and the technical interface of the Juno robot. This resulted in changes in our navigation approach and proposed solution, which will be discussed in this paragraph.

2.6.1 Junos

The first Juno robot was open, i.e. it did not have the red cap on it. Therefore, it was possible to mount the LiDAR scanner to it on the front using a 3D designed attachment. The second Juno was closed making it very hard to attach any Sonar sensor to it. Therefore, the camera was mounted on the second Juno.

2.6.2 Navigation

Due to the difference in sensor mounting with respect to the proposed solution, the navigation approach was changed as well. For the first Juno, mounted with the LiDAR scanner, the behaviour based approach was used instead of the model based approach that was proposed. The main reason for this was the fact that it was not possible to receive a 360 degrees range from the LiDAR scanner which resulted in a map that was not accurate enough for path planning. Therefore, the behaviour based approach was used to navigate the first Juno through the barn. For the second Juno, mounted with the camera, a blob following algorithm, as described in section 2.5.1, was used to navigate the robot instead of the behaviour based approach. The main reason for this was the fact that the behaviour based approach required distance sensors such as a Sonar sensor which was not attached to this Juno.

2.6.3 Homing

Due to the difference of the environment the blob detection algorithm to make the Juno move back to its home also differed from the proposed solution. A lot of red objects were positioned within the barn environment created by Lely. To prevent our blob detection algorithm from detecting red objects as the traffic cones (the base), the colour green was used in the blob detection algorithm of the developed product.

2.6.4 Vicinity lights

For the Junos to detect if there is another Juno around, blob detection was used as well. The Juno with the LiDAR was given a blue colour, so whenever it would approach the other Juno it would appear as a blue blob within its camera, and after a certain amount of time being in its vision the light would go on.

3 Conclusion

In conclusion, there were a few methods that worked quite well, both at the UvA and at the ERF, and some that didn't. There were a couple of adjustments that were made at the ERF, because some things that worked well on the Turtlebots did not work on the Junos.

Because of the structure of the Junos, the LiDAR could not be placed in the middle with a 360 degree range like it could with the Turtlebots. Instead, it was mounted on the front of the Juno. As a consequence of this, the scanning of the environment went worse, and in some situations the Juno performed a turn too early. In the future, this could be prevented by giving it a delay so that it waits a bit longer before taking the turn or transforming the LiDAR input as if it was mounted in the center of the robot.

Furthermore, the SONAR distance sensors could not be used, due to an inefficient way of mounting. This resulted in inaccurate measurements. In the future, it would be sensible to find a good way to mount them correctly, as the SONARS distance sensors can be a very helpful part of the robot.

With regard to navigation, a model based approach seemed better in theory since it allows for more diverse applications. When the model is accurate this approach has the potential to be a great fitting solution for our problem, as it can more accurately handle unique cases in our environment which other approaches will tend to struggle with. Unfortunately, the model was not accurate enough and could not be used in practice because of the offset of the scanner with regards to the centre of the robot. In the future, this approach could work if a LiDAR range of 360 degrees could be realised, if more time could be spend on fine tuning the map and localization and if the LiDAR could be transformed.

To improve upon the communication between the Junos, two parts of the proposed to solution should be revised and their problems should be taken into account. First of all, by implementing the multi-agent system with a master and client-system , the Junos share the same ROS-process and they also share the same "cmd vel" topic which sends the movement commands to the robot. This means that the two robots perform the exact same movements. To resolve this, names paces can be created for the topics for which a separation between the two Junos is required. Secondly, the communication could be made smarter in the future. In the current implementation, the client Juno always has to wait for the master Juno to pass the corridor before the client Juno is allowed to pass. However, this could be very time consuming whenever the client Juno arrives at the corridor first. In future work, this could be resolved by dynamically assigning the roles of master and client, based on the corridor status of either robot.

In future work, a mobile application should be implemented in order to update to Farmer about the robot's states. This way it is easier for the farmer to cooperate with the robot while understanding what it is doing and be informed whenever a human operation is needed. The app should be implemented as it is our client's goal to make the life of the farmer more pleasant. For the mobile application the usage of the Azure services is proposed.

With regard to the homing aspect of the final product, this implementation has a couple of distinct disadvantages when trying to use this in a real barn setting. The implemented way of homing back to the start/stop base relies on the distinct colour of the object and the recognised blobs, done with camera vision. Since a real barn environment would definitely not be clean, this dependence on colour would hinder the robustness of this homing solution. Another, more robust method for implementing the back to base feature would be to integrate the base locations in a cognitive map, to be able to simply provide this location as the final waypoint in the Juno's route. Since a model-based approached was used for the final product, this method was not looked at, but could be valuable for future work.

References

[1] Josep Aulinas et al. “The SLAM problem: a survey”. In: *Artificial Intelligence Research and Development* (2008), pp. 363–371.

