“LUCIAN BLAGA” UNIVERSITY OF SIBIU
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER SCIENCE, ELECTRICAL AND ELECTRONICS ENGINEERING

DISSERTATION

SCIENTIFIC ADVISOR: Prof. dr. Ing. Remus BRAD

GRADUATE:
Carmen POPA
Embedded Systems

- Sibiu, 2020 –

“LUCIAN BLAGA” UNIVERSITY OF SIBIU
ENGINEERING FACULTY

DEPARTMENT OF COMPUTER SCIENCE, ELECTRICAL

AND ELECTRONICS ENGINEERING

Localization and mapping with autonomous robot, based on LIDAR sensor

SCIENTIFIC ADVISOR: Prof. dr. Ing. Remus BRAD

GRADUATE:

Carmen POPA

Embedded Systems

"LUCIAN BLAGA" UNIVERSITY OF SIBIU
FACULTY OF ENGINEERING
Department of Computer Science, Electrical
Electrical and Electronics Engineering

APPROVED (date)
15.07.2020
Head of Computer Science, and Electronics Engineering

ES

THEMATIC PLAN FOR DISSERTATION

Student's name and surname *Popa Carmen*.
Study program *Embedded Systems*.
1. Subject ***Localization and mapping with autonomous robot, based on lidar sensor.***
2. Deadline ***01.07.2020***
3. Basic elements for the project
Science articles, books;
Development tool: Arduino IDE and MATLAB.
Hardware: Arduino, chassis (wheels, motors, motor driver), lidar sensor, wireless connector
4. Problems to be solved
Creating a robot that will be able to construct and use a map of the environment, based on the lidar sensor, and localize itself within it. The map will be in 2D and 3D formats.
Based on this map, the robot will automatically detect the obstacle in front of it and avoid it by turning itself in another direction.
6. Advices schedule (every week)
Twice a week.
7. Theme released on 10.03.2020.

Theme received by student:
Date 10.03.2020
Student's signature _____

SCIENTIFIC ADVISOR,
_____
(name and signature)

ABSTRACT

Robot technologies are used more and more every day in many applications, from automotive to home appliance and military. But in order to use them, they must have some information about the environment, like their position in the environment, about the obstacles, about the condition of the environment and so on.

This information can be given to the robot by humans, or the robot can read its information, based on the type of sensor it has. Now there are plenty of types of sensors that can be used by a robot, but the sensor used in this paper will be a Lidar sensor. Using this type of sensor, a robot can create a map of the environment, and therefore use it for determining the environment characteristics.

Therefore, this paper aims to take the idea of a robot in an unknown environment and create the appropriate algorithm based on a Lidar sensor to create the map and localize itself in it. The map format will be displayed in the 2D and 3D scenarios.

Having in mind the complexity of the approached subject, the suggestion would be to split the paper into more chapters, to cover both the theoretical and also the practical part.

Chapter 1 "*Introduction in mapping and localization*" presents a short description for this domain like theoretic elements, elements that represent the fundaments of this paper.

Chapter 2 "*Science accomplishments using lidar sensor*" contains three examples of projects, which present the same idea of using a Lidar sensor to create a map.

Chapter 3 "*Application"* consists of the techniques and ways in which the practical application has been approached. Thereby, the application is based on a robot, Lidar sensor, Bluetooth module, Arduino Mega, and Matlab application.

Chapter 4 "*Experimental results"* present algorithm results, for the 2D and 3D maps and a comparison between the generated map and the map with the real measurements.

The last part of the paper contains the conclusion of the paper as well as the possible future direction that it could follow.

CONTENT

1 INTRODUCTION IN MAPPING AND LOCALIZATION

1.1 Mapping

Robotic mapping is a field still under development and it is combining computer vision field with cartography. Mapping robots can now be found in several areas like industry, military, home appliances, exploration, and self-driving cars [6]. In most cases, these robots are used where humans cannot reach, or it is very hard for a human to reach those areas. The goal of this robot is the creation and usage of a new map inside an unknown environment (could be indoor or outdoor) and also the possibility to localize its own position relative to this map [8].

"*Robotic mapping is that branch which deals with the study and application of the ability to localize itself in a map/plan and sometimes to construct the map or floor plan by the autonomous robot*" [8].

To create the map, the robot must be equipped with several types of sensors, like Sonar sensor, laser, radar, infrared, touch sensor, GPS, camera, and so on [4]. Of course, the sensors have limitations because the precision is never perfect and also their operating range is limited to a certain one, depending on the quality of the sensor. These limitations and errors will not lead to a 100% accurate map so in order to improve map accuracy, several algorithms and methods and methods are being used [4].

