# Autonomous Robot Design to Extract The Plan of Field and Objects

**FINAL REPORT**

For

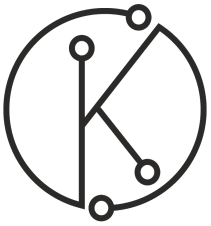
**Dr. Arzu KOÇ**

Middle East Technical University

Electrical and Electronics Engineering

D-124/2

By



**ÖZ KARDEŞLER OTOMASYON**


| Mustafa KILINÇ | Oğuz ÖZDEMİR | Uğur AÇIKGÖZ | Sami Alperen AKGÜN | Furkan GÜLLÜ |
|---|---|---|---|---|
| *2094092* | *2094969* | *1936350* | *2093185* | *1876218* |
| 0536-543-7170 | 0531-465-7536 | 0555-805-8954 | 0530-961-2447 | 0506-832-4664 |


Project Duration

**09.11.2018 - 03.06.2019**

Date of Submission

**10.05.2019**

# Table of Contents

# 1. Executive Summary

Recent leaps in autonomous robotics have expanded the range and complexity of tasks that robots can perform, making them increasingly suitable for real-world use. The Simultaneous Localization and Mapping (SLAM) problem is one of them and it is concerned with building an autonomous robot that can identify its position in an unknown environment, while creating a map of the environment it is placed in. SLAM has been solved by the Robotics community and several algorithms already exist but the cost of the solutions are very high and the solution to the localization and reaching to a destination by an autonomous robot is not that easy because of the existence of noise and errors in the sensor readings.

The aim of this project is to build an cost-effective autonomous robot which has a unique LIDAR design. We used a Raspberry Pi to run the SLAM algorithms. We have demonstrated an affordable implementation platform using low cost computational hardware and open-source algorithms. Utilizing the wrappers offered by Robot Operating System [ROS], STM32 and VL53L0X based LIDAR and a Raspberry-Pi based robot has been successfully built.

To solve simultaneous localization and mapping problems and to create robust systems, we used complex computer algorithms and multitude of sensor technologies. Our robot uses landmarks to determine its location and identify objects and walls using its Time of Flight sensors. Since the start of the project, it has been quite a bit of a diverse experience for us due to unorthodox requirements in the problem we are dealing with. Yet, with its dynamic structure, our company has succeeded in making significant amount of progress in a limited amount of time. And finally, Öz Kardeşler Otomasyon produced a robot which has ability to localize itself in a precise way and create a map of unknown environments in the shortest time possible with a wireless and compatible LiDAR unit which is embedded to our first commercial product, Kamu Robotu.

# 2. Introduction

Autonomous robots are used to explore environments that are difficult for humans to explore. For example, space exploration, investigations of dangerous sites, exploration of oceans and study about marine creatures. Also, autonomous robots can be used as guides in buildings to show directions. For all the listed applications the robot has to localize in an environment and move to destination. In a fully autonomous system, robot left at an arbitrary position in a place where the coordinates of landmarks and destination are known. To detect distances from landmarks and objects, robots use different kind of sensors such as time of flight sensors. While navigating, robot uses encoders to estimate its position. But the solution to the localization and reaching to a destination by an autonomous robot is not that easy because of the existence of noise and errors in the sensor readings.

Therefore, our company Öz Kardeşler Otomasyon was founded to offer an affordable platform which has the capability to navigate autonomously and extract map of the unknown environments. For this purpose, we designed Kamu Robotu.

Our platform named Kamu Robotu is composed of two main units which are odometry and LIDAR. LIDAR unit is used to receive detailed information about the unknown environment and its information is used by SLAM (Simultaneous Localization and Mapping) algorithm to extract map. There exists various LIDARs available in the market, but they are not cost friendly. Hence, our company designed a unique LIDAR named OKO LIDAR. This LIDAR includes four Time of Flight Sensors which are rotated by a step motor to measure distance. Thanks to our elegant design of wireless power submodule and optical encoder, OKO LIDAR promises budget friendly and efficient range measurement device for autonomous systems. Odometry unit consists of two DC motors with encoders to estimate robot's position while moving. By combining odometry information with OKO LIDAR we localize Kamu Robotu precisely. Robot Operating System (ROS), globally known and widely used middleware software in robotics, is responsible for connection between different subunits.

This report will explain details of all production and design process of Kamu Robotu. Overall design, test results and technical analysis will be shared.

# 3. Design Description

## 3.1 Overall Block Diagram

Overall block diagram can be seen in Figure 1.

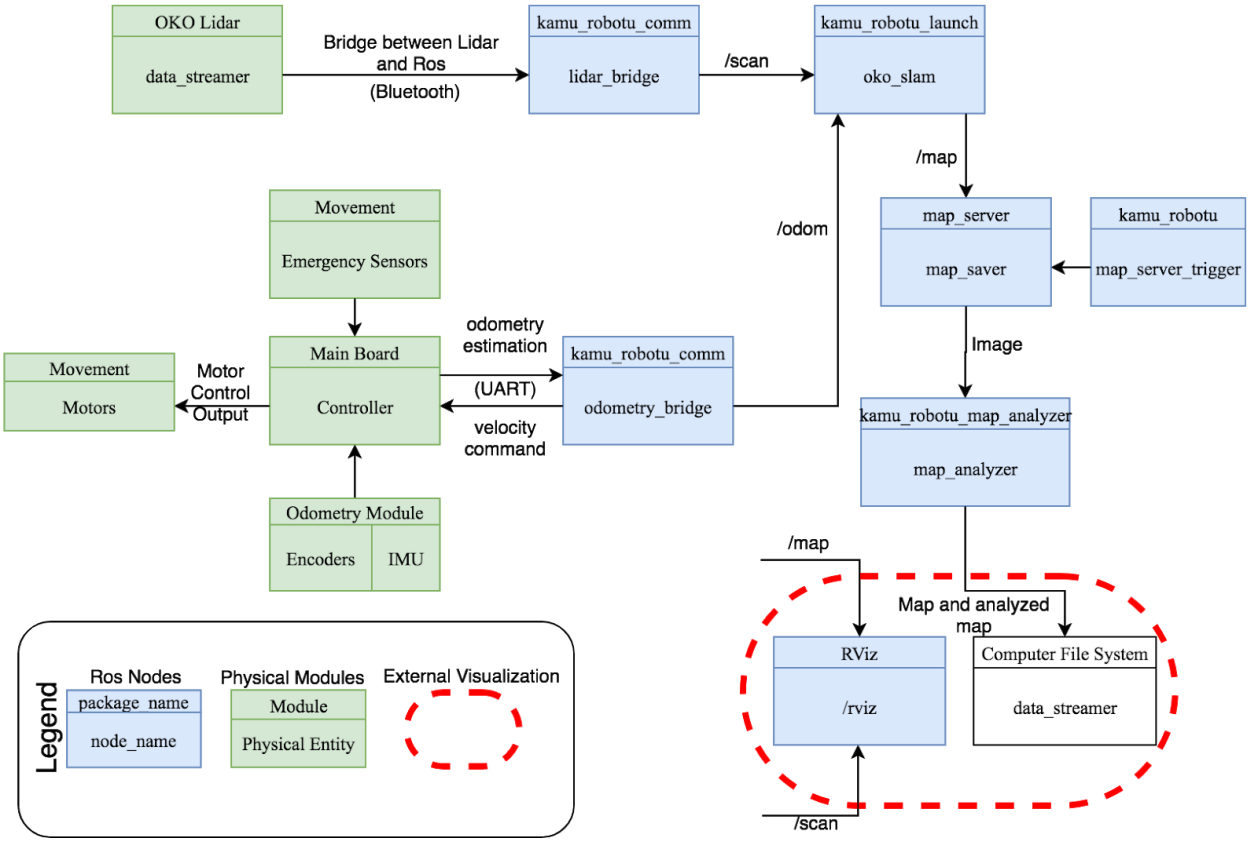



Figure 1. Overall Block Diagram

Inspecting Figure 1, green filled blocks represents physical blocks while blue filled blocks represents ROS packages/nodes. Connection labels describes the data content that is transferred. Three microcontrollers are used synchronously in this project as can be seen in Table 1. Further details about overall block diagram and submodules will be explained in the following sections.