[2] Maxim A Batalin, Gaurav S Sukhatme, and Myron Hattig. “Mobile robot navigation using a sensor network”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 1. IEEE. 2004, pp. 636–641.

[3] Li Yi-bo and Li Jun-Jun. “Harris corner detection algorithm based on improved contourlet transform”. In: *Procedia Engineering* 15 (2011), pp. 2239–2243.

[4] Li Caihong et al. “A complete coverage path planning algorithm for mobile robot based on FSM and rolling window approach in unknown environment”. In: *2015 34th Chinese Control Conference (CCC)*. IEEE. 2015, pp. 5881–5885.

[5] Guilherme N DeSouza and Avinash C Kak. “Vision for mobile robot navigation: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 24.2 (2002), pp. 237–267.

[6] Faiza Gul, Wan Rahiman, and Syed Sahal Nazli Alhady. “A comprehensive study for robot navigation techniques”. In: *Cogent Engineering* 6.1 (2019), p. 1632046.

[7] Tony Huang. *RPLIDAR-A3 Laser Range Scanner*$_r$*obotlaserrangescanner*. URL: https://www.slamtec.com/en/Lidar/A3.

[8] *Master and client communication documentation.* URL: http://wiki.ros.org/ROS/Tutorials/MultipleMachines.

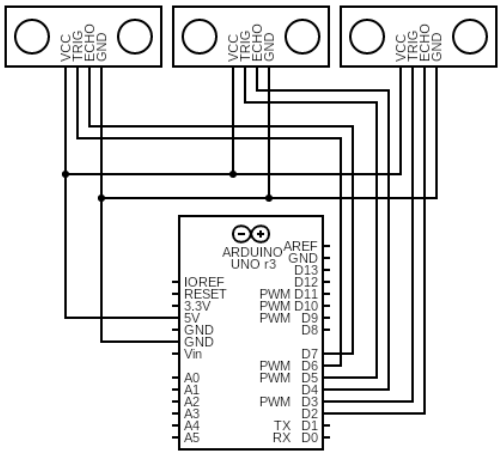


Figure 7: Circuit diagram of three HC-SR04 SONAR sensors connected to an Arduino UNO 3 circuit board.

4 Appendix

4.1 Sensor Use and Reading

The RoboHouse Mirte robot is intended for use of two SONAR based Time-of-Flight (ToF) sensors (HC-SR04) and two IR-scanners (Amg883). Because the whole Mirte PCB is designed for these types and amounts of sensors, the required three ToF sensors could not be connected to the OPI directly and thus an alternative connection schematic was created for connecting these SONAR distance sensors to an Arduino Uno R3, as seen in figure 7. All VCC (5V) and GND (ground) pins were wired in series, but the TRIG (trigger of signal) and ECHO (return of signal) pins needed their own digital pins on the Arduino.
The RPLiDAR-A1 also couldn't be used on the Mirte's OPI. Since the OPI is a low level integrated computer, it does not rely on a lot of accessible random access memory. The around 480 megabytes of RAM were by far not enough to process or relay the LiDAR's point cloud data, which caused the LiDAR to crash after a couple of seconds of receiving power. A solution would have been to assign more memory to a swap file to increase the amount of effective random access memory the OPI could use, but this method in itself posed several problems regarding stability. The choice was made to directly connect the LiDAR scanner to a laptop to receive and process the point cloud data instead of relaying the information through a microprocessor of microcontroller.
Even though the reception of the point cloud was simplified, the optimal mounting location for the LiDAR in the center of the Juno robot was not accessible, due to the hay bales being too low for mounting on the top and the insides of the robot being too cluttered with electronics. This prompted the search for a

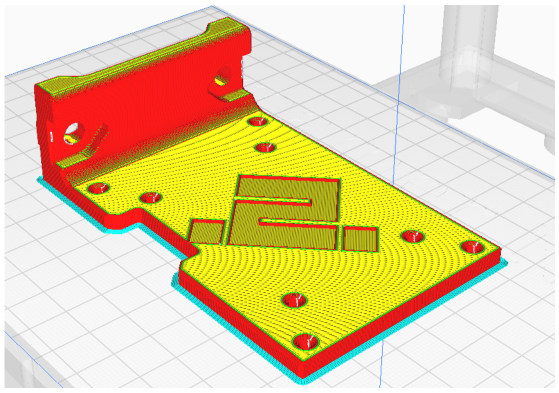


Figure 8: Open source, 3d printable RP-LiDAR mount, designed by David M, used for mounting the LiDAR scanner on the Lely Juno Robot.

specific 3D printable sensor mount, for relative easy installation on the front of the robot, near the ground. The open source mount chosen, is made by David M and downloadable at https://www.grabcad.com and can be seen in figure 8.