These maps can be split into four categories: the metric map, the topological map, the conceptual map, and the cognitive map [4], as shown in figure 1.

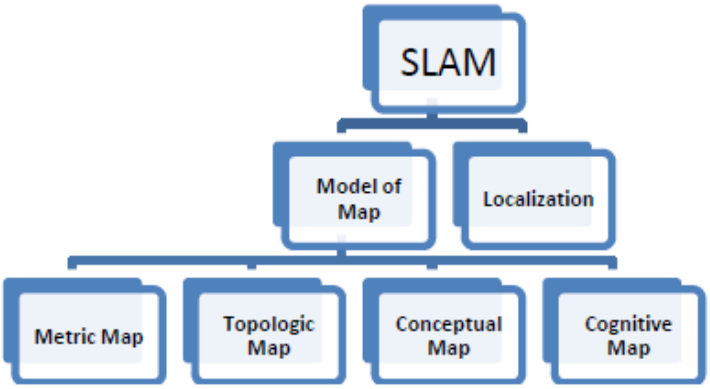


Fig. 1. Map categories [4]

1. *Metric Map*

The environment coordinates are being scaled accordingly using the metric map. One example of the metric map is the occupancy grid, which shows the environment as a "discrete network of cells" [23]. The most common scenario for using this map is when the robot is equipped with a distance measurement sensor, for example a LIDAR or a sonar. Of course, this method has its weakness, more precisely the usage of high memory for saving all the information.

The main objective of this map is the ability to "represent a map of the environment as an evenly spaced field of binary random variables each representing the presence of an obstacle at that location in the environment" [23]. In more simple words, each cell that is occupied will have a specific value, for example, "1", and each free cell will have another value, for example, "0". This kind of map is a non-parametric model because it does not use environment parameters to create the map.

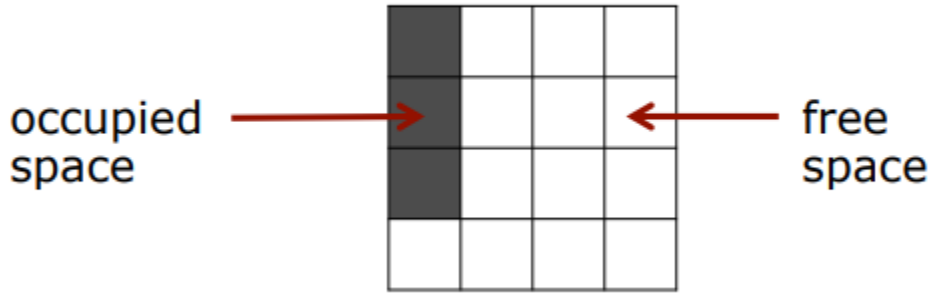


Fig. 2. Occupancy grid example

2. *Topological Map*

This map uses the characteristic of the environment that are effective in robot localization, and avoid characteristic like geometric measurements. In most cases, this map "is a graph of nodes and links" [4]. Here, the nodes inside the graph are pointing to obstacles or places on the environment and the links are pointing to a connection between two places.

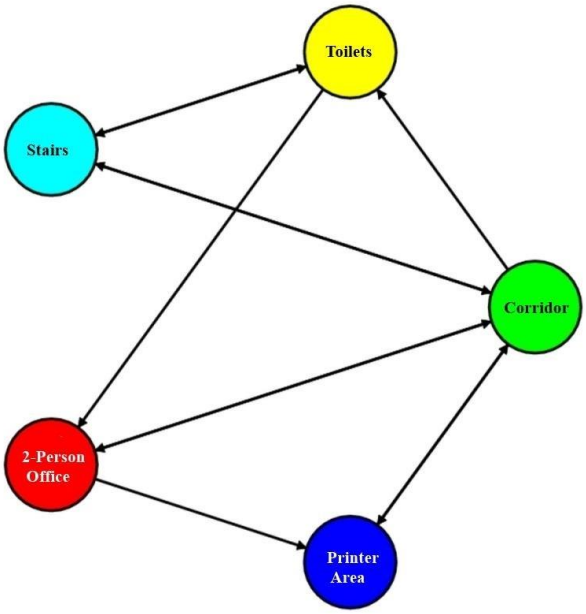


Fig.3 Topological map [24]

3. Conceptual Map

"A concept map is a diagram that depicts suggested relationships between concepts" [25]. The concept uses, in this case, organizing and representing knowledge. The conceptual map diagram could be observed in the following picture:

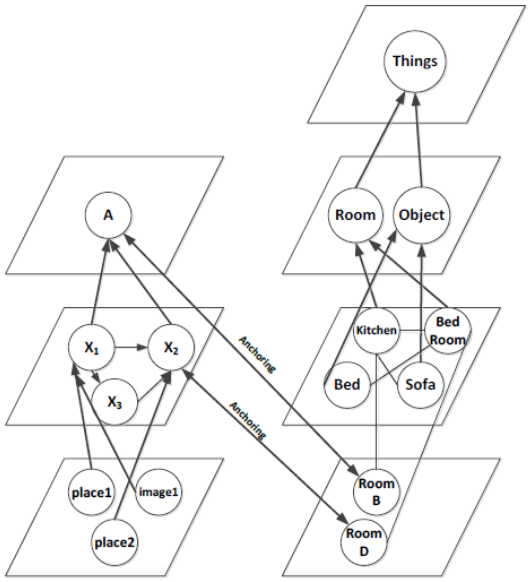


Fig. 4 Conceptual map [25]

4. Cognitive Map

This map will be used for robots with artificial intelligence, which "understand the environment map like humans" [4]. This map will be presented "based on anatomy and function of the human brain" [4]. Using this map, the robot will be able to act and take action like humans.

1.2 Localization

The problem of robot mapping is to acquire information about the robot environment based on a set of data received from a set of sensors. To acquire a map, robots must use a set of multiple sensors, which will allow them to perceive the outside world. Most of the sensors used in this task are cameras, distance sensors (using sonar, laser, and infrared technology), radar, compasses, and GPS. However, all these sensors are subject to errors, like measurement errors or noises, but also some limitations (range limitation, sensor position limitation) [9].

One important aspect when creating a map is the robot localization on that specific map. "Localization is the problem of using sensor measurements to estimate the robot's pose relative to some map" [1]. Localization is essential to decide on future actions, to avoid dangerous situations, for example, collisions or unsafe conditions, for example radiation.

"Robot navigation means the robot's ability to determine its position in its frame of reference and then plan a path toward some goal location" [5]. Considering this aspect, two important factors would need to be accepted: the representational factor meaning that the robot would need to create a map of the environment (based on the input information gathered from sensors) and also the interpretation factor, meaning the process of calculating the robot coordinates in that specific environment.

Coordinates calculation

To establish the agent localization, it is imperative to know the coordinates of the robot. For this, there are multiple methods, like azimuth and elevation method, or extended Kalman filter in a landmark-based map.

In essence, elevation and azimuth are angles, which set the position of an object in the sky. This position has to be relative to a specific observation point [40]. Actually, "azimuth is the angle between a satellite (also, celestial bodies like sun or moon are used) and the North, measured clockwise around the observer's horizon" [42]. This can be translated as a compass bearing, which

establishes the orientation of the object, according to North. Elevation represents the altitude of the observed object, which is the angle between the object and the observer's horizon. Usually, the elevation value is between 0° and 90°.

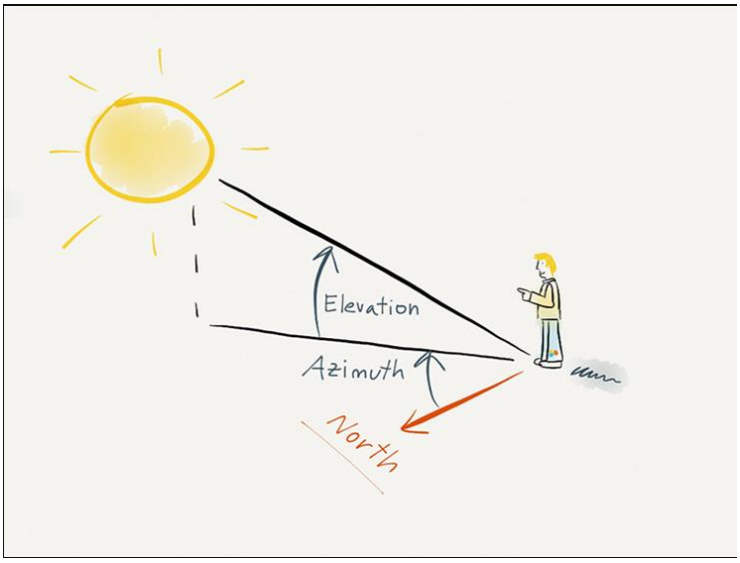


Fig. 5 Azimuth and elevation [41]