Table 1. Division of Labor in Microcontrollers

| **Raspberry Pi 3** | **Main STM** | **STM on the OKO LIDAR** |
|---|---|---|
| Converting incoming distance data to scan | Odometry measurement from Encoders | Data acquisition from proximity sensors |
| Creating map and finding object positions/orientations | Motor control | Position reading from absolute encoder |
| Transmitting created map and positions/orientations | Odometry data transmission to Raspberry Pi | Transmission of the distance measurements to Raspberry Pi |
| | Power Controls of Raspberry Pi and OKO LIDAR | |

## 3.2 Main Body

Main body design is one of the most important design tasks for a feasible and cost friendly robot. Considering whole design, our robot can be placed in an upright cylinder with 21 cm diameter and 17 cm height as stated in the standarts. CAD rendering of the design and kamu_robotu in an environment ( kamu_map ), which perfectly satisfies standards determined, in Figure 2.

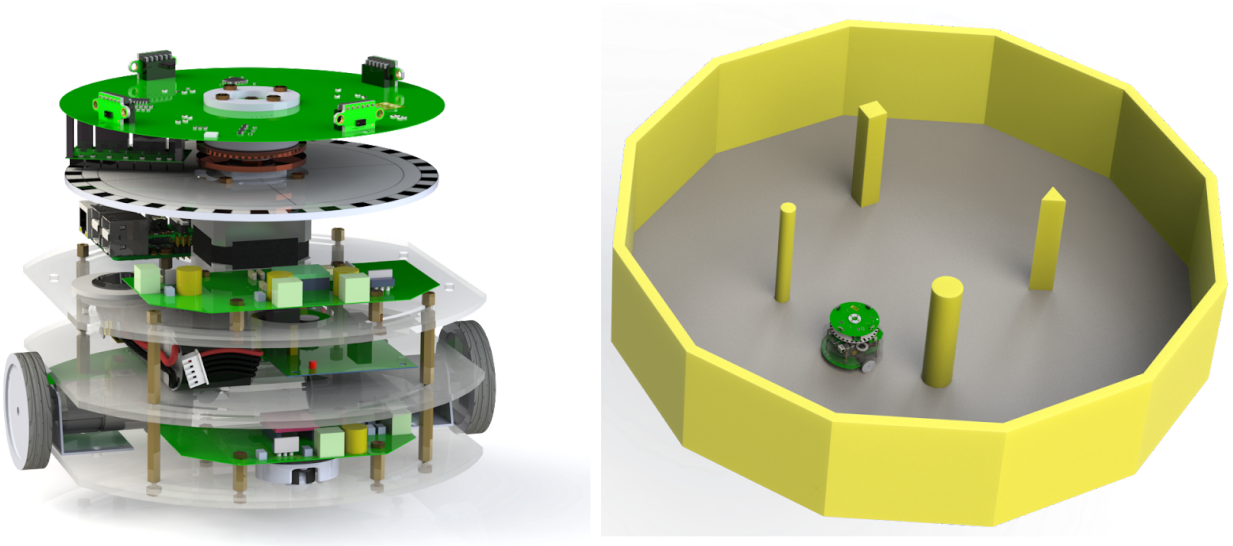


Figure 2: Mechanical Design and Kamu robotu in an environment that suits dimensions in project description

The final product can be seen in Figure 3.

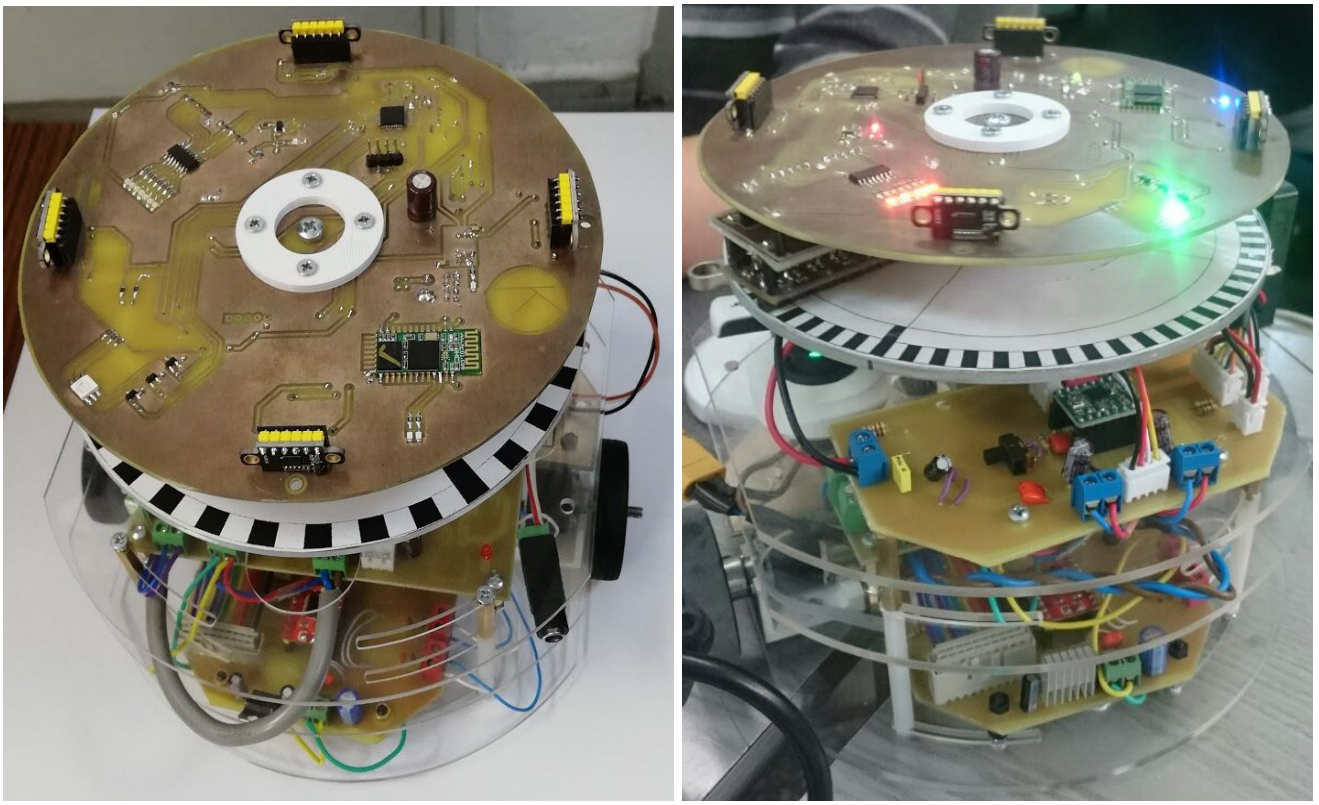


Figure 3: Pictures of Kamu Robotu not working (left) and working (right)

## 3.3 OKO LIDAR

OKO LIDAR collects information about the environment by using VL53L0X Time of Flight (ToF) sensors. Although VL53L0X requires 30ms to give accurate measurement result and I2C interface, it is preferred over other alternatives because its measurements are linear and independent of characteristics of the measured environment. OKO LIDAR sends measured data via bluetooth connection.

A Nema 17 stepper motor with 1.8 angle resolution (42SHD0001) is used to rotate LIDAR unit. We choose to work with it due to its synchronous drive mechanism. To be able to measure the angle of the lidar with respect to robot, we used optical encoder with 128 point incremental pattern and 1 point index pattern. We are using wireless power to run the LIDAR.

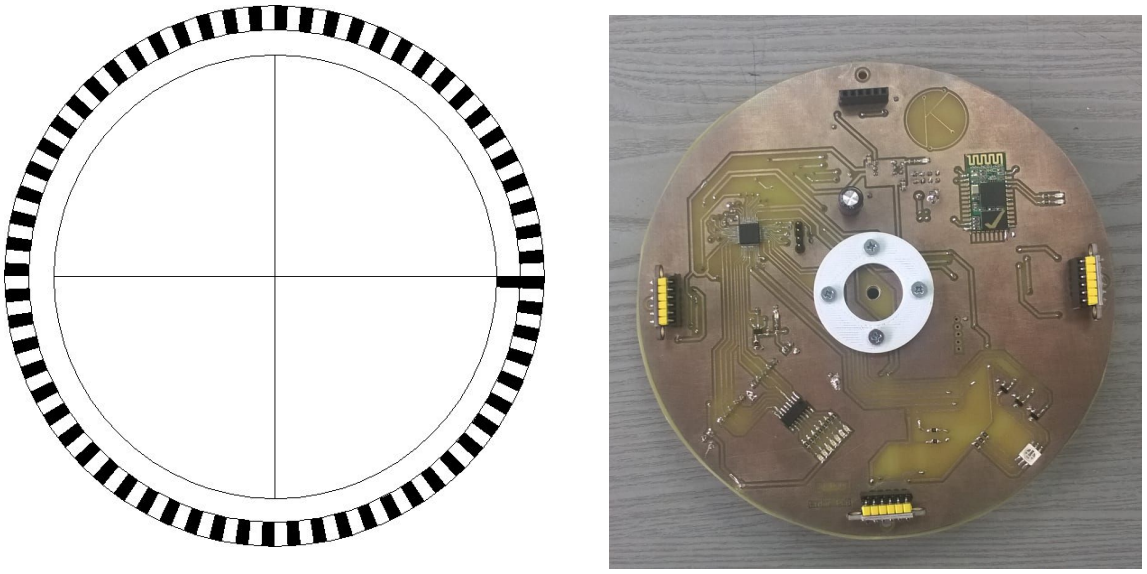


Figure 4: 128 Point Optical Encoder Pattern and LIDAR Board

The tests regarding OKO LIDAR unit is as follows in the Table 2.

