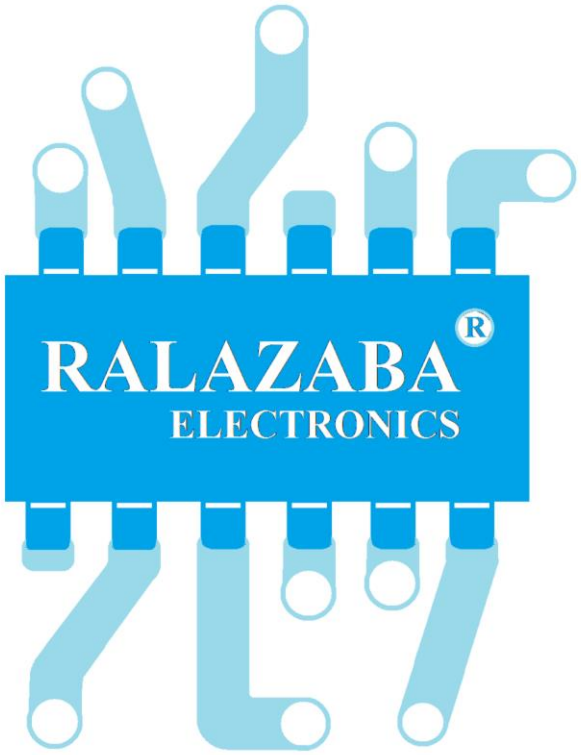


RALAZABA ELECTRONICS

PROJECT AMR

Conceptual Design Report

A device trying to extract the plan of their surroundings

Prepared for

Assoc. Prof. Lale ALATAN

Project Start Date: 30.10.2018
Project Completion Date: 18.05.2019
Project Duration: 7 Month

Table of Contents

Executive Summary

RaLaZaBa Electronics design and construct a device that extract and display the plan of the closed area containing placed object the device competes the other pair in terms of the better plan and operation time. We, as RaLaZaBa Electronics, ensure solving the problem in the most efficient and creative way.

The project can be reviewed in two branches. One of them is 'main constraints and environment specification' and other one is 'solution approaches and algorithms'.

Firstly, the environment is created by 50 cm long wall and objects that are limited to

- Cylinders of 10 cm diameter,
- Cylinders of 5 cm diameter,
- Square prisms with 7 cm edge length,
- Prisms with an equilateral triangular base of 8 cm edge length.

In addition, design must be 'onboard', there is no communication and no external sensors. There are no constraints about camera or sensor usage in design.

Secondly, our solution to design is consist of some scopes. The scopes:

- Decision and Control Unit
- Environment Sensing Unit
- Motion Unit
- Mapping
- Sensing Unit for Self-Localization
- Motion of sensor
- Power Unit

Decision and Control unit is the brain of the device. Timing of operation and decision are determined by this unit. Eye of the device is Environment Sensing unit which provides the recognition of objects and ambient. Foot of the device is motion unit which help the device go from somewhere to somewhere. Heart of the device is Self-Localization unit which sense the its position to reach the different viewpoints. Power unit is an energy source of the all system to make all duty properly. Mouth of the device is Mapping Output unit which tells results when operation is over. The subsystems should be work properly to extract the mapping of environment at better and faster.

Finally, when we merge all of this subsystem, we create a mapping device.

Introduction

In this project, the main objective is to design an autonomous robot which plans a bounded environment containing an unknown number of objects by telling the number of objects, their shapes and centers, produces a map and sends it to the monitor. Moreover, the robot should finish this planning operation in the shortest time and minimum error as possible. Main features of the robot are:

- To move in x y directions a bounded environment by visiting all regions of the environment
- To make self-localization for detecting the current position of the robot
- To sense objects and bounds of the environment
- To get all necessary data from the objects and environment and combine these data and transfer to the data processing unit
- To process data and determine the number of objects, their shapes and their center information

These main features are also subsystems of our project. The detailed solution approach and subsystems and overall system definitions are in the following sections.

Currently, we made some experiments for our subsystems and determined some possible solutions for our subsystems. At the first stage of the project, we created a system for sensing the environment and processing measured data to determine objects features and their locations. Then, we moved our robot without actuators and collected data with different locations. With our algorithms, the robot can combine data come from different observation points to building the overall map. Besides, we developed a shape finder algorithm to determine which shapes are in closed area. Measured data gives 2-dimension (x and y) coordinates of the object and closed region. The measured data is not enough to take a better accuracy. So, the data are enriched and classified with respect to shapes.

Our solution idea starts from collecting data from environment via sensing unit. From first collection, the robot finds the closest object relative to its position and builds a road. After completing the road, the robot takes measurements from the object at different points and labels this object as ‘seen’. Then, again it starts to search for new closest and ‘unseen’ object. After the robot labels all objects and identifies their type and location, the robot transmits the map to the monitor via communication channel. Alternatively, probabilistic road-map algorithms like SLAM and PAM can be also used. Since they are relatively complex and hard to implement real time hardware, we chose this idea as alternative.

The first part of this report, we examine the detailed problem statement, measurable objectives. Then we explained our detailed solution approaches, our subsystem definitions, overall system definition, and experimental results. The final part of the report consists of a detailed breakdown of planned work, team’s work sharing as well as Gantt chart to present project timeline. Moreover, test procedures, measure of success, cost analysis and deliverable also explained.

Problem Statement

The project is required that a robot extracts the features of the closed area and gives the object specification and map of the closed area. In addition, the robot competes with other robots. So, accuracy of plan and operation time is significant.

The problems can be stated as:

- Detection of position and orientation of the surroundings and targets.
- Detection of the position of the vehicle.
- Designing fast, agile and accurate actuator system.
- Optimizing searching algorithm that minimize the time for mapping.

Finally, for the supposed robot can overcome to the problem to reach the goals and missions.

Solution

Overall System

In this project, the robot moves autonomously and draws the map of the closed area with the defined 4 types objects. The project is required that the robot senses the environment and decides for the movement and gives the object type, size and orientation as output. Thus, considering this requirement, the project is separated to 7 subsystems and overall system has a closed loop control to make our systems work properly.

The main solution is starting to ON command via button or starting signal to the robot. Then, the actuators of the sensor start to operate, and first data collected. After data collection operation done, it labels the closest object then move to that object. Then data collection operation starts again iteratively. After all objects in the field are labeled and their data is collected, using it observation (the points where the data collected) locations information the final map built with various data processing algorithms. After building the map, the map transmitted to monitor for display.

Our searching algorithm is not the quickest searching algorithm for our project. There are considerable amount probabilistic road-map algorithms like SLAM and PAM. As an alternative these stochastic models can be an alternative solution for road-map algorithms. Since implementing probabilistic models to hardware is excessively difficult, building more deterministic road-map algorithm is more feasible. Therefore, we build our main solution idea based on a deterministic model.