Now, to use azimuth and elevation system to establish a robot position, it is also necessary to use some trigonometric computations, for converting azimuth and elevation degree values in gradian values and then in Cartesian coordinates. In a three dimensional map, 'x', 'y' and 'z' coordinates can be calculated using the following formulas:

$$x = distance * \sin(elevation) * \cos(azimuth) \quad (1)$$

$$y = distance * \sin(elevation) * \sin(azimuth) \quad (2)$$

$$z = distance * \cos(elevation) \quad (3)$$

The problem with localization and mapping is that each process is based on having data and information from the robot sensor and actuators. The sensor information is not 100% accurate, so in order to avoid measurement errors, some complex algorithms and calculation will be needed. To create an accurate map of the environment, we need information about the robot position meaning that in order to create the mapping, we need localization. But to determine the robot position, we need to have information about the environment, like a map. This problem is known as the "chicken and egg" problem [3].

1.2.1 SLAM

This problem is being overcome by the SLAM technique (Simultaneous Localization and Mapping) [3]. The fundamental object of SLAM is the detection of the unknown environment by using sensors and also the construction of the map while estimating the pose – localization, and orientation – of the robot. SLAM uses a set of algorithms in order to solve the localization issue and also the mapping problem. One of the key benefits of using SLAM is its ability to constantly monitor the location of the robot that creates a map of the unknown environment. Thus "SLAM is more like a concept than a single algorithm" [21].

Some examples of SLAM techniques are the following: EFK SLAM, FastSLAM, Graph-based SLAM, Lidar-SLAM, and so on, but on each techniques, the problem is the same: we need a map for localization and we need the position estimation for creating the map. So both problems are unknown, and the purpose of the SLAM technique is to solve them.

The most used SLAM techniques algorithms include Kalman filters and particle filters (Monte Carlo methods). Those algorithms use an estimation of the probability function for robot pose and map parameters [21]. Most of SLAM algorithms are implemented on ROS – robot operating systems, which is an open-source library, which is used together with Point Cloud Library for creating the 3D map or with visual features from OpenCV.

SLAM application can be found in automatic car piloting or unrehearsed off-road terrains, rescue tasks for the high-risk or difficult-navigation environment, planetary, aerial, terrestrial, oceanic exploration, medicine, and many more.

The most important parameters in the investigation of the SLAM problem are [4]:

- Sensor uncertainty
- Correspondence
- Loop closing
- Time complexity
- Dynamic environment

1. Sensor uncertainty can be explained by the accumulation of small errors. This uncertainty can be described by:

a. Restriction of incoming – this is related to sensor limitations; most of the sensors have small distances limitation, which will lead to errors when trying to read further distances.
b. Sensor fault – this is caused by noises read by the sensors.
c. Mistake/Slip – this is caused by robot movement errors; a small slip can cause problems.

2. Correspondence issue which is also called data relation. This problem will determine if "sensor measurements at different times relate to the same physical object or not" [4].
3. Loop closing performs after the previous issue. When the robot is in a loop, it has to decide its position on the map. "It is therefore difficult because during the closing loop the accumulated error can be too high" [4].
4. The time complexity issue is related to robot performance. This means that the robot must perform in real-time, in order to be fast.
5. Dynamic environment – this issue is related to changes in the environment. "This issue can make two hypotheses for the robot: first, the environment has changed. Second, the robot has entered in a new place" [4].

1.2.2 Obstacle detection

"Obstacle detection is the process of using sensors, data structures, and algorithms to detect objects or terrain types that impede motion" [26]. To detect the obstacles, there are several methods, based on the equipment or sensor used [7], most of them based on distance measurement. So, we can have the following categories of sensors:

- Ultrasonic sensors – which uses ultrasonic waves to measure the distance to an object.
- Infrared sensors – which uses a light source to measure the distance to an object.
- Proximity sensors – which uses an electromagnetic field to measure the distance to an object.
- Push sensors – uses physical contact to detect the distance to the object.
- Accelerometer – uses the orientation of the robot. "When a robot hits an object, the accelerometer marks a motion in the opposite direction according to Newton's third law" [7].
- Gyroscope – is constantly monitoring the changes in the orientation. "When the robot is being hit by an obstacle, the large variation is marked as detection of obstacles" [7].

Obstacle avoidance methods can be classified into two categories: Direct detection and avoidance and Indirect detection and avoidance [7].

1. Direct detection and avoidance – DDA – this method allow the robot to touch the obstacle and react according to the collision. This means that after the collision, the robot will move away from the obstacle.