Table 2 : Tests Regarding OKO LIDAR

| **Topic** | **Result** |
|---|---|
| Interference with bluetooth | No interference is observed due to its operating frequencies being quite distinct. |
| Power delivery limits | We calculated required power for the OKO LIDAR as approximately 400mA. So we tested with its limits on both partially complete actual system and with resistive load. And its delivery rate was enough. |
| Contrast detection test | We tested one of the QRD1114 sensor which is the functional component of the absolute encoder submodule. In test setup, we used a rotating disk with white/black pattern and observed the output of the sensor. As a result we could observe the pattern on the wheel exactly from the output of the sensor. |
| Field of view of the sensor (25°) | We tested whether sensor can distinguish a gap between objects being 5 cm apart from each other with a distance 70 cm to the sensor. And be observed that when there is a object that can be seen from the gap, sensor could sense it but when there is no object can be seen from the gap, sensor acted like there was simply a wall. |
| The bluetooth connection test | A ROS node is created as a bridge between OKO LIDAR module that reads sensor data using STM and ROS on Raspberry-Pi. This node also converts taken messages from LIDAR to ROS data type laserscan which is used by rest of the ROS structure. |

## 3.4 Odometry

Odometry unit has two Namiki 22CL-3501PG 12V DC Motor with Encoder and main STM Controller. By using these sensor measurement we need to develop a control algorithm for position estimation. We can easily combine the sensor measurements in ROS environment.

We used nav_msgs/Odometry message for two different ROS topics which are odom topic and vo topic. The formal one is for encoder measurements, the letter one is for a visual sensor measurements. We obtained 2D odometry data from each motor encoder and our visual sensor unit which is LIDAR. Data structure of the nav_msgs/Odometry ROS message can be seen below.

- geometry_msgs/PoseWithCovariance pose : Position information for each coordinate axis.
- geometry_msgs/TwistWithCovariance twist : Velocity measurements for each coordinate axis and the rotational velocity of the robot.

Each sensor source can appear and disappear over time robot_pos_ekf ros package will continue to estimate the position, since each sensor unit operate at different rates and with different latencies.

An additional ROS package might be necessary to convert laser scan data of the LIDAR to an odometry data. Hence, we will use rf20_laser_odometry ROS package to observe odometry data. Control algorithm schematics of the odometry unit can be seen in Figure 5.

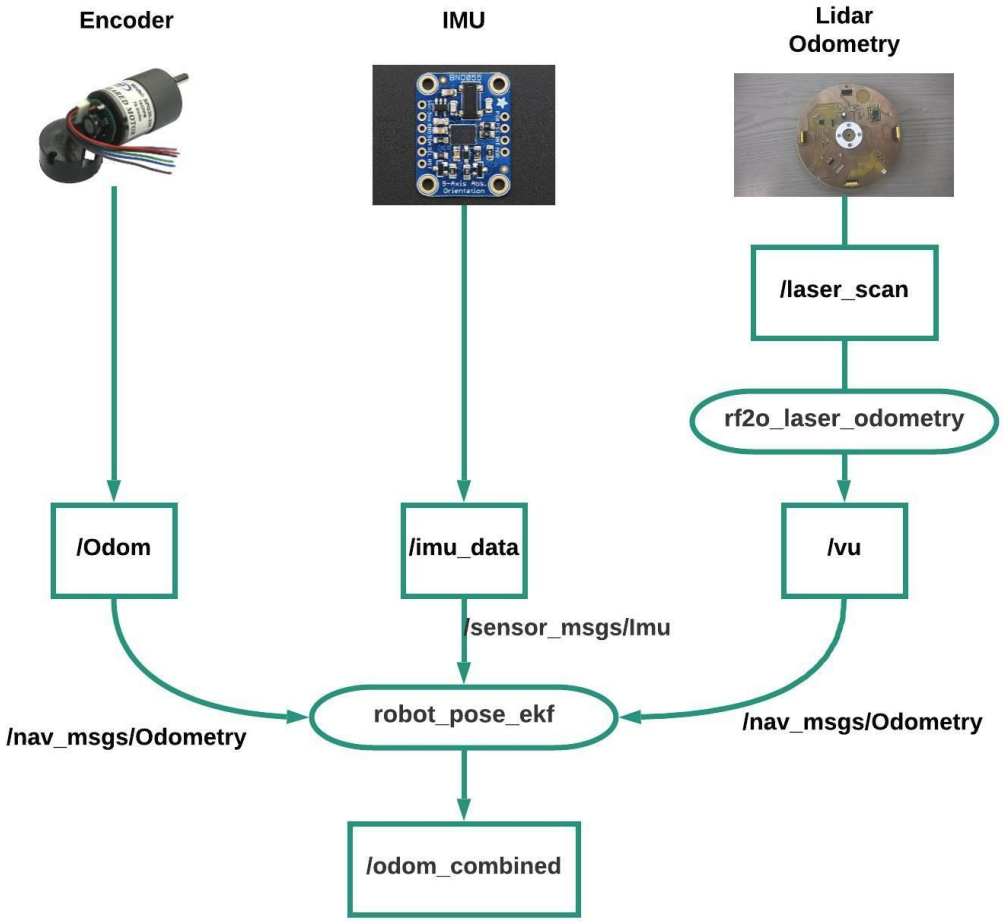



Figure 5 : Control Algorithm Schematics of the Odometry Unit

## 3.5 Power Subsystem

The requirements of power systems are to control battery charge level in order to avoid above dangerous discharge level to protect the batteries and control the voltage of regulator output in order to detect any overload situation. Also, this subsystem is required to output enough power for all subsystems.

To satisfy voltage monitor requirements, our power subsystem has an STM32 microcontroller which periodically reads necessary voltages through its ADC. Also it is capable of turning on and off the regulator. To satisfy power output, our subsystem has a switching mode regulator, LM2596-5V. Figure 6 shows the overall power distribution diagram.

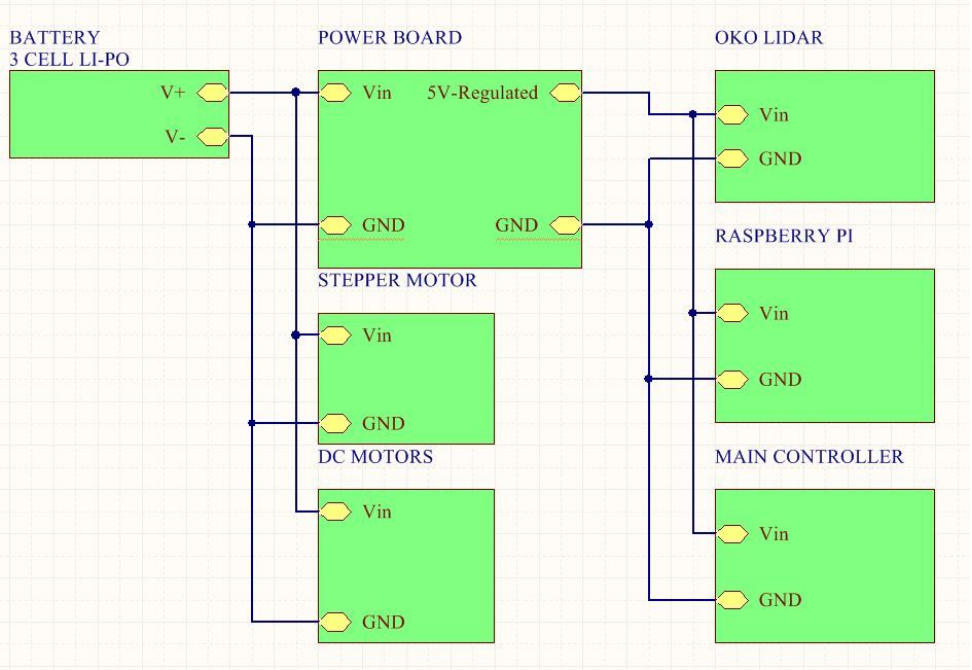



Figure 6: Power Distribution Diagram

To calculate total power consumption of the robot the following currents are measured :

- LIDAR board draws 350mA at 5V supply, since the efficiency of wireless power is 70%, it makes about 438mA from regulated 5V
- Stepper Motor of LiDAR draws 0.5A from 12V
- Raspberry Pi 3 draws around 1.5A at maximum CPU load from regulated 5V
- DC motors draw 1.5A at max load from 12V
- Main Board draws 500mA maximum from regulated 5V.