Overall Flow-chart

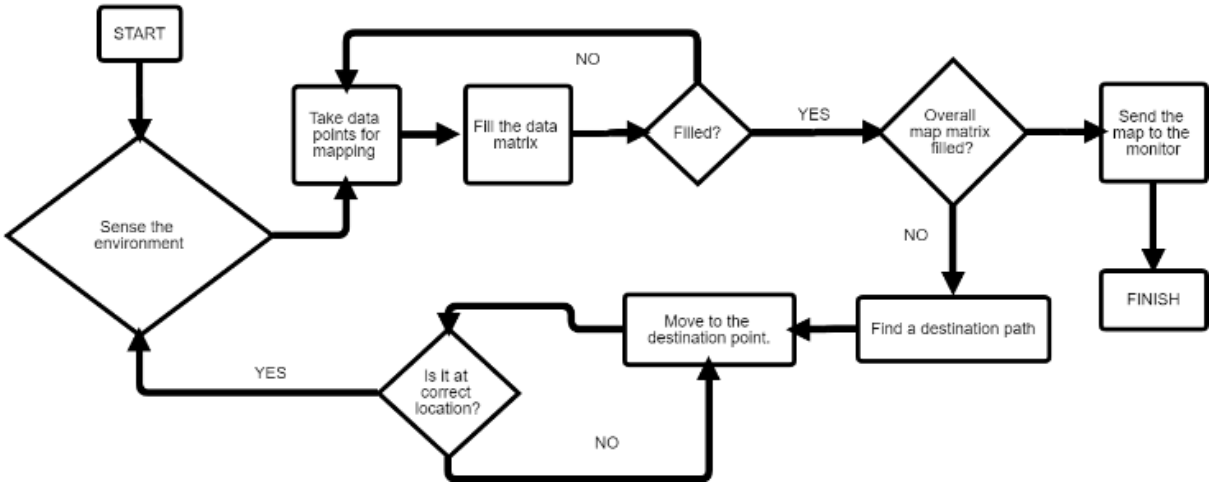


Figure 1 Overall Flow-chart

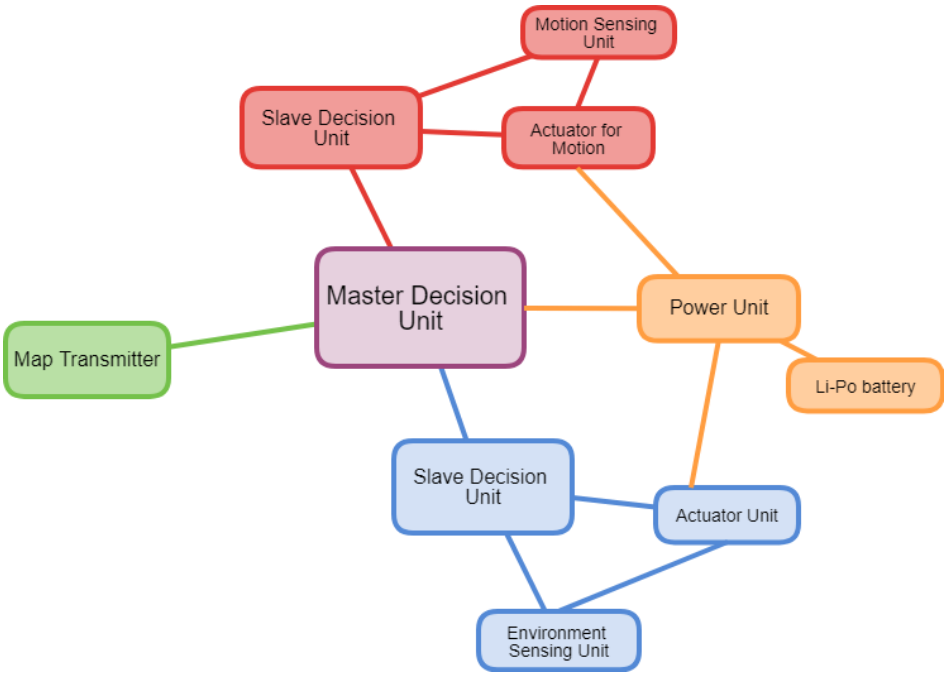


Figure 2 Block Diagram

There are 7 subsystems which are;

- Decision and Control Unit
- Environment Sensing Unit
- Motion Unit
- Mapping
- Sensing Unit for Self-Localization
- Motion of sensor
- Power Unit

Interaction and Interfaces of Subsystems

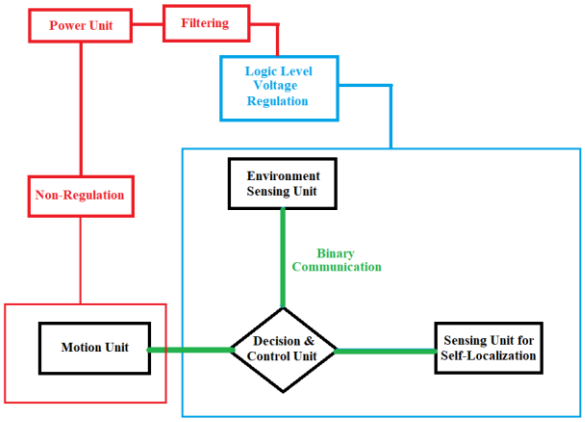


Figure 3 System interactions and interferences

In our system, we need to regulate the voltage level of power unit since most of the microcontrollers and sensors do not tolerate more than 5 volts. This is established via linear or switch mode dc-dc converters. The boundaries in figure 3 represents, voltage levels in the system. Also, we need to isolate the voltage changes due to high-current need of the motion unit. To get rid of the voltage changes, we use a diode and large-value capacitor is used as shown in figure 4.

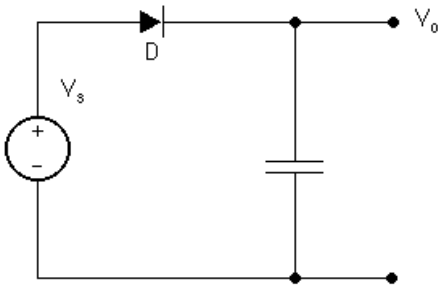


Figure 4 Filtering Circuit

In environment sensing unit, the sensors communicate with the control unit via 5V tolerable I2C or SPI communication protocols.

In sensing unit for self-localization, it is required to continuous tracking of position change. This task is accomplished by using low-level microcontroller. And the microcontroller communicates with Control unit with I2C with fast clock signal.

In decision unit, we use a main microcontroller with high clock frequency and large memory to conveniently process data coming from environment and self-localization units. In civilian IC market, most of high-performance microcontrollers can tolerate 5 volts.

In motion unit, we need lots of power to actuate the whole mass of the device. Since the unit requires higher voltage and current draw, it is connected to the power unit without any regulation. The communication is established by binary signal. Then, the communication should be isolated via logic level converters. They convert 5V-logic signals to lower or higher voltage levels of logic signals.

Subsystem Design

Decision and Control Unit

The unit governs the other subsystems and it makes other subsystems connect each other. Other subsystems send or receive data with the unit. In addition, the control and optimization are made in this manner. Thus, the unit may be divided into two basic parts. One of parts is controlling timing and communication of sensors and movement of actuators. Other one is creating a map and destination point (road map) from taken data.

As seen flow chart Figure 1, data collection from environment sensing and self-localization are not parallel operations. Also, the movement of sensor and robot come true one by one. Therefore, the unit determines the order of operation and commands the operations. Meanwhile, the unit creates map and route. The route is required that movement of devices.

As seen block body diagram Figure 2, the unit connected all other subsystems. The connections are explained as:

- Decision & Control unit with Environment Sensing Unit:

Decision & Control unit give a command that Sensing Unit takes data or not. The taken data are stored and processing for mapping algorithm. The communication between the units come true I^2C protocol. It is explained at Environment Sensing Unit part.

- Decision & Control unit with Actuator for Sensing Unit:

Actuator for Sensing Unit is required for proper data collection in total ambient. It is supplied by motors that rotates the sensing unit. The rotation changes the direction of sensor. Thus, Decision & Control unit commands the motors to rotate specific angles and takes the rotation angle to use in mapping algorithm.

- Decision & Control unit with Actuator for Self-Localization Unit:

After the sensing environment operation, the device can move along specific route that is determined by the decision unit. PID controller are used for motor controlling and reach the destination point properly.

- Decision & Control unit with Self Localization Unit:

Self-localization is used for the creating map from the sensing environment. The sensing environment and self-localization data are merged at decision and control unit. Also, the self-localization data are used to control the robot movement.

Algorithms for Mapping

Decision unit takes data from the sensing units. The data should be converted to map. Before the mapping, the data from environment sensing and self-localization should be merged.

Environment sensing unit gives the distance from sensor direction. The direction of sensor is governed by decision and control unit by actuators. The distance is converted vector space based on device position. The device position is determined by self-localization unit. Thus, the device position and the vector are added. Also, device direction changes after the motion. The direction changes the coordinate system of sensing unit. The changes in coordinate system at different position are illustrated Figure 5. The coordinate system is converted to stationary coordinates by using rotation matrices at Figure 6-7.

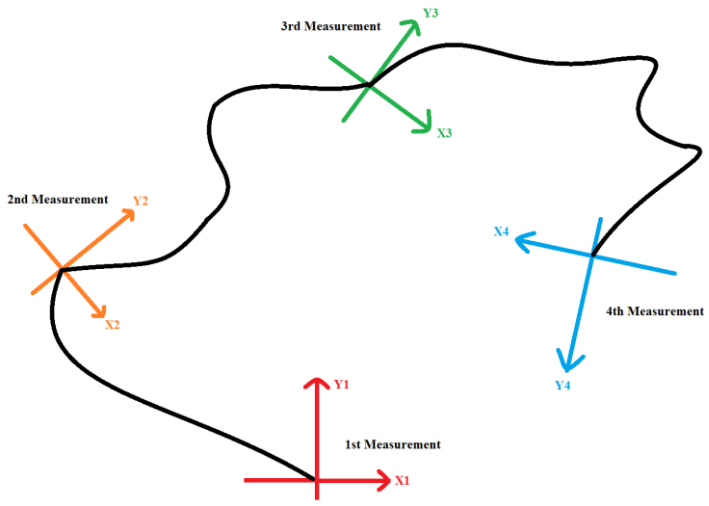


Figure 5 Coordinate axes with different direction of the device

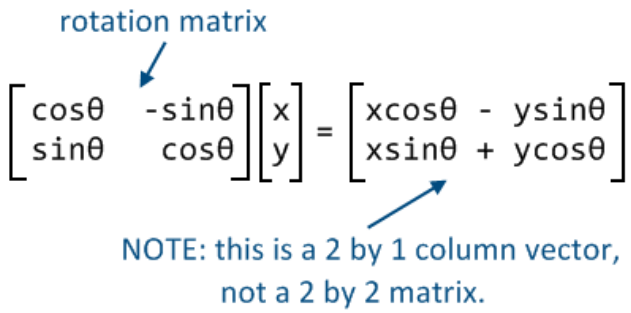


Figure 6 Rotation matrix

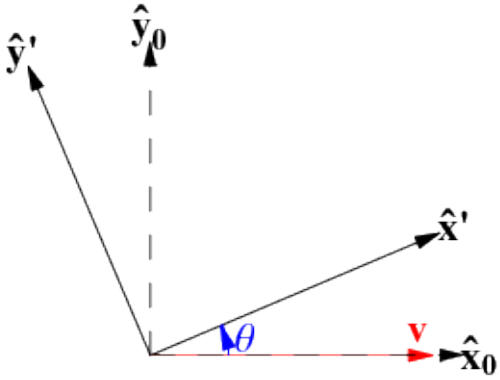


Figure 7 Rotation of coordinate axes

Therefore, the map is created in this manner. The map is binary matrix that shows whether object is there or not. At every iteration of the different location, the binary matrix is updated.