In this case, the most commonly used sensors include push sensors, accelerometer, and gyroscope, which permit the robot to touch the obstacle and react. One example of a robot using a push sensor for obstacle avoidance can be seen in the following image:

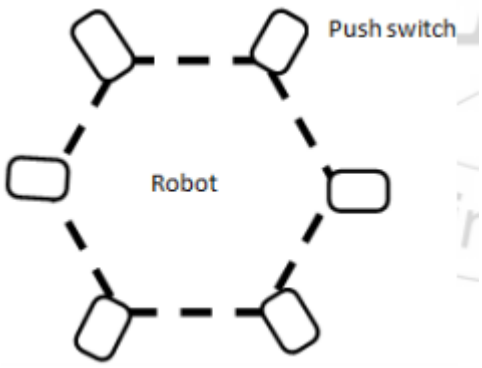


Fig. 6. Direct detection and avoidance robot [7]

2. Indirect Detection and Avoidance – IDA – this method does not allow the robot to reach the obstacle and react according to sensor information.

In this case, the most used sensors are ultrasonic, infrared, or proximity sensors which read the distance to the object, send the information to the robot, on which the path calculation will take place, in order to avoid the obstacle. One example of a robot using an infrared sensor for obstacle avoidance can be seen in the following image:

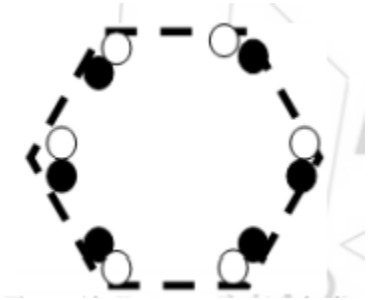


Fig. 7 Indirect detection and avoidance robot [7]

1.2.3 Lidar technology

Lidar technology, also known as 'Light Detection and Ranging' technology, uses light technology to detect the distance to an object. The basic principle of the lidar sensor is based on laser light and measures the reflection of it. "Lidar is a remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to the Earth" [11].

A lidar device or sensor sends laser lights and measure how long does it take for the light to come back. The basic principle is known as "time of flight" measurement, where light beams are emitted by the device or sensor toward an obstacle or object, and then reflected and collected in the device. The returned light beam contains information regarding distance to the obstacles and sometimes optical characteristic, like reflexivity [27].

The main goal of a lidar sensor is supporting an autonomous car to navigate in an environment, but the sensors have also many additional applications, across various industries and fields. Lidar technology is similar to radar or sonar technology, but still has some improvements like the precision of sensing [27]. The lidar precision can create a map, also known as "point cloud" [27], which represent a dense map of measurements, which can be seen on a display like physical objects.

Lidar technology can be used in several modes, from autonomous vehicles to agriculture or image processing. According to the *Level Five supplies* website [12], which found "100 Uses for LIDAR 3D sensing technology", the most used application can be found in digital elevation models, agriculture, astronomy, biology, and conservation, image recognition, and so on.

Further on will be presented some examples of lidar sensors used in different areas.

1. Autonomous vehicle

In this case, Lidar is popular as guidance systems for autonomous vehicles and has some particular usage like collision avoidance, thanks to its speed and accuracy, autonomous cruise control and obstacle detection. "Lidar enables a self-driving car" [12].

2. Digital elevation models(DEMs)

These models are used to create 3D models or representations of a surface. In this case, Lidar made huge progress in this domain, based on its speed and easiness to use; before lidar, the map was created using photogrammetry or ground surveys.

3. Agriculture

In agriculture, LIDAR sensors are used to take precise measurements, which can be used for topographic analysis and prediction of soil properties. Also, with LIDAR technology, it is easier to categorize crops based on their characteristics. “A crop may thrive in one area of the farm, but may not do well in another area” [12].

4. Image recognition

This area includes gesture recognition, motion analysis, and lip-reading. In the first case, a lidar sensor can be used to take very fast measurements, in order to keep up with the person's gestures. These measurements can be used in automotive, in order to detect driver gestures or also can be used in-game industry. For the leap reading, it is still much to do. After all, it is hard to read someone's lips, because it depends on the speaker's lips, or spoken language and pronunciation.

5. Biology and conservation

Lidar technology can be used from biodiversity, to flood modeling or earthquake damage. A lidar was also used to create a map, which shows flooding extension in New Orleans after Hurricane Katrina.