At total, it makes 2438 mA from regulated 5V and 2000 from 12V. Since our regulator has 85% efficiency, finally it makes 3168 mA at 12V which is 38 Watts. Since this calculation is all the power we need and we designed the power subsystem according to it. Figure 7 Shows the power management board.

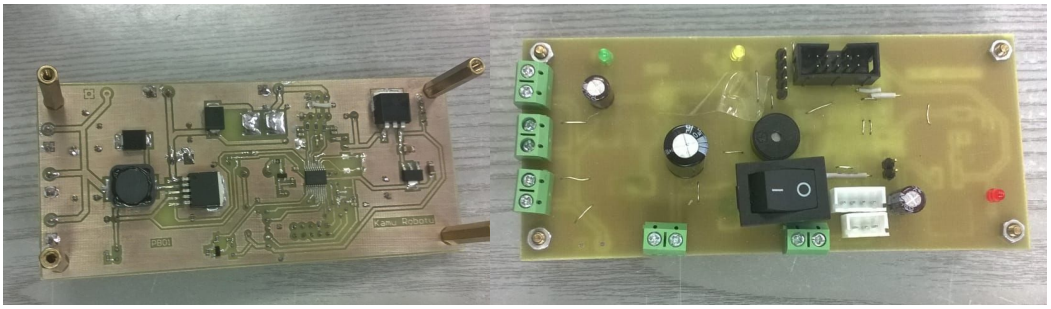


Figure 7: Power Management Board

## 3.6 Main Controller Subsystem

This subsystem is required to handle DC motors and their encoders, stepper motor, power outputs to OKO LIDAR unit and Raspberry Pi. For this purpose we designed DC Motor controller board which has a TB6612 DC motor controller and 2 connectors for encoder input, stepper motor driver board which has an A4988 stepper motor driver, various MOSFETs and various connectors, and controller board which has an STM32 microcontroller and various connectors. It also has a bluetooth for bootloader implementation.

Main controller board, stepper motor driver board and DC motor driver board can be seen in Figures 8-10 respectively.

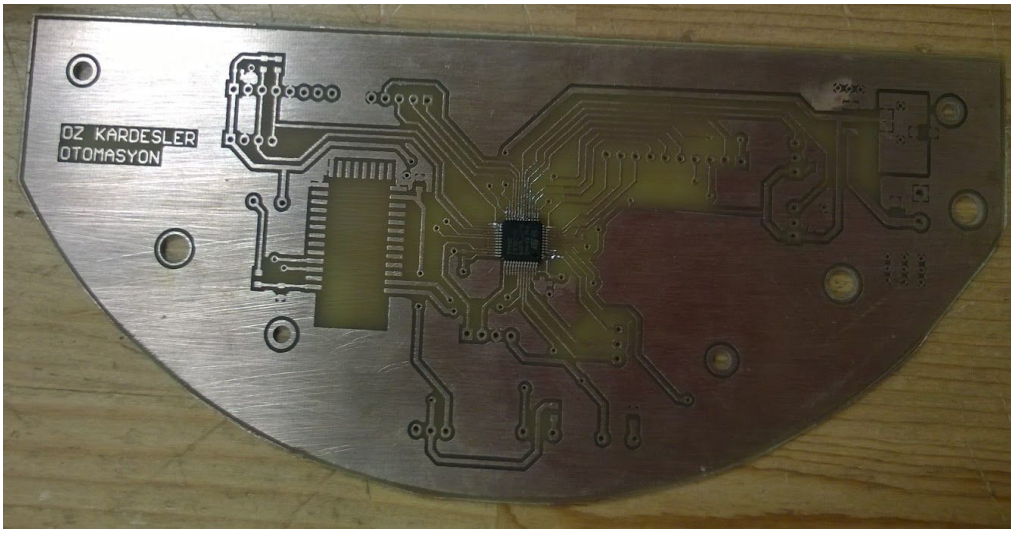



Figure 8: Main Controller Motherboard

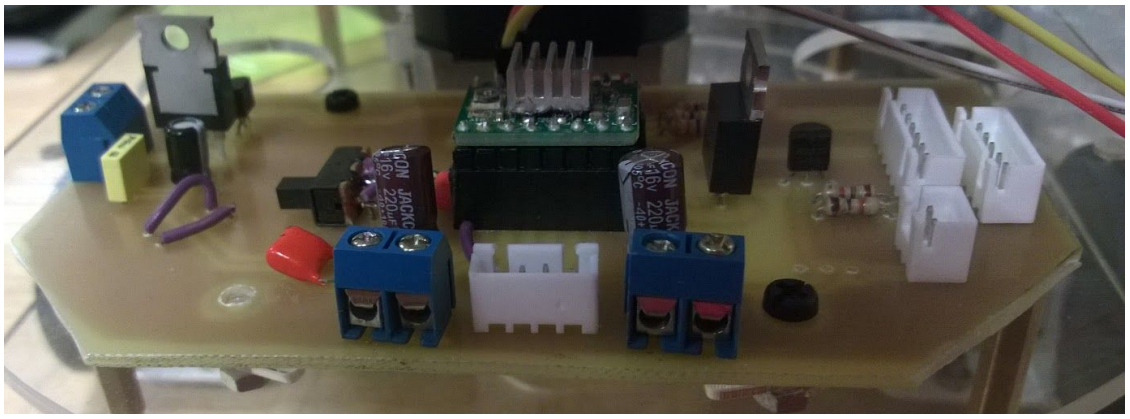


Figure 9: Stepper Motor Driver Board

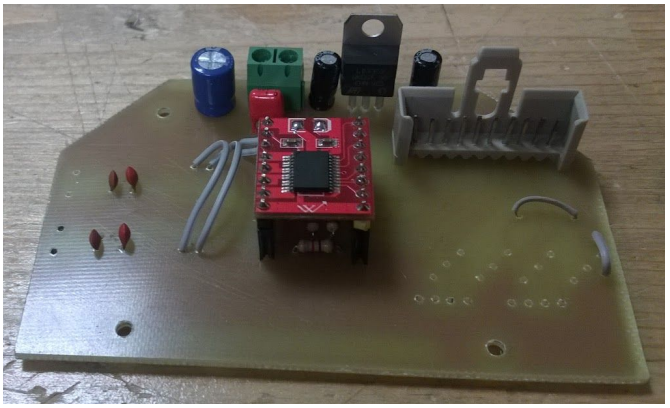


Figure 10: DC Motor Driver Board

## 3.7 Graphical User Interface

As Öz Kardeşler Otomasyon, we designed a GUI for operating the robot easily. It requires a bluetooth hardware to connect to the main controller subsystem. User may choose from automatic mod and manual mode, and navigate the robot in manual mode with this GUI.

The GUI also provides the image of lastly calculated map in a pop-up window. Currently, it only supported on Windows operating systems. Figures 11 and 12 shows the GUI.