Finally, the binary matrix is converted to image. It is basically a sketch of environment.

Shape Finder

Shape finder algorithm takes the x-y coordinates information from sensing unit. Then, these x and y information is embedded to a 2D array which is also an image of the environment. Our algorithm must make a decision by using this image. From that time, we have a map as image and we use some image processing tools to be able to identify the properties of the objects in the environment. After preprocessing operation, we have an edged image which contains only the necessary data for decision. Using this edged image, we find all contours on the image firstly, then, we masked these contours according to the area of these contours considering the areas of the objects. To specify the area range, we used experimental results and the area information of the object in our map image in terms of pixels. Then, by identifying the number of sides of the contours that we have, we decide the shape of the objects. After identifying the shapes, we used geometry to find the centers of different types of objects.

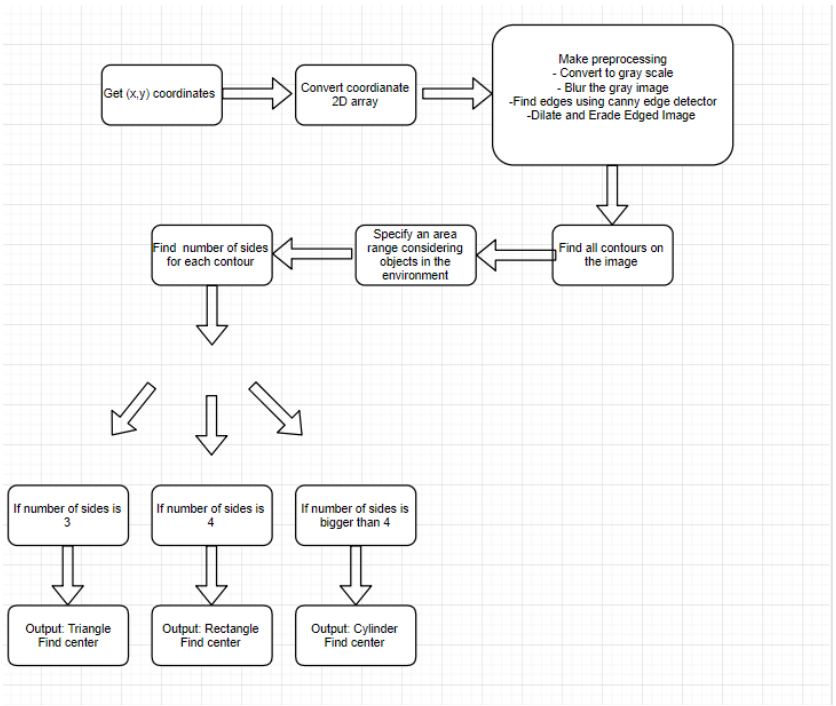


Chart 1 Flow chart of shape finder

Road Map Algorithm

As mentioned, mapping algorithm, the robot should move the different location. This location is determined by the probabilistic algorithms. The algorithms are stated as:

SLAM (Plan A):

Simultaneous Localization and Mapping is abstracted SLAM. SLAM is actually a problem that is stated:

- A map is required for localization
- A localization is required for map

Slam problem can be solved by Extended Kalman Filter. Extended Kalman filter is the nonlinear version of Kalman filter. It linearizes current mean and covariance.

EKF SLAM builds the Map with cycle:

- State Prediction
- Measurement Prediction
- Observation
- Data Association
- Update
- Integration

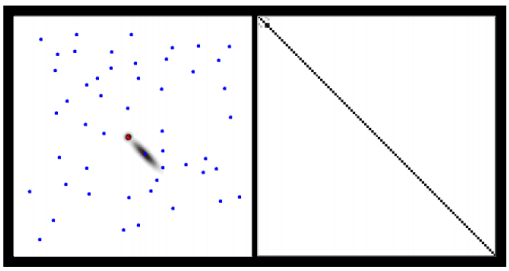


Figure 8 Original map and its correlation matrix

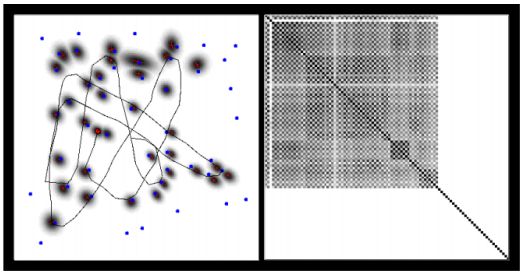


Figure 9 Route map after probabilistic algorithm

Plan(B)

For the simplicity, the devices can follow the wall. Firstly, device goes to wall and stops the specific distance from the wall. In addition, the device stops at every cycle that is determined by the calculation of optimum destination. Figure 10, the probable route is illustrated.

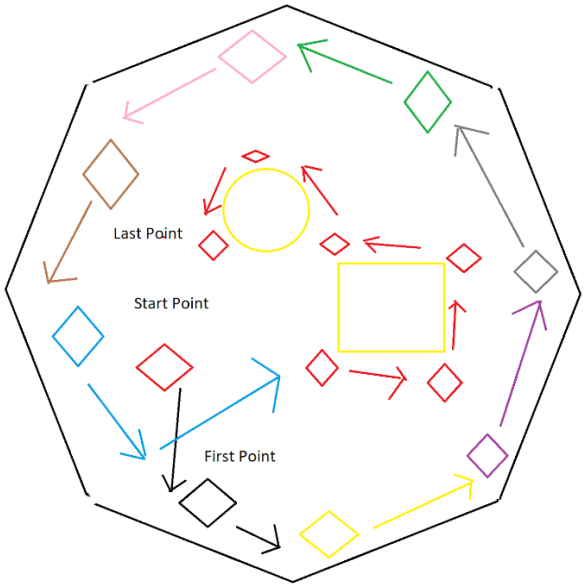


Figure 10 Probable route of the device

Environment Sensing Unit

In this unit, we need to determine the objects that surround the device like figure 11. To determine the positions and types of objects, we need to transfer the field into a binary understanding of the device. To accomplish this task, we need to understand whether there is an object or not. Our first aim is detecting the object. For this purpose, we can use wave since waves can reflect a specific object and, in this project, our equipment can reflect the waves. Now, we know that we can use waves and there are different types of the module which can make this mission. The second approach is recognizing objects. Our eyes can determine the objects easily since we can understand object by looking at their edges and, our eyes can understand the distance of object dimensions. As we can see in figure 12, we can understand the type of objects. Hence, we decide to use a distance sensor as the first solution since distance modules are inexpensive and we can use easily with a microcontroller.

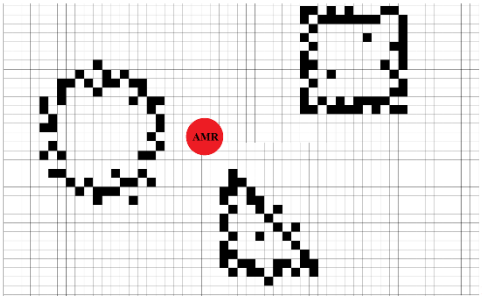


Figure 11 Realization of Sensing Environment

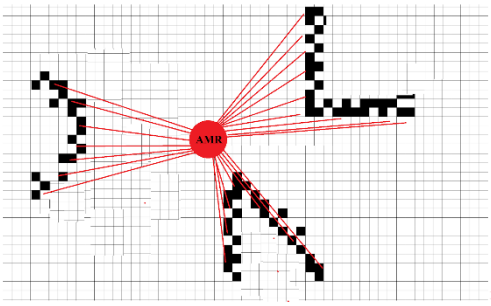
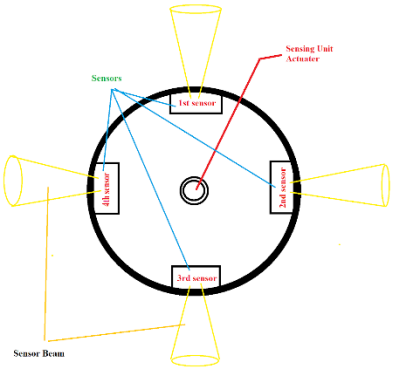


Figure 12 Realization of Sensing Environment



However, there are some problems and one of them is accuracy. To solve this problem, we decide to use more than one module. Therefore, we can use four distance modules like in the figure 13 but that is not enough because beam width is not large enough and we decide to rotate the circular module. Hence, we rotate 90 degrees clockwise or counterclockwise and after completing the tour, we rotate the opposite direction. Therefore, we scan the whole environment of the device two times.