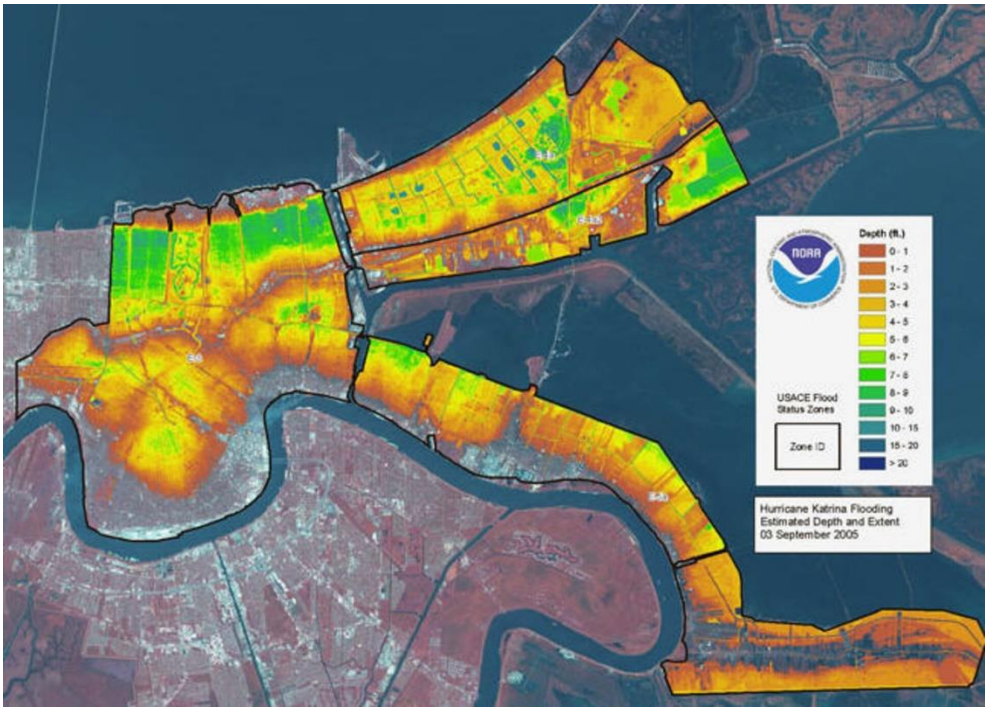


Fig. 8 Lidar map after the Katrina hurricane [28]

2 SCIENCE ACCOMPLISHMENTS USING LIDAR SENSOR

2.1 Combination of low-cost LIDAR sensor and vision sensor

A method of SLAM is developed by Guolai Jiang, Lei Yin, Shaokun Jin, Chaoran Tian, Xinbo Ma, and Yongsheng OU at the Chinese Academy of Sciences and University of Chinese Academy of Science, which uses light detection and ranging sensor, adopted for robot navigation. [2]. Their paper proposes a "new graph optimization-based SLAM framework through the combination of low-cost LIDAR sensor and vision sensor" [2]. The project will create a 2.5D map, which will include obstacles and vision features and also a relocation method.

The experimental robot will be equipped with 360° low-cost LiDar and a front-view RGB-D camera and will be used in a real indoor scene. The main goal of the project was the comparison between the current approach and other approaches (such as using only a Lidar sensor or camera) and as consequence, this method had a better result. The SLAM framework based on graph optimization is divided into 2 parts: front-end and back-end [2], as shown in the following picture:

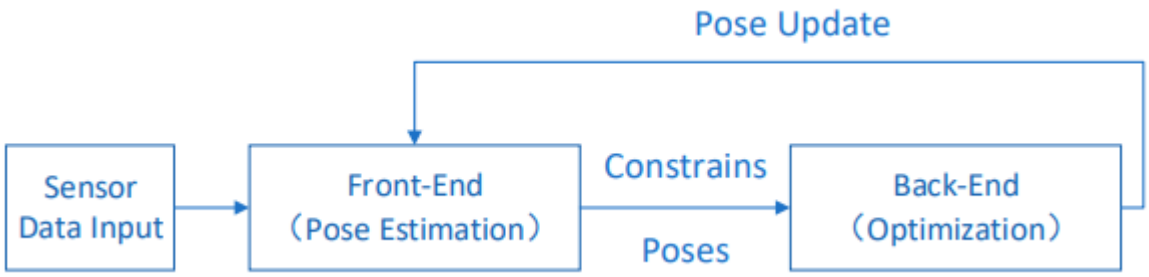


Fig. 9 SLAM framework [2]

The front-end estimates the position of the robot, using information from the sensor, however, this data contains noises (both image and laser data). The noises present in data will lead to cumulative errors in pose estimation, which will increase with time. The back-end part is responsible to eliminate the errors and improve the positioning of the robot. Also in the back-end, graph optimization will be used, and the error is "minimized by the descending gradient through nonlinear optimization" [2]. This graph optimization describes the problem of optimization in the "form of a graph" [2]. Each node of the graph represents the position and the attitude, and each edge represents the relationship between the position and the attitude [2].