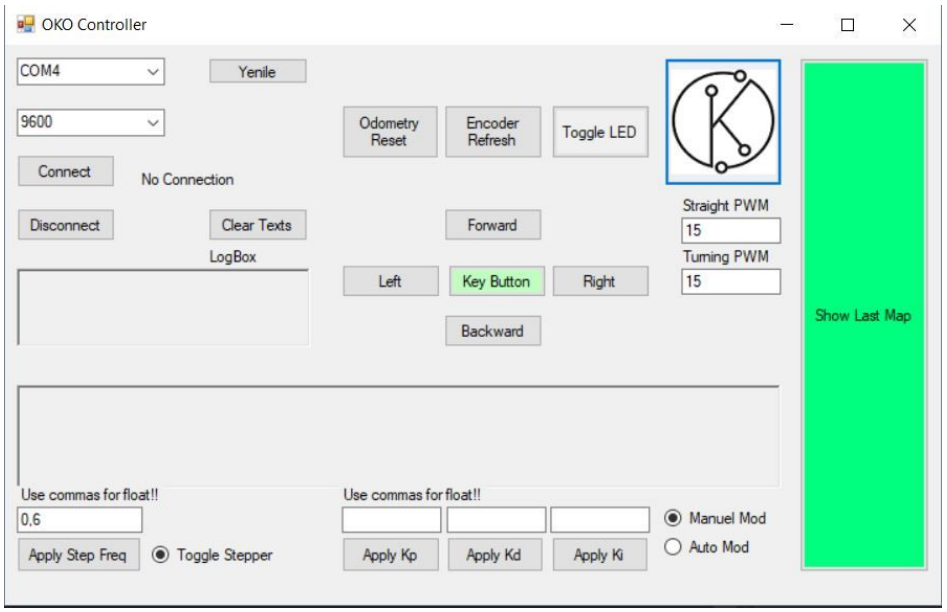



Figure 11: Graphic User Interface

,

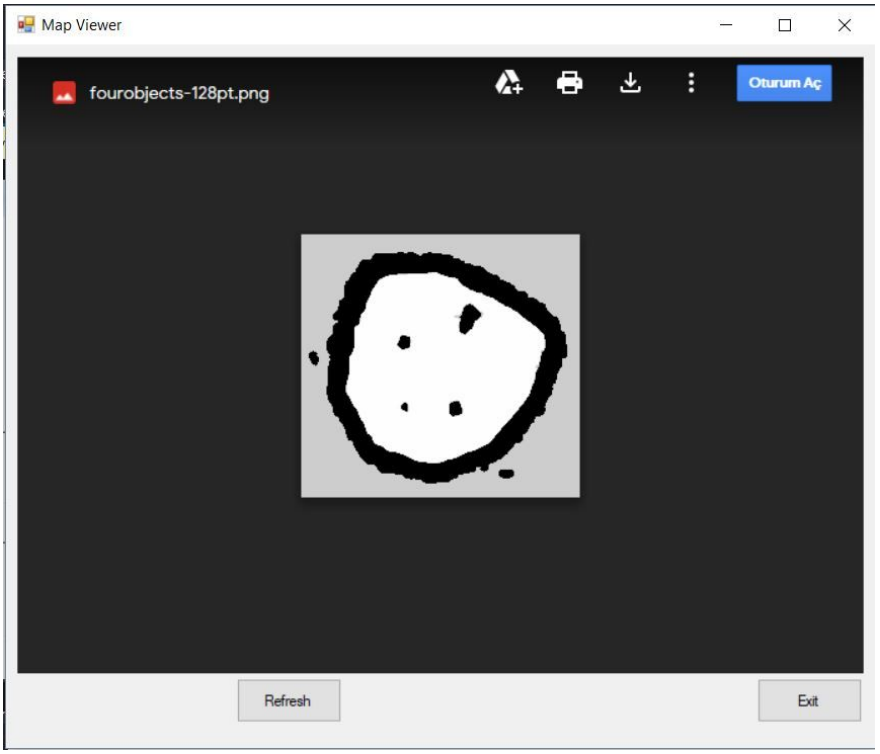



Figure 12: Map Viewer window

## 4. Results and Analyses of Performance Tests

In this section, different SLAM algorithms that we have used will be explained. At the end our decision regarding SLAM algorithm will be given by comparing simulation and real life mapping performance.

There are many different SLAM algorithms which are implemented in ROS middleware. Hector-SLAM, Gmapping and Cartographer are the most efficient and powerful ones among them [7]. As Öz Kardeşler Otomasyon, we fine tuned each of these algorithms and try to figure out which one is perfect for our application. One can see the simulation and real environment that we used during tests in Figure 13 and Figure 14 respectively.

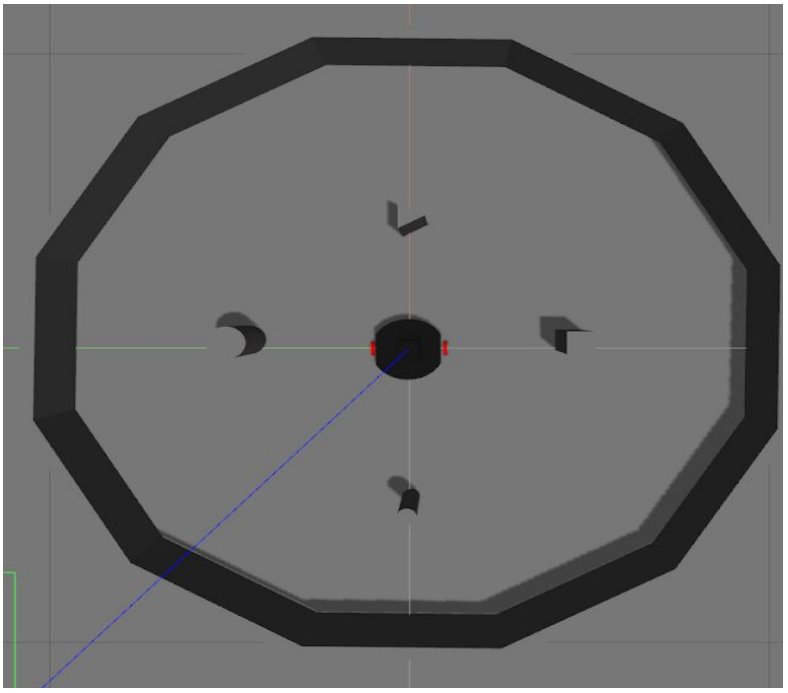


Figure 13: Simulation environment to test SLAM algorithms

Figure 14: Real life environment to test SLAM algorithms

First of all, Hector SLAM is a SLAM algorithm which only relies on laser scan measurements obtained by LIDAR. It assumes LIDAR is fast and accurate enough. It is based on robust scan matching process and it outputs occupancy probabilities which is used to illustrate the map. Since our LIDAR frequency is not fast enough, Hector SLAM does not provide good results for our case.

Gmapping is the most widely used SLAM package in ROS and it is particle filter based SLAM approach. It combined odometry and laser scan information. It achieves to give good results with even small particle numbers (30-100) thanks to this unique combination. Result related to gmapping can be seen in Figure 15. However, as it can be clearly seen it does not give promising results for our case. Hence, we decided to abandon to use it for final product.

Cartographer SLAM is the newest SLAM algorithm developed by Google. It is based on backmap mapping method which is basically divides the overall map into submaps and try to achieve the maximum efficiency in these submaps. Then, these high quality submaps are combined in an efficient way. Cartographer results obtained in simulation and real environment can be seen in Figure 15. These results are the most satisfying results that we have obtained, so we decided to use Cartographer SLAM for our final product.

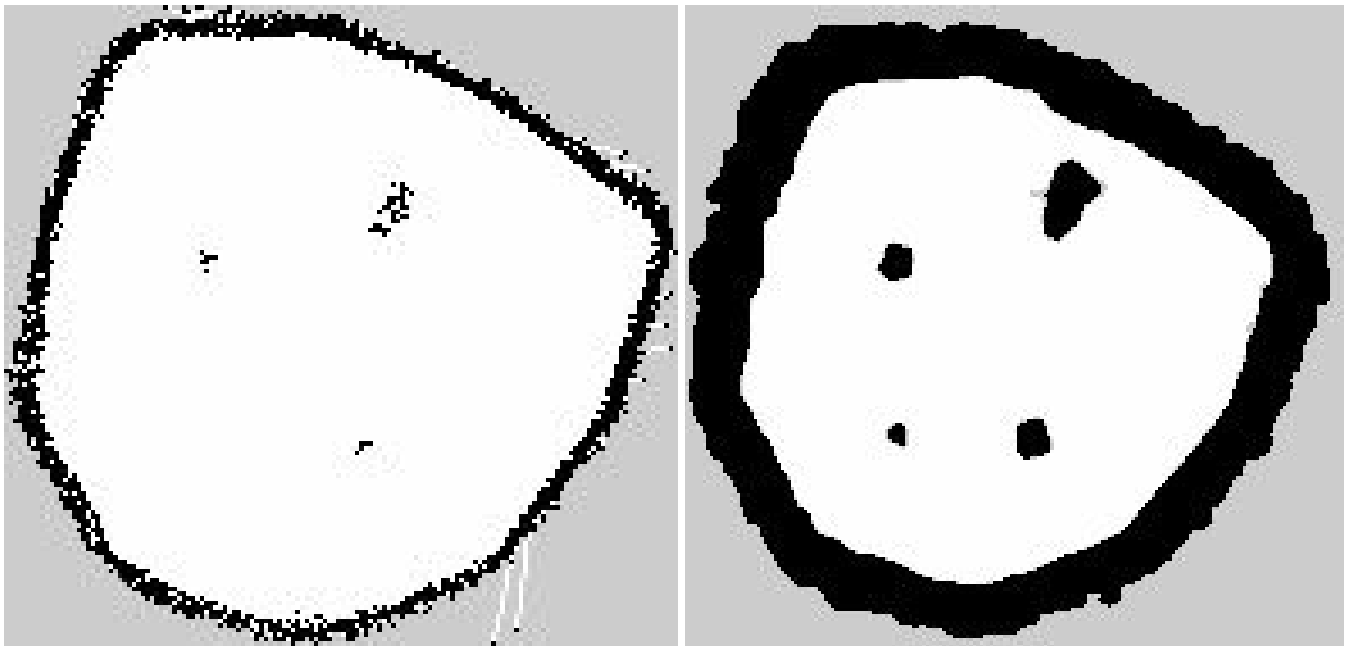


Figure 15: Slam results from same data recorded (Gmapping on the left, cartographer on the right)

The problem of autonomous navigation in an unknown map is called as exploration. For exploration problem, we tried two different exploration packages that are frontier-exploration and explore-lite ROS packages. After that, we created our own navigation algorithm based on finite state machine logic.