Figure 13 Sensing module

Second approach to this problem is camera. Camera make 3D- 2D transformation. For this transformation, camera takes 3 data which are Height, Width, Object Distance. The three data are converted to 2 dimensions. In addition, the transformation depends on Focal Length of camera, as seen figure 16. Focal length of camera is constant. Thus, the object distance is calculated by basic geometry. In addition, the width of object is determined by using height of object. The formulations are illustrated at Figure 14-15.

Therefore, the sensing environment the distance of the object is required. The distance of objects is calculated by the camera. In addition, the camera directly gives the width of object that make decision unit works easily.

$$(x, y, z) \mapsto \frac{h}{h-z}(x, y).$$

Figure 14 Formula of

$$(x', y') = \left(x\frac{d}{z}, y\frac{d}{z}\right) = \frac{d}{z}(x, y)$$

Figure 15 Formula of

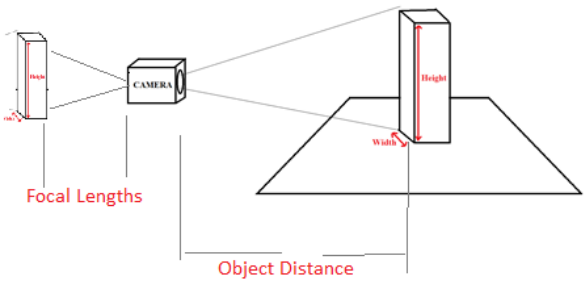


Figure 16 Basic Working Principle of Pinhole Camera

Motion of sensor

We need to sense surrounding the device to extract mapping. To achieve this aim, we need a module that can move 360 degrees. Today, LIDAR has used this type of project, but we have a low-cost objective and, we have a limit on the whole project budget. Since LIDAR is so expensive, we do not want to use it. Due to that, we decided to make a module like LIDAR. Hence, we decided to use step motor to rotate the sensing module since we can find various type of step motor which is fit our cost objective. Since we need to have an accurate result for extracting map, we need a step motor that has a small step angle. Because of that, we can give a lot of point from sensing module in one cycle of rotation. Our rotation plan is that first we rotate sensing unit 90 degree in the clockwise or counterclockwise direction. After completing rotation, we rotate the sensing module in opposite direction.

When step motor does not work properly, we can use a servo motor. In the first plan, we do not want to use servo since the rotation angle of the servo is not constantly divided. For example, when we want to rotate 90 degrees, the servo motor cannot rotate exactly 90 degrees and we can have false labeled data (r is true $\boldsymbol{\theta}$ is wrong).

	Servo motor	**Step motor**
Pros	Small unit volume. Inexpensive. No need for an extra motor driver.	High resolution for the angular position. Less positional error.
Cons	Low angular position resolution. High angular position error.	Costly. The magnetic field generated by step motor can affect the compass or other sensors. Motor driver (DRV8825) needed.

Motion Unit

In our project, we need to extract a map of unknown environment since in environment, number of objects and shape walls are changeable. Due to that, we need to move the device to collect information to extract the map clearly.

To move properly in mapping field, the device should rotate left or right. To achieve this aim, we think two ideas which are omni wheel and differential drive with the freewheel.

A differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion.

Table 1 Rotation direction of motor with different voltage level

Left Motor	Right Motor	Direction of Device
0	0	Stop
0	1	Left - Forward
1	0	Right - Forward
1	1	Forward
-1	-1	Back
-1	0	Right - Back
0	-1	Left - Back

As we can see in the table xx, by using differential drive method, the device can move all direction by giving proper voltage level. Also, as we can see in the picture xx, we need one free turning wheel. However, we do not use any other wheel since as wheel, we use trackball mouse that uses for sensing unit.

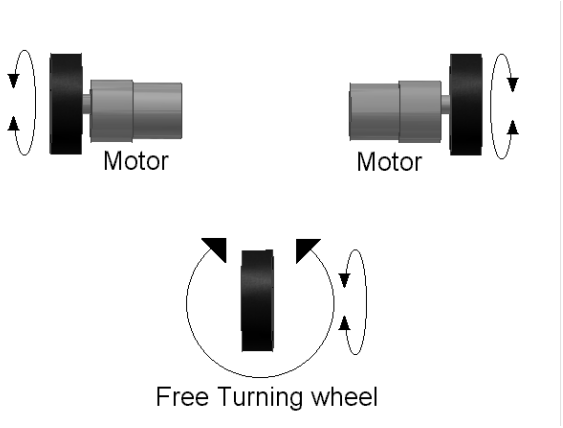


Figure 17 Differential motor drive

However, we have a problem and that problem is motor`s response. When we apply some voltage, motor moves but not accurate since when giving high voltage level to motor, motor move suddenly and the device shakes. Therefore, in order to avoid this shaking, we decide to use PID controller. A PID controller continuously calculates an error value e(t) as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name. In practical terms, it automatically applies accurate and responsive correction to a control function. Thanks to PID controller, the device does not shake.

When we face a problem with differential drive technique, we use omni wheel method or mecanum wheel. A robot that uses omni-wheels can move in any direction, at any angle, without rotating beforehand. The omni-wheel is not actually just a single wheel, but many wheels in one. There is the big core wheel, and along the peripheral there are many additional small wheels that have the axis perpendicular to the axis of the core wheel. The core wheel can rotate around its axis like any normal wheel, but now that there are additional wheels perpendicular to it, the core wheel can also move parallel to its own axis. We can see shapes of omni wheel in the figure 18.

Figure 18 Omni Wheel

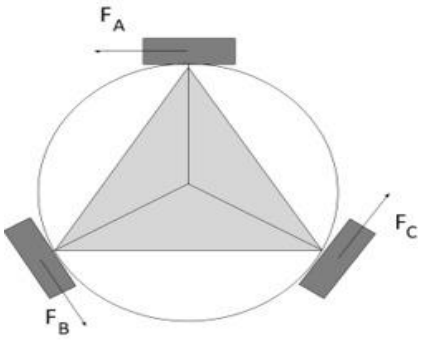


Figure 19 Vector Diagram

As the first solution, we do not want this driving type since one of the main challenges for using omni wheel is controlling the motion so that the robot is able to go to a desired position with a desired orientation compared to differential drive. Also, we need to use at least three motor for this configuration to move any direction. We can think motor direction as vectors like in the figure 19 and we need three vectors to scan all plane.

However, we can eliminate one of the mousses since we move the device on xy plane, in other words, the device moves forward- backward and right-left but we do not change heading direction during the motion. Therefore, we do not need any rotation angle for self-localization unit.

	Omni wheel	**Differential drive with freewheel**
Pros	All package omni wheel (motors drivers and chassis and encoders) are available. Flexible position control	Easy to implement. 2 DC motor and 1 L298(dual) are sufficient. Inexpensive
Cons	2+ DC motor and driver needed. Quite expensive	Mechanical design should be done by us

Mapping Unit

In this project, we need to show map, type of objects and how many objects there are as output. We decide to use computer screen to achieve this aim. We determine two concepts which are sending information real time and after completing whole process.

As the first solution, the device sends information after completing whole process since sending real time information is extra work load for processor. In other words, the device collects only mapping

information to extract map clearly. After completing mapping information, it can send information by using two methods. One of these methods is Wi-Fi. It is the cheapest and easiest method since if we use raspberry pi, raspberry has Wi-Fi module. Thanks to that, we do not any external module and to send picture and text file which has information about how many objects there are in the field, their dimensions and coordinate of center. Second approach is that we can use Bluetooth module to send pictures and text file and thank to raspberry pi, it has Bluetooth module. Hence, we can connect a laptop to send information. However, if we do not use raspberry pi, we need Wi-Fi or Bluetooth module.

As the second solution, we can send information real time. In this configuration, the device both takes data around and sends mapping information to a computer. This requires large parallel processing and also, we need to design a GUI to follow the creating map. We do not want to divide processing of microprocessor since module can lose information.

Sensing Unit for Self-Localization

Our robot should move autonomously since it is a project requirement. The motion of autonomous devices is governed by the algorithms that mostly rely on probabilistic models such as SLAM, PAM. Since these algorithms are strongly dependent on the environment information and self-localization, both motion and environment sensing have critical significance. At this part of the report, we introduce solution ideas for sensing for the motion of the robot, which we call self-localization.

Self-localization means that the device can measure how much move away start point or former point which device is there so that it can know its location relative to the environment at all operation time. It is also called dead reckoning in literature. Solutions of the self-localization problem are argued at the subsystem. There are our five solution methods.