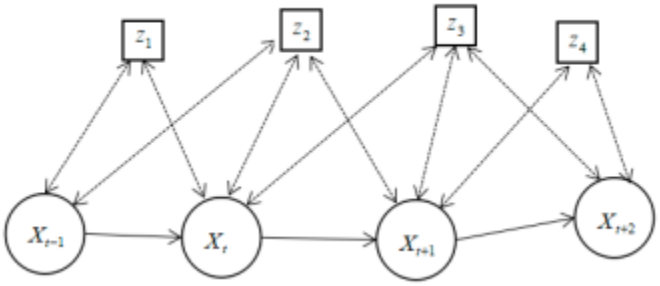


Fig. 10 Graph representation [2]

In this figure, ‘X’ values represent the position and pose of the robot and ’Z’ represents the observation (Z will be a combination between camera and obstacles detected by Lidar).

The SLAM framework of Lidar and vision fusion has a new united error function, which combines visual data-matching error and laser data-matching error. Also, it is used as a loop detection method, to solve the problem of loop detection of tradition Lidar-SLAM.

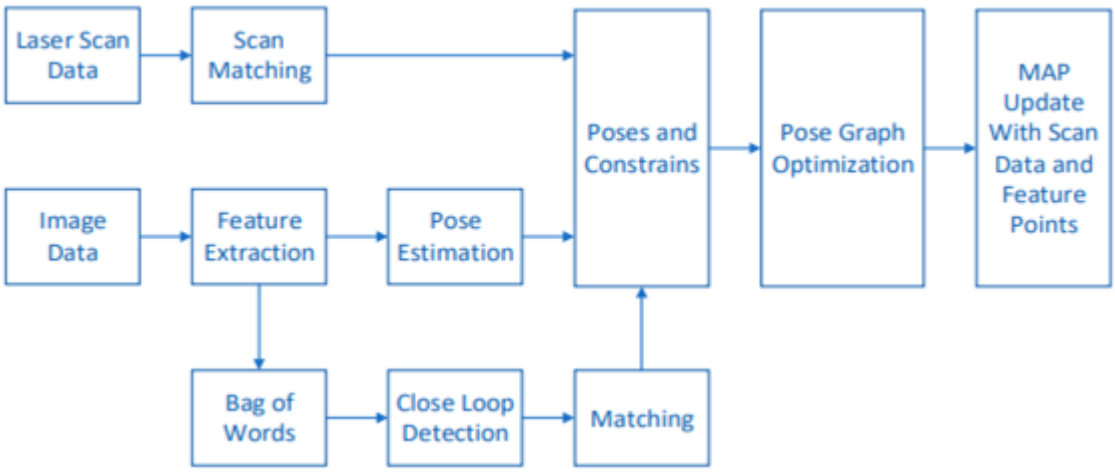


Fig. 11 SLAM framework [2]

2.5D map

Here the authors introduce a new concept, the 2.5D map, which is based on visual collected data and scan data. The difference between the traditional grid map and 2.5D map is that the traditional grid is a simply 2D map that will present the obstacles on the LiDar plane, while the 2.5D map combines obstacle representation on a 2D map with other backup features, called feature list, all represented in 3D space.

The experiment developed is divided into three main parts:

- The first part uses a “comparative experiment of fixed-point positioning accuracy” in a small range of scenes [2].
- The second part uses a large loop experiment to verify the efficiency of the proposed method.
- The third part is used to load the build map for the relocalization experiment.

As hardware parts, the authors use a platform based on Turtlebot 2, which will contain the following parts: a notebook, a Lidar(model RPLIDAR-A2), and an RGB-D camera (model Xtion-pro).