Frontier exploration algorithm is based on navigating around detected frontiers (unknown regions in the map). The algorithm itself is quite successful, but it has a significant drawback which is the need of initial exploration area. In other words, one need to provide an initial exploration area to the algorithm. This could be impractical for our case. Although we have tested this autonomous navigation algorithm in the simulation as it can be seen in Figure 16, we decided to abandon it for real life usage since it is not applicable for our case.

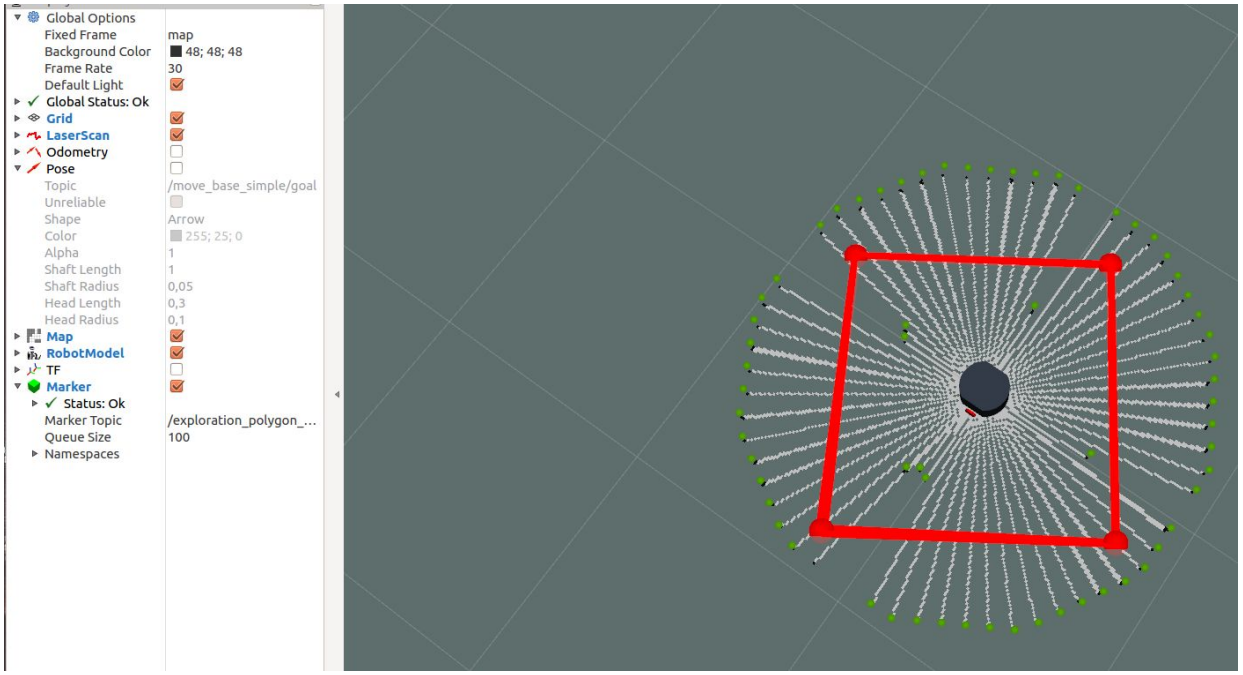



Figure 16: Frontier exploration package during the retrieval of initial exploration area input

Explore-lite algorithm is also based on frontier exploration. However, it works in a greedy way so that there is no need to give an initial exploration area input to the algorithm. Therefore, it is a good candidate for our scenario. A visualization of frontier exploration algorithm can be seen in Figure 17.

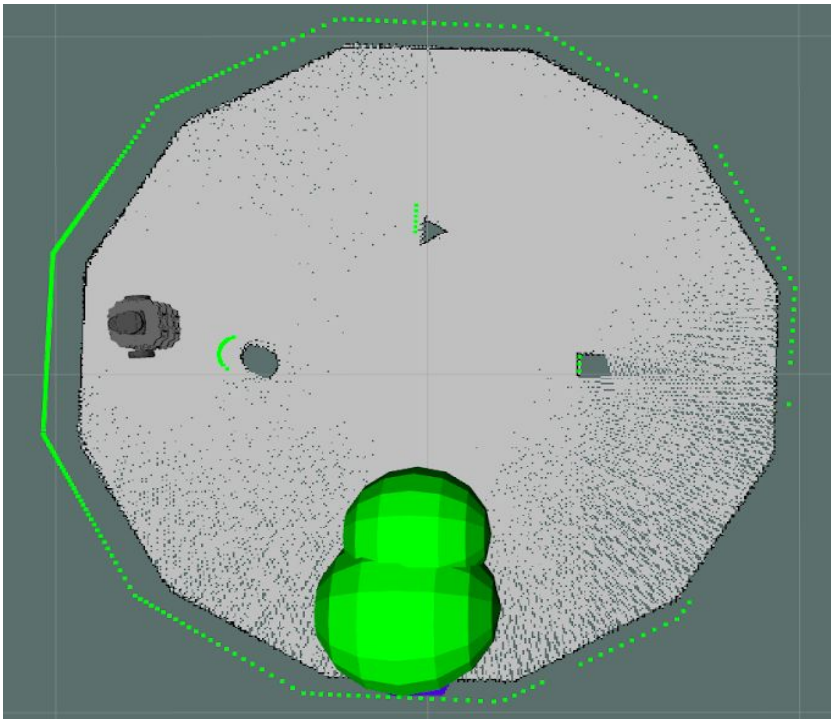
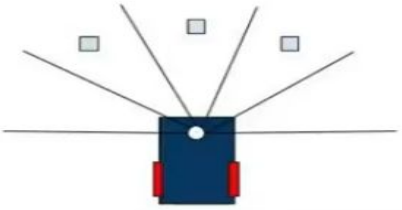


Figure 17: Explore-Lite Algorithm Result in Simulation (Green Bubbles are frontiers that are not discovered)

Lastly, our algorithm is a simple finite-state machine. We divided laser scan data we have obtained into different regions and then we take minimum of the laser data to detect closest object/wall in that region. Divided regions around the robot are used in finite state machine to navigate autonomously. Three of these regions and corresponding state table can be seen in Figure 18. For example, Case 2 corresponds to the case where there is an object/wall in the North direction. This algorithm is used in the final product since it is enough for our scenario and it is easy to modify.



| | Front Left | Front | Front Right |
|---|---|---|---|
| Case 1 | | | |
| Case 2 | | x | |
| Case 3 | | | x |
| Case 4 | x | | |
| Case 5 | | x | x |
| Case 6 | x | x | |
| Case 7 | x | x | x |
| Case 8 | x | | x |

Figure 18: Kamu_robotu_navigation algorithm explanation

## 5. Budget

Cost analysis of the final product can be seen in Appendix A.

Total cost, including engineering costs for development, is 1.380 TL. Since Dollar / TL exchange rate is changing a lot, we calculated it in TL.

## 6. List of Deliverables

“Kamu Robotu” produced by Öz Kardeşler Otomasyon has abilities:

- Localizing itself in a precise way
- Creating a map of unknown environments in the shortest time

The company itself will provide following services:

- 2 year guarantee of the product
- Technical assistance in case of any issue

- Consultancy for Simultaneous Localization and Mapping (SLAM) problems

The system will involve a complete package which includes:

- A robot,
- A battery pack,
- A user manual that involves a quick start guide, operating conditions, safety notices, instructions and procedures, troubleshooting section, system requirements, warranties, and reference information section. User manual can be seen in Appendix B.
- A developer manual which involves technical details of the algorithms.