- 2 mousses (Plan A)
- 1 mouse + compass (Plan B)
- 1 mouse + IMU (Plan C)
- Encoder + IMU (Plan D)
- 1 mouse + Omni drive (Plan E)

2 mousses (Plan A)

PC mousses have capabilities of tracking position inherently. However, they can only give information about linear motion on the plane, not angular motion. To overcome this problem, we used 2 mousses side by side to detect angular motion using their relative motion. Mathematical derivations and related know-how can be found at [1]

Features

- Precise optical navigation technology
- Small form factor (10 mm x 12.5 mm footprint)
- No mechanical moving parts
- Complete 2D motion sensor
- Common interface for general purpose controller
- Smooth surface navigation
- Programmable frame speed up to 3000 frames per sec (fps)
- Accurate motion up to 12 ips
- 400 cpi resolution
- High reliability
- High speed motion detector
- Wave solderable
- Single 5.0 volt power supply
- Conforms to USB suspend mode specifications
- Power conservation mode during times of no movement
- Serial port registers
  - Programming
  - Data transfer
- 8-pin staggered dual inline package (DIP)

Figure 20 Key features of a typical mouse

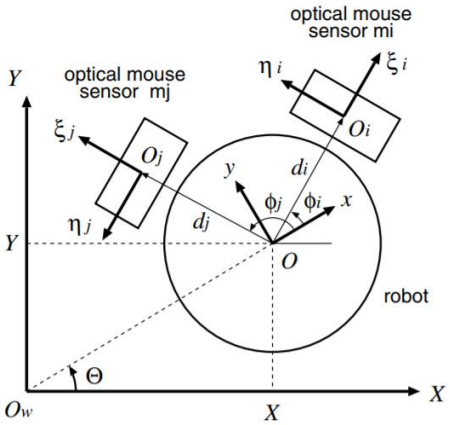


Figure 21 An example configuration of two mousses

The mouse has better resolution (about 63.5 µm) relative to encoder option [2]. Related measurements and test plants can be found figure 20 and 21. Optical mousses cannot work every plane due to their nature of electromagnetic wave. Therefore, we test them both in the design studio and KKM hall. Although it worked floor of the design studio, it didn't work at KKM hall's floor. For that reason, we changed our mouse selection from optical to tracked-ball.

1 mouse + compass (Plan B)

Instead of two mousses, we can use a mouse and a compass. Compass gives three output these x, y, and z coordinate with respect to the magnetic field of the earth. The reason that compass is used in such configuration (with a mouse) is compass gives as rotational information that (direction in XY plane) one mouse cannot give. With directional information, it compensates necessity of the second mouse.

This method like two mousses usage since in this method, with respect to x, y and z values, we calculate rotation angle. One can think that the data comes from the mouse is r, and compass data is $\boldsymbol{\theta}$. Changing cylindrical coordinates (r, $\boldsymbol{\theta}$) to cartesian coordinates (x,y) the position of the robot is easily can tracked.

1 mouse + IMU (Plan C)

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. We can use this sensor to perceive rotation device since there will be acceleration when the device rotates. We can use to determine the rotation angle. After finding the rotation angle, like the above case, we determine the device position.

1 mouse + Omni wheel drive (Plan D)

With Omni wheel drive, we can control the robot at 2 dimensions without changing robot heading direction. This extra feature gives us the ability that self-localization with one mouse.

Encoder + IMU (Plan E)

Encoders can be used as position tracker of movement actuators, in this case, DC motors. There are different types of encoders that used for DC motors rotary encoders, optical encoders, and magnetic encoders. We will use incremental encoders since our scope is dead reckoning.

The advantage of encoders is they can easily connect the second shaft of the DC motor. Moreover, their relative size is smaller (considering the mouse option). These make encoder option is the *most compact* and *minimum volume* solution. The drawback of the encoders is about reliability. With poorly control or sliminess of the ground, the motor can slip meaning that motor is not actually moved. However, since motors are still rotating the encoder data is not steady meaning that motor is moved. This conflict results in a wrong map because the robot lost its location on the map. Therefore, the position is data is *less reliable* considering mouse option. To overcome this problem either we can drive the actuators slow acceleration (soft drive) with effective controllers or use a second measurement to confirm position data, or filtering position data with Kalman filters.

- Soft drive

With only one measurement device (only encoder) the actuators controlled smoothly and the controller should have high disturbance rejection. However, the smooth drive can be slow which will affect the *total time* for mapping operation.

- Kalman filter

After considering two options we decide to use two measurement devices, encoder, and IMU considering *operation time* and *self-position accuracy*. Drawbacks are the cost of the subunit and its complexity is increased.

Different drive techniques require different encoder configurations and data processes. In the case of the differential drive, 2 DC motors and 2 encoders are needed. With two channel encoder data, we can catch both rotational and linear motion information. However, both encoder configuration and the data processing unit is completely different with omni wheel drive which will be discussed next sections.

	2 mousses	**1 mouse + compass**	**1 mouse + IMU**	**Encoder + IMU**	**1 mouse + Omni wheel**
Pros	High resolution	Mid-high resolution	Mid-high resolution	The overall volume of the subsystem is lower relative to mouse options. It can work almost every flat ground.	High resolution No need for extra data processing*
Cons	Using two mousses can increase overall dimensions Optical mouse cannot work every flow.	Changing magnetic field can affect compass operation.	Optical mouse cannot work every flow.	Aggressive motor movements can create slip, therefore encoder gives false position data. Extra engineering time is required for Kalman filtering Expensive	Requires Omni Drive Expensive

Table 2 Pros and Cons of 5 option

Power Unit

This is most important unit since if energy does not exist, the device does not move or process anything. In our project, motors draw more current with respect to other component and we use at least three motors which are two 12V DC motor to move device and a step motor to rotate sensing unit. If all of the motor work at the same time, they draw almost 4A current but we do not work together since when the device moves. step do not rotate. Also, we have other components in the device and they can draw almost 1A current. To not have a problem, we need a battery that provides at least 12V - 4A continuously.

Conceptual Design Tests & Results

To test our conceptual design ideas, we created two test setups. In the first setup, we created a virtual map to determine whether our algorithm works properly to find the shape and position of objects. The maps that we tested are shown in from Figure 22 to Figure 23. In the first map, we inserted a circle sitting at (241,181), square (665,172) and rectangular at (625, 509) coordinates of pixels.

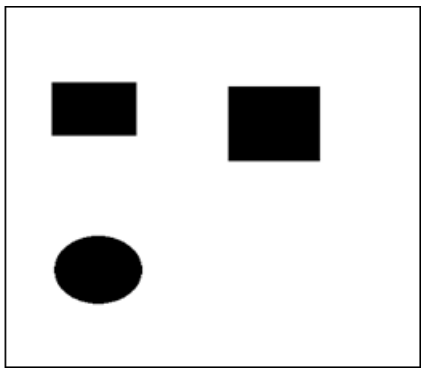


Figure 22 First Test Map

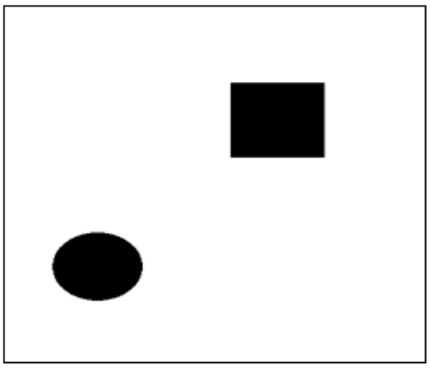


Figure 23 Second Test Map

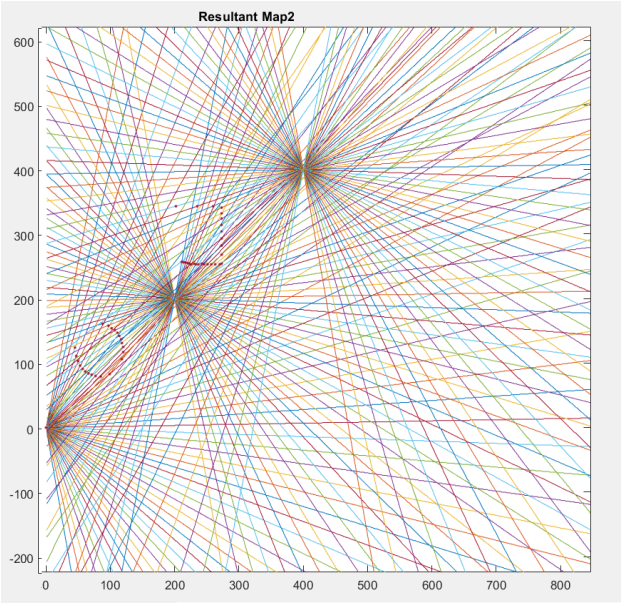


Figure 24 The simulation environment, dimensions in mm.

We created a simulation environment such that, we loaded these maps to a MATLAB script and we selected 3 points for sensing these shapes. From these 3 points, we created lines with different slopes and the intersection points represented the lidar scans with 4.5 degree apart. These lines will represent how the map will be seen by the lidar sensor. Our simulation environment is seen like in Figure 24. In the figure 24, the lines (lidar measurements) is intercepted with the boundaries of the objects. The interceptions are shown in the figure as red dots. The intersection points for the test maps are shown in from Figure 25-26.