## 7. Discussion

The main danger about Kamu Robotu is related to lipo battery. Lipo batteries have to be carefully charged and discharged. If one can not provide a controlled current to feed the battery itself during the charging process, battery might blast or starts to fire during the usage. Therefore, one have to be very cautious about lipo batteries. For example, we use 3 cell lipo battery and the operating voltage range is 10.5V and 12.6V. If the voltage of the battery exceeds these boundaries significantly, then the possibility that battery starts to fire increases exponentially.

Since it is practically difficult to measure the voltage of the battery and check the balance between cells of the battery all the time, we implemented a notification system in power subunit. As it is mentioned before, this unit measures the voltage of lipo battery and it notifies the user when the battery comes close to run out. The notifications are done by the buzzer through sounds as follows:

- If lipo battery voltage is between 11.5V and 11V, the buzzer bleeps shortly in each 10 seconds.
- If lipo battery voltage is between 11V and 10.5V, the buzzer bleeps long in each second.
- If lipo battery voltage is less than 10.5V, the power unit directly shuts down the power.

As Öz Kardeşler Otomasyon, we designed Kamu Robotu in such a way that it could be easy to use by other people. We especially designed it as ROS compatible so that any hardware or software related to Kamu Robotu will be used by the community. We believe that its impact will be huge in the robotic community especially for the people who conduct research on SLAM and autonomous navigation since Kamu Robotu is an alternative affordable platform to test different algorithms and it is quite easy to modify the design itself. We submitted a paper to Turkish National Robotics Conference (ToRK 2019) to share our knowledge with other people. Moreover, all documentation, hardware, mechanical designs and software related to Kamu Robotu will be shared with public through Github at the end of semester.

There is not any serious potential environment effects of Kamu Robotu. However, it may cause headaches if it is extensively used due to high pitch noise by stepper motor. Otherwise, the robot does not have any potential harm to environment.

## 8. Conclusion

In this report, information related to our robot named Kamu Robotu is given. The main function of the robot is to extract the mapping of the surrounded environment while navigation autonomously at the same time. Kamu Robotu provides an open-source, ROS compatible, easy to build and modular robotic platform to people who study on mapping problem. It has unique design of LIDAR to sense the environment. The odometry information is achieved thanks to encoders embedded in two DC motor of the robot. SLAM and navigation algorithms runs in Raspberry Pi 3 embedded board under Robot Operating System (ROS) middleware. Low level control of the robot is done thanks to ARM Cortex M10 STM microprocessors. The connection between STM and Raspberry-Pi boards are achieved through serial and bluetooth communication. It is highly recommended for users to follow user manual to get deep knowledge about the algorithms and design of the robot. Thanks to Kamu Robotu, it becomes unnecessary to use expensive robotic platforms or LIDARs to get a map of a small scale static indoor environment.

# 9. References

[1] Groups.csail.mit.edu. (2018). *Robo*. [online] Available at: https://groups.csail.mit.edu/drl/courses/cs54-2001s/diffdrive.html [Accessed 26 Dec. 2018].

[2] Wiki.ros.org. (2018). *hector_slam - ROS Wiki*. [online] Available at: http://wiki.ros.org/hector_slam [Accessed 26 Dec. 2018].

[3] Core.ac.uk. (2018). [online] Available at: https://core.ac.uk/download/pdf/29175747.pdf [Accessed 26 Dec. 2018].

[4] Ieeexplore.ieee.org. (2018). *A flexible and scalable SLAM system with full 3D motion estimation - IEEE Conference Publication*. [online] Available at: https://ieeexplore.ieee.org/document/6106777 [Accessed 26 Dec. 2018].

[5] Wiki.ros.org. (2018). *gmapping - ROS Wiki*. [online] Available at: http://wiki.ros.org/gmapping [Accessed 26 Dec. 2018].

[6] Wiki.ros.org. (2018). *cartographer - ROS Wiki*. [online] Available at: http://wiki.ros.org/cartographer [Accessed 26 Dec. 2018].

[7] B. M. Da Silva, R. S. Xavier, T. P. Do Nascimento, and L. M. Gonsalves, "Experimental evaluation of ROS compatible SLAM algorithms for RGB-D sensors," Proceedings - 2017 LARS 14th Latin American Robotics Symposium and 2017 5th SBR Brazilian Symposium on Robotics, LARS-SBR 2017 - Part of the Robotics Conference 2017, vol. 2017-Decem, pp. 1–6, 2017.

# 10. Appendices

## Appendix A. Cost Analysis

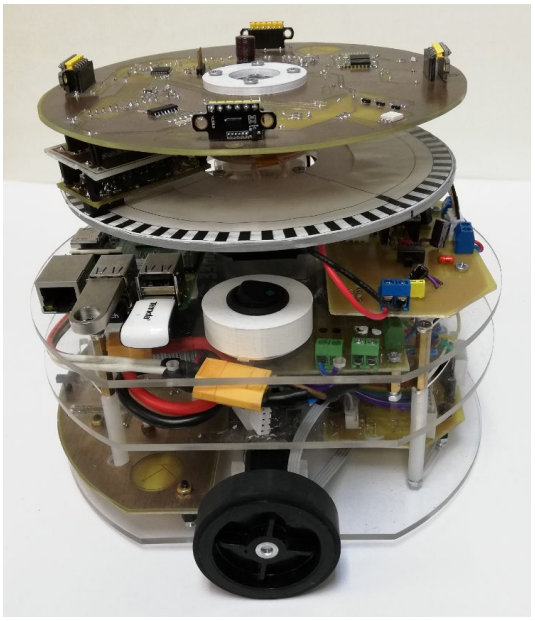
Cost analysis of the project can be seen in Table A.1.

| | Equipments | Prices per unit | Amount | Prices |
|---|---|---|---|---|
| Main Solution | **a) Drive Unit** | | | |
| | DC motor | 10$ | 2 | 20$ |
| | Differential drive wheel | 5$ | 2 | 10$ |
| | Ball Caster | 1$ | 2 | 2$ |
| | PCB | 4$ | 1 | 4$ |
| | **b) Sensor Unit** | | | |
| | Encoder | 1$ | 2 | 2$ |
| | Lidar | 48$ | 1 | 48$ |
| | **c) Microprocessor Unit** | | | |
| | Raspberry Pi | 35$ | 1 | 35$ |
| | STM Microprocessor | 3$ | 3 | 9$ |
| | Motor Driver IC | 1$ | 2 | 2$ |
| | **d) Power Unit** | | | |
| | PCB | 3$ | 1 | 4$ |
| | Components | 7$ | 1 | 3$ |
| | **e) Main Controller** | | | |
| | PCB | 8$ | 1 | 8$ |
| | Components | 7$ | 1 | 7$ |
| | **e) Other Components** | | | |
| | Li-po Battery | 20$ | 1 | 20$ |
| | Spacers | 0.08$ | 20 | 1.6$ |
| | 3D Printings | 0.07$ | 80 gr | 5.7$ |
| | Laser Cuttings/Plexiglass | 8.3$ | 1 | 8.3$ |

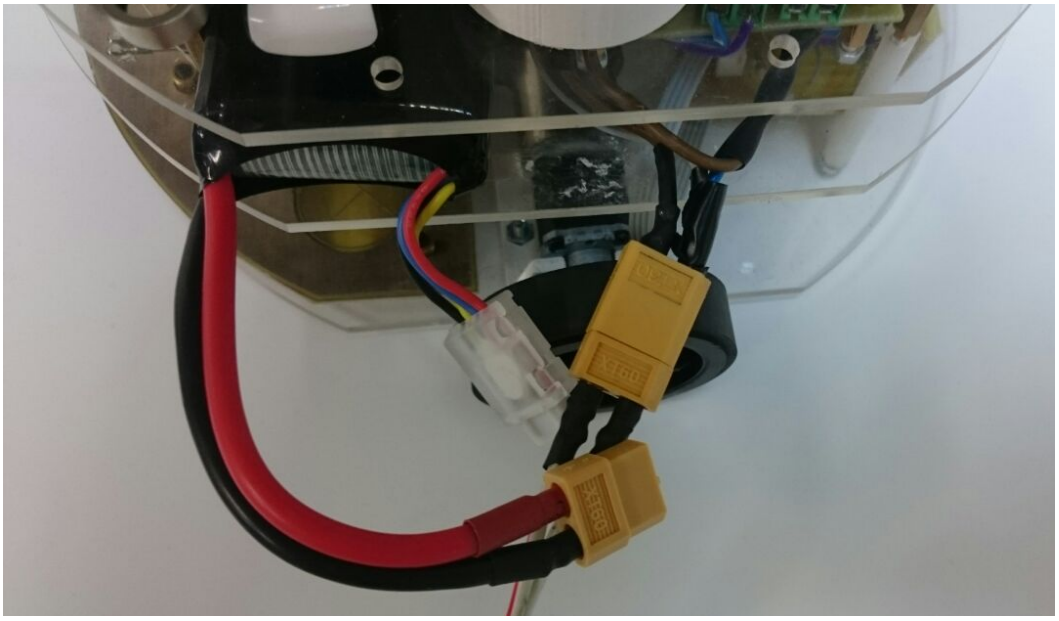
| | | | | |
|---|---|---|---|---|
| | Flash Disc for RPi3 | 2$ | 1 | 2$ |
| | SD Card for RPi3 | 2.3$ | 1 | 2.3$ |
| | Cables | 1.5$ | 1 | 1.5$ |
| | Wifi dongle | 2.1$ | 1 | 2.1$ |
| | **TOTAL** | | | **197.5$** |