After intersecting lines and our objects, some parts of objects are not shown which will be the case in real data. The missing parts of the objects can be seen in resultant maps. To come up with this problem first, we combined x and y coordinates of resultant map data and converted these data to an image to be able to process and create objects.

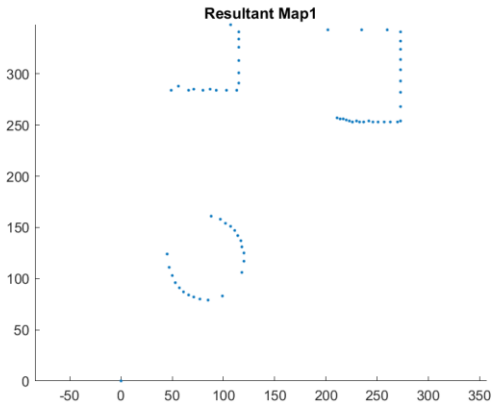


Figure 25 Lidar Scan Result for First Map

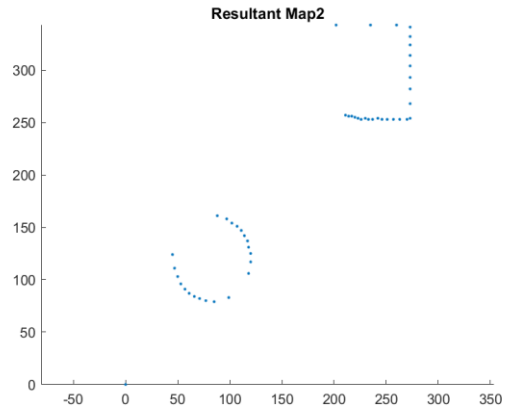


Figure 26 Lidar Scan Result for Second Map

Thus, after determining the simulated data of mapping, we fed it up to our algorithm to determine the boundaries and edges of the objects using Multiple Kernels. After that, we found contours in the image and using some constraints such as the area of contours and the length of contour arrays, the algorithm gives us the shape and center of the objects with no error as shown in figure 27 to figure 28.

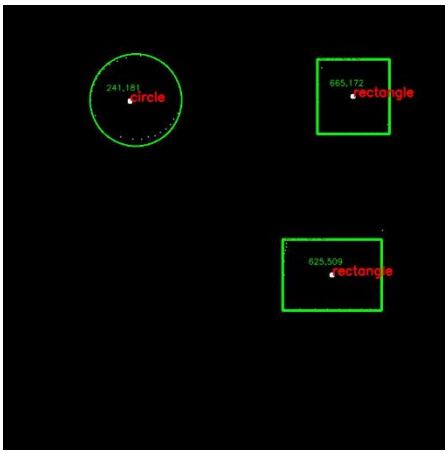


Figure 27 Resultant shapes and centers for First Map

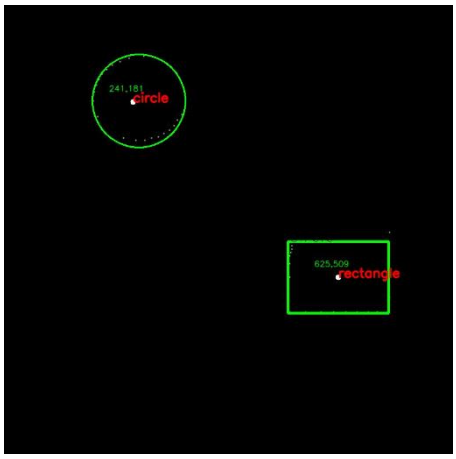


Figure 28 Resultant shapes and centers for Second Map

After confirming our conceptual algorithm is working flawlessly, we took some measurements via VL53L0X on a test setup as shown in Figure 29. In this setup, a servo used to rotate the distance sensor from -60 degree to 60 degree normal to the front surface of the servo. An Arduino Nano is used to transfer the measurements to MATLAB environment. After storage and manipulation of measurements, the data is fed to the algorithm. To have a better understanding, let us make clear definition of the sensor.

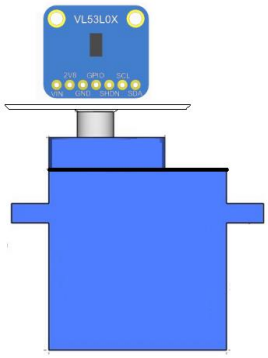


Figure 29 The test setup using VL53L0X and servo.

STMicroelectronics has various distance measurement sensor that uses time-of-flight. The device measures the distance by using infrared laser pulse. The sensor emits infrared laser pulse and counts the time of flight of infrared light beam to return its avalanche photodiode. The sensor has its circuitry that takes the measurement into meaningful figures. The sensor can communicate with microcontroller via I2C protocol. The sensor is 5V tolerable, enabling us to drive it by microcontroller without any logic level conversion. The device emits beam within the expansion cone of 35 degree. This leads us to use the sensor in close ranges, while the sensor can measure distances up to 2 meters. The sensor takes its measurements and sends distance information within 66 milliseconds. It takes measurements with 1 mm resolution, then resolution of each pixel of map array has 1mm edge length.

In measurement setup we observe a major error due to beam width of the sensors. As can be seen in figure 30, the measurements taken for cylinder and triangular prism are very similar.

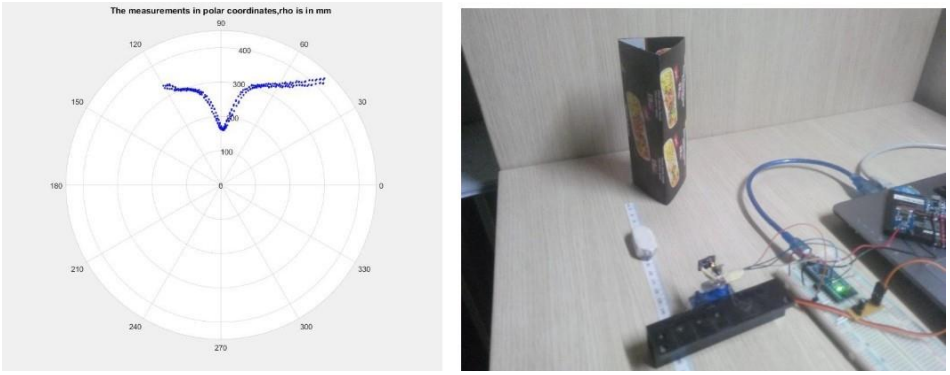


Figure 30 Measurement results of triangular prism

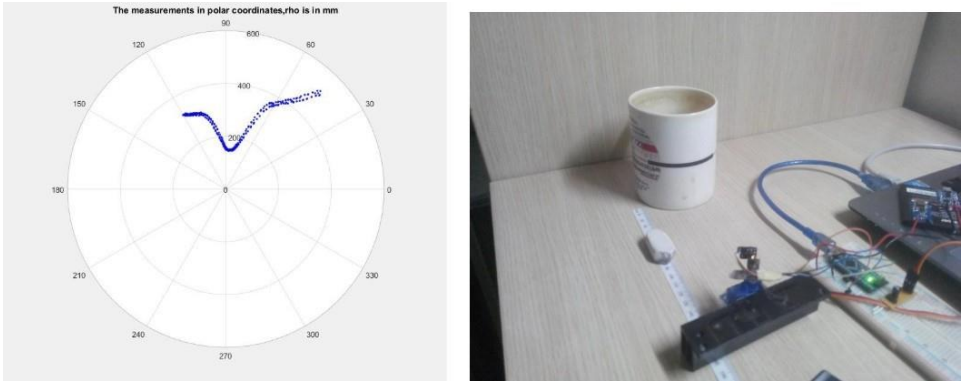
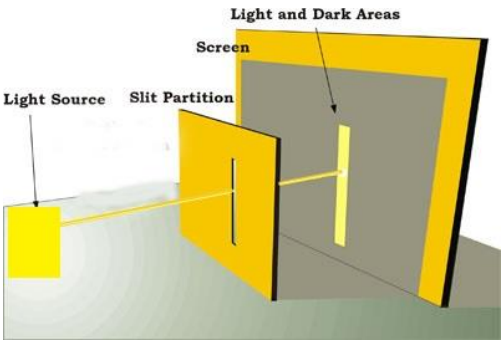


Figure 31 Measurement results of cylindrical prism



This problem can be solved by restricting beam width of the infrared light in horizontal axis. We used a vertical slit front of the transmitter to squeeze the light in horizontal axis as shown in figure 32. With this improvement, we obtained sharp measurements as shown in figure 30

Figure 32 Restricting beam width

We set the measurement environment to test our mapping algorithm with real data. The setup, Figure 33-35, has 3 different viewpoints with respect to object. The test objects are scanned via the lidar sensor from three different points. These measurements are merged by using an algorithm that takes a position of sensor (in 2 Dimension) and angle between stationary coordinate system and sensor data. Normally, the lidar will measure the distance with related angle. This angle information is in the frame of the lidar. We need to project the measurements to the global frame of the map. To accomplish this task, the measurements are rotated using rotational transformation matrix and positioned according to the position vector of the lidar. The output of algorithm is illustrated in Figure 33 to 35.