## Appendix B. User Manual

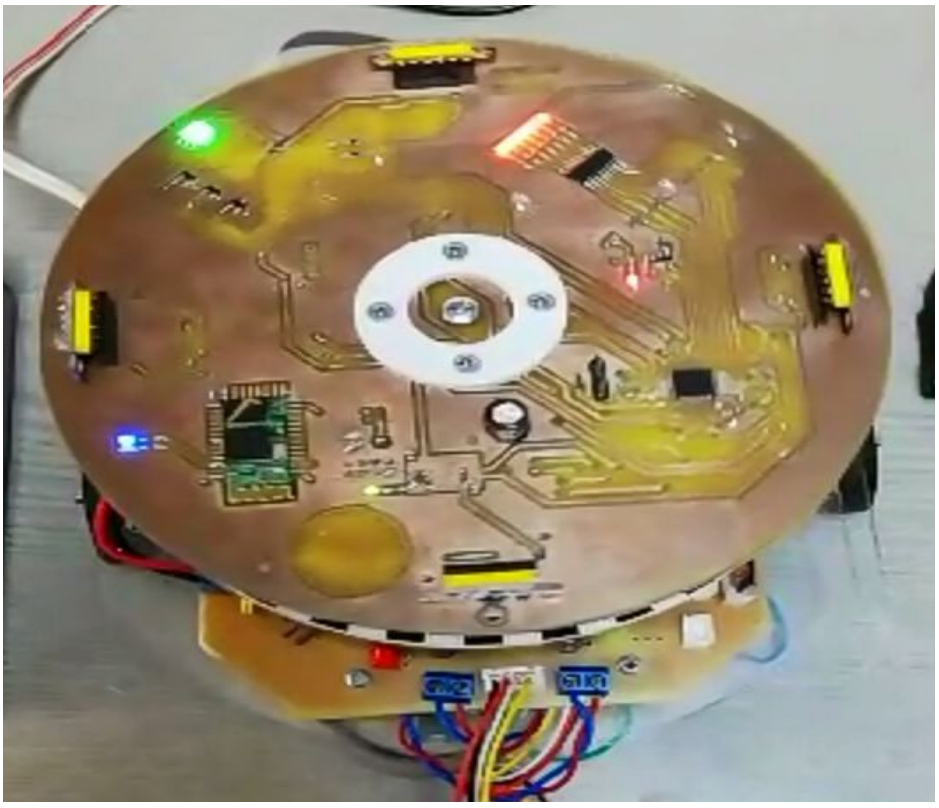
- Turn on the white-black power switch to start the robot.



- Green led on the power switch and leds on the boards must be turn on. If you can not see any led, please check the lipo-battery power cables. It should be connected to power board itself through yellow port.

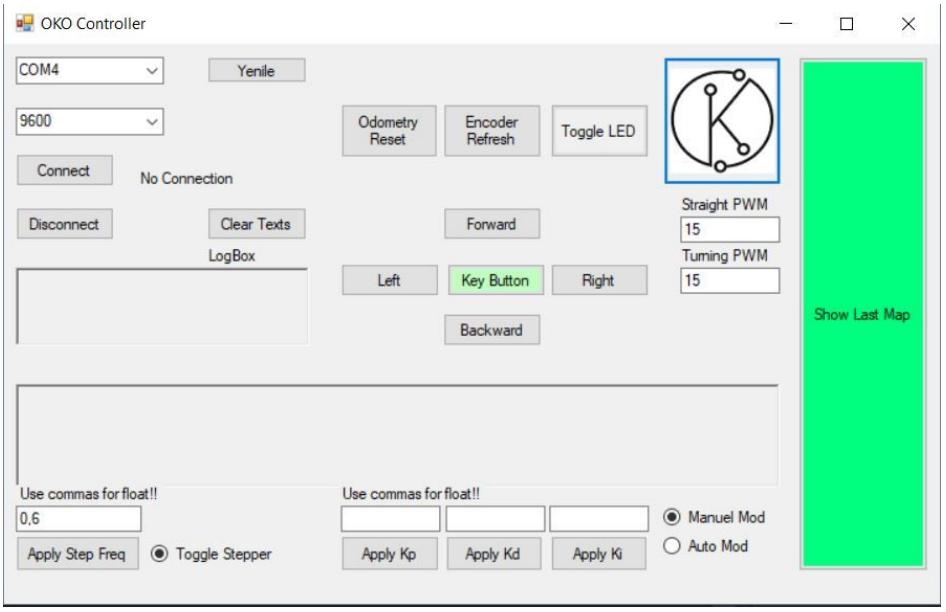


- Now you should be able to see the leds on the robot.

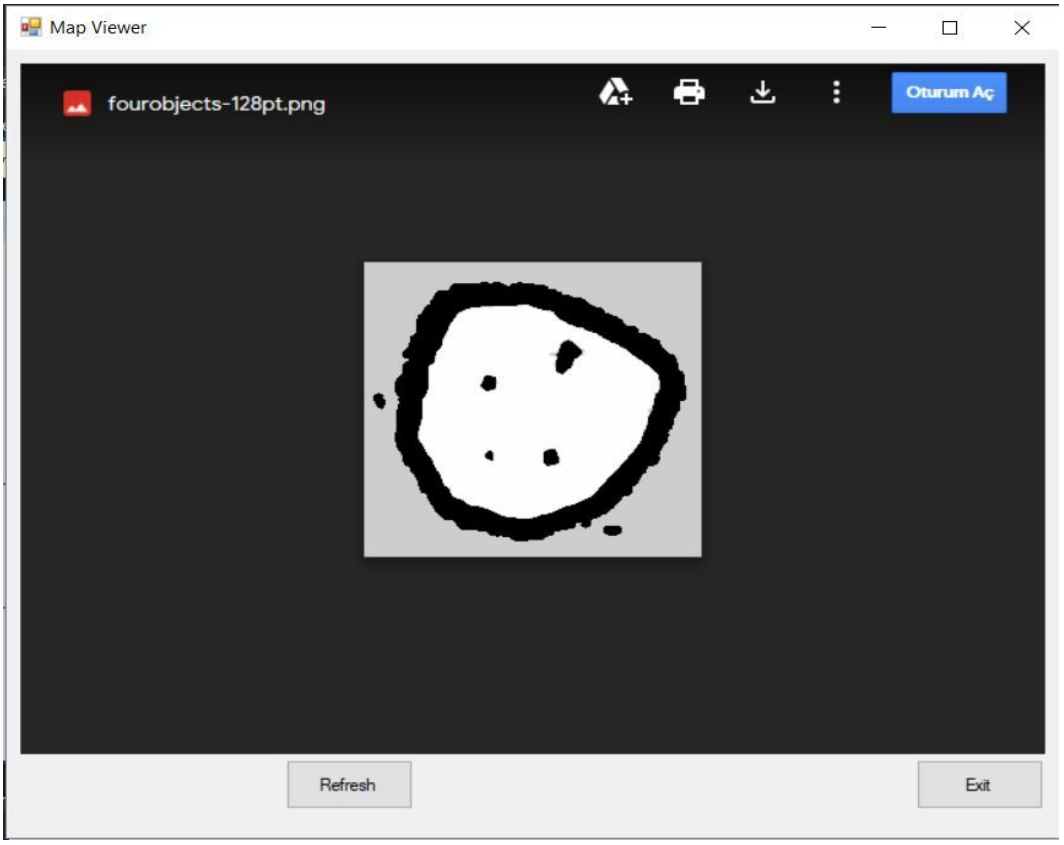


- Put the robot into environment and wait for the extraction of the map.

- Open the Graphical User Interface.




- Click Show Last Map Button to see the current map




For further information about the software and how to manipulate it please see developer manual.