Later, the obtained data is fed to the object finder algorithm.

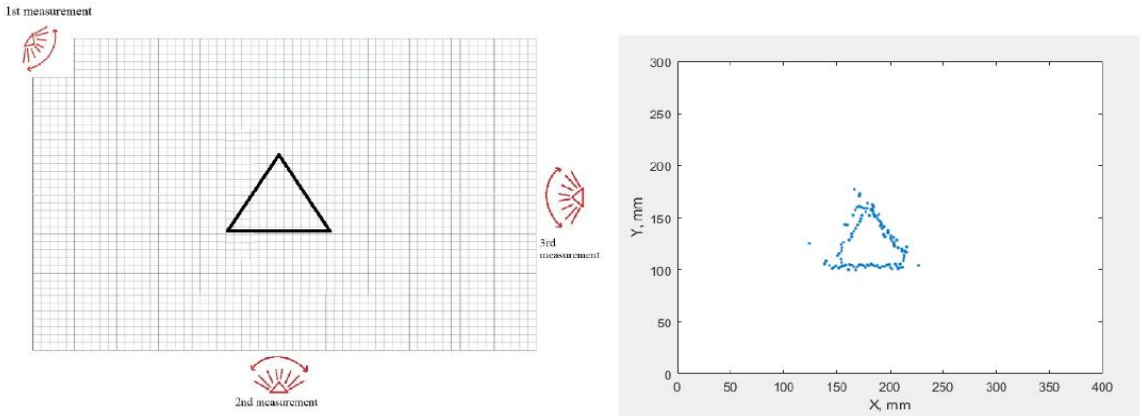


Figure 33 First Test Map and its output

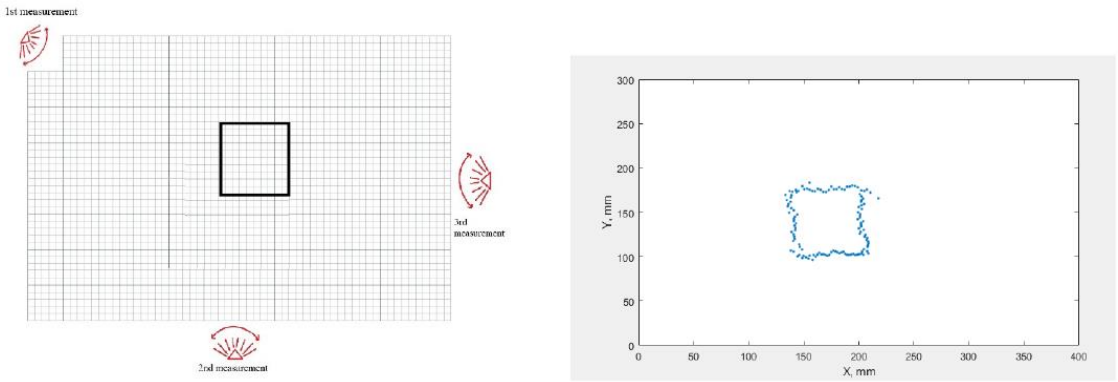


Figure 34 Second Test Map and its output

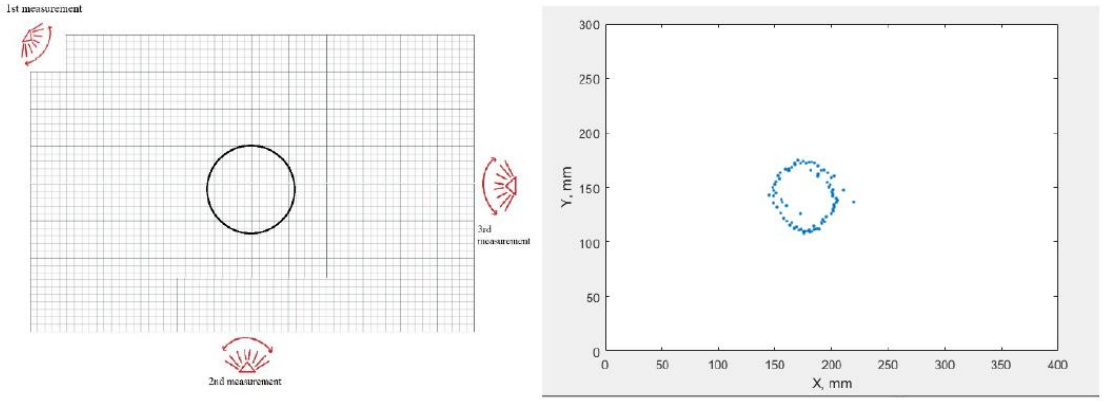


Figure 35 Third Test Map and its output

When we tested our shape finder algorithm with real erroneous data, we had some problem. Most important problem is that the spaces between measured points are large and this causes that our algorithm can find wrong contours. To eliminate this, we added new points to resultant array by adding neighbor points of real data clouds. After that, our thresholding operation worked, and we were able to find true contours. The output for rectangular object is shown in Figure 36. We found the center of rectangle as 146,135. The real center was 146,140. This means that we had 3% error for y coordinate of center. The output for cylindrical object is shown in Figure 37. For cylindrical object we found the center as 154,135. Real center was 146,140. Thus, we had 6% error for x coordinate and 3% for y coordinate. The output for triangular object is shown in Figure 38. Here, we found the center as 148,146. Real center was 184,130. We had 24% error for x coordinate and 12% error for y coordinate.

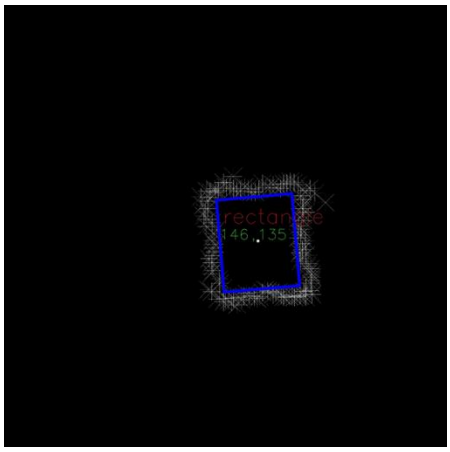


Figure 36 Output of rectangular object

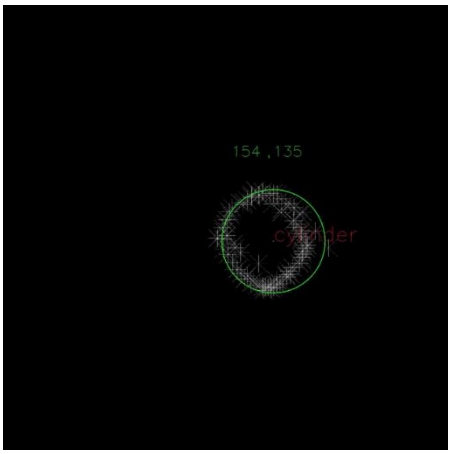


Figure 37 Output for cylindrical object

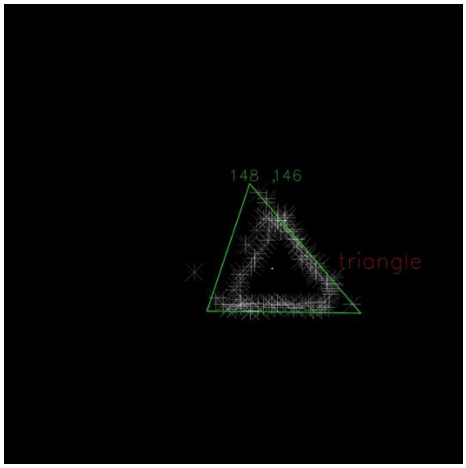


Figure 38 Output of triangular object

Plans

Cost analysis

Table 3 Cost Table

2 DC Motor	20$
1 Stepper Motor	10$
2 Mousses	2$
4 Distance Module	10$
Battery	10$
2 Microcontroller	5$
1 Microprocessor	30$
chassis	30$

We write cost of some important components. We can use other small components. Cost of this project will be almost 150$.

Foreseeable difficulties

Error Sources and Possible Solutions

In our design, we might face with some errors due to imperfectness of real-world realization of our ideas. The possible error sources are accumulated in the subsystems that take measurements of the environment and perform sensitive physical motion. In this sense, the error can mainly occur in unit for self-localization, environment sensing unit.

- Environment Unit Errors

Ideally step motor actuate with fixed step size. However, in real life due to nonlinearities step size can change. Especially at turning points (changing rotation direction) the error is maximum due to inertia of the unit. To prevent this effect, we build mechanical obstacle with every 90 degree so that even though step motor saturates (want to turn >90 degree) it regulates itself mechanically.

Another error source is measurement sensor reading errors. Our solution method is modelling the noise gaussian, Laplacian or uniform noise and normalize the measurement with these models.

- Self-localization Errors

Depending on the solutions we used for self-localization we will face different types of errors. Since self-localization is a critical part for accuracy of the map, error handling is mandatory. The robot is always tracking its location information via motion sensing method. We will use secondary information from environment sensor to handling errors. The labelled objects locations are used for feedback.

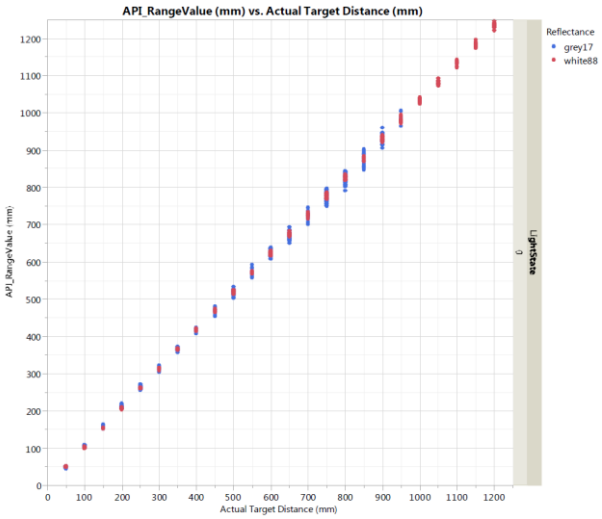


Figure 39 Error line of distance module

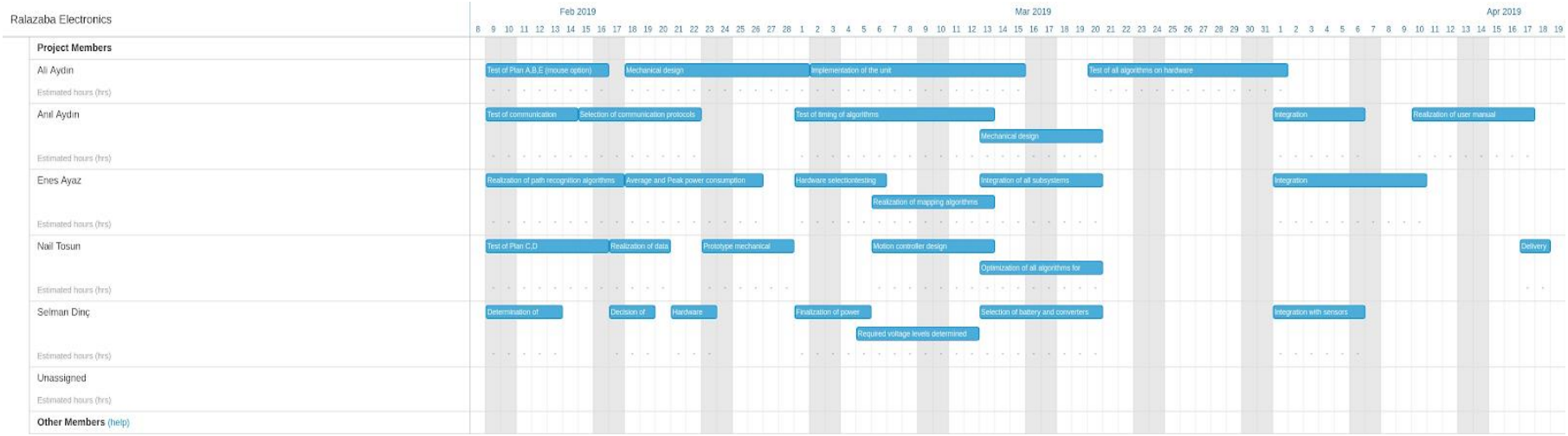


Figure 41 Workload of team members

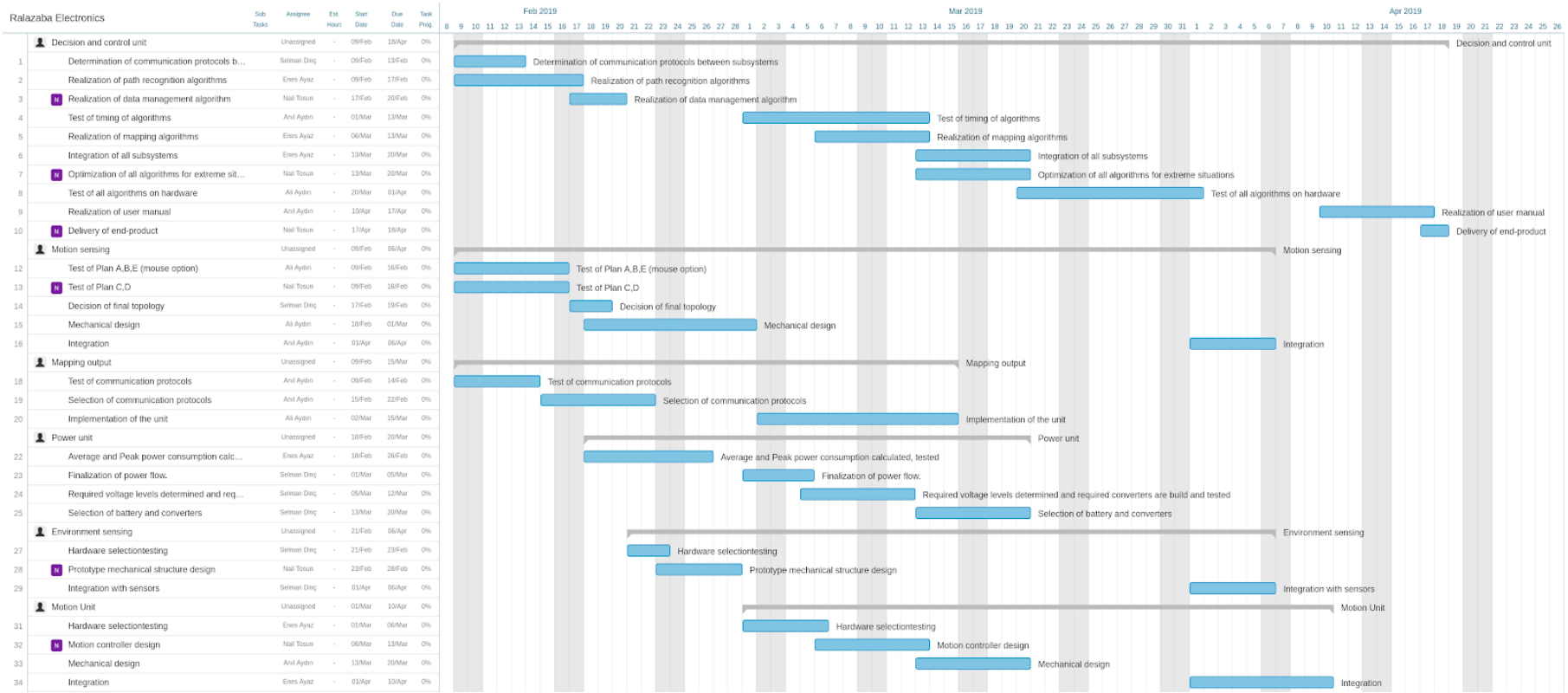


Figure 40 Gannt Chart

Deliverables

Documents

Manual: Our company give a user manual for users. Manual contains information about hardware, software components and describes usage of the device.

Technical Support

Our company gives 24/7 technical support by Selman Dinç. The support is accepted via e-mail communication.

Equipment

Device: Robot is provided for users

Platform: 6 walls and 4 objects will be provided for users

Battery and Charging Unit: We will provide chargeable battery to run the robot.

Software

Interface: We will provide a software package which contains a user interface to see a readable map from a screen.

Conclusion

Our solution idea starts with data collection from the environment. From the first collection, the robot finds the nearest object according to its location and constructs a path. After completion of the path, the robot takes measurements at different points from the object and labels it as 'seen'. Then it starts searching for the newest 'unseen' object. Once the robot has tagged all objects and identified their types and locations, the robot transmits the map to the monitor over a communication channel. Alternatively, probabilistic route map algorithms such as SLAM and PAM can be used. We chose this idea as an alternative because they were relatively complex and real-time hard to implement.

In the first part of this report, we examine detailed problem reporting, measurable objectives. Then we explained our detailed solution approaches, sub-system definitions, general system definition and experimental results. The last part of the report consists of a detailed breakdown of the planned work, the work sharing of the team as well as the Gantt chart to present the project timeline. Test procedures, measurement of success, cost analysis and feasibility are also explained.

Disclaimer

We, as RaLaZaBa Electronics, guarantee that we will provide you to best product according to standards.

Hardware Engineer	Software Engineer	Software Engineer	System Engineer	Project Manager
Ali AYDIN	Anıl AYDIN	Enes AYAZ	Nail TOSUN	Selman DİNÇ

References

[1] Dead-Reckoning for Mobile Robots Using Multiple Optical Mouse Sensors

[2] https://alumni.media.mit.edu/~mellis/mouse/ADNS2620.pdf