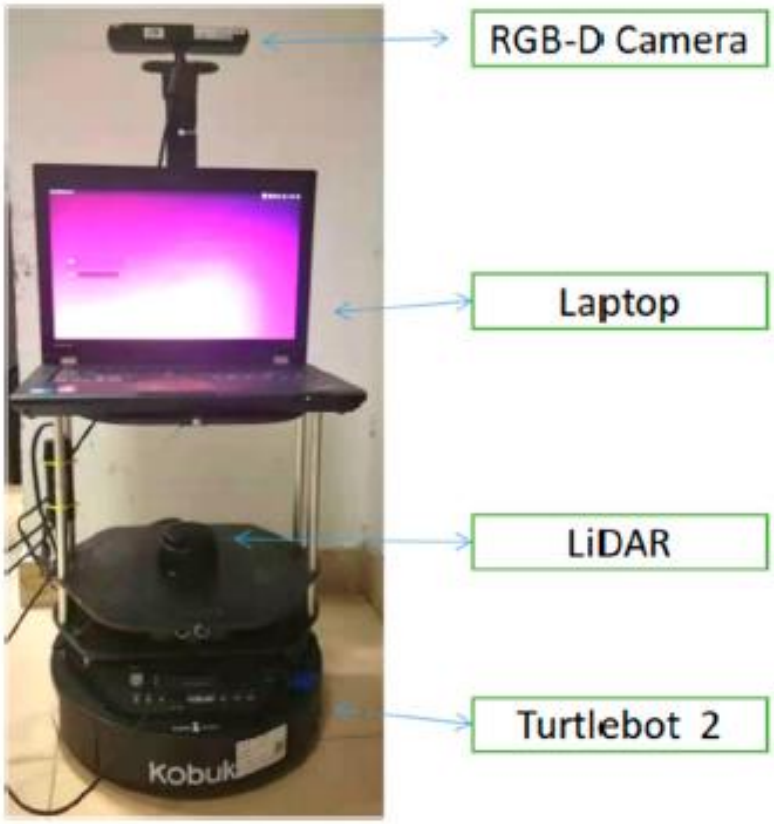


Fig. 12 Hardware components [2]

Robot platform

For a better evaluation of the result, the authors used 6 positions: the start point position which is represented with (0,0), the first position represented with (3,0), the second position represented with (6,0), the third position represented with (6,-8), the fourth position represented with (6,-16) and the last position represented with (3,-16).

Table 1 will present a comparison between the real position, Karto-SLAM, and the method proposed by the authors.

Table 1. Proposed method against Karto-SLAM and real position[2]

Position Number	Real Position (m)	Karto-SLAM (m)	Our Method (m)
1	(3,0)	(2.995,−0.002)	(2.994,−0.003)
2	(6,0)	(5.987,−0.003)	(5.990,−0.005)
3	(6,−8)	(6.025,−8.032)	(6.019,−8.021)
4	(6,−16)	(6.038,−15.954)	(6.028,−15.973)
5	(3,−16)	(2.949,−15.946)	(2.951,−15.965)

After several experiments, they created the 2.5D map, by comparing the Adaptive Monte Carlo Localization method, orb-SLAM localization method, and their proposed method.

Table 2. Proposed method against Karto-SLAM and real position [2]

method	Map and Sensor	With Given Initial Pose		Without Given Initial Pose	
		Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
AMCL [34]	Grid map with LiDAR	95%	6.2	30%	20.8
Orb-SLAM Localization [9]	Feature map with camera	-	-	88%	5.8
Proposed method	2.5D map with LiDAR and Camera	95%	8.7	92%	6.6

The AMCL method has a success rate of 95% when the initial pose is given but has a very small success rate when the initial pose is missing. The orb-SLAM method is capable of fast global relocation, with the fast response time. The method proposed by the authors is faster and has a higher success rate, without the initial pose, in comparison with the above methods.

2.2 LIDAR sensor and servo-motor

The second project studied is developed by I.Maulana, A. Rusdinar, and R.A. Priramadhi at School of Electrical Engineering, Indonesia[6]. Their project proposes the creation of the LIDAR sensor, using one laser sensor, which will be rotated by a servo-motor. The information read by the sensor will be transformed in Cartesian axes, which will be later used to create the local map and to localize the position.

To create the map, several stages of data collection are used:

- Rotation of the sensor using the stepper motor
- Information read by Arduino from LIDAR
- Sending data from Arduino to Matlab, using a Bluetooth device.
- Data processing by Matlab and map creation.
- Sending command from Matlab to the robot, for controlling the robot.

The flowchart of the project follows the above stages and is presented in the following picture. The occupancy grid is represented in tow formats: binary occupancy grid and probability occupancy grid. The binary occupancy grid uses Boolean values: True values to represent the occupied environment – obstacles and False values represent the free environment – non-obstacles. In the probability occupancy grid, there are used probability values to create a more presentative map. "This representation is the preferred method for using the Residential grid" [6]. The Residential grid uses cells, and each cell has a value representing the probability of cell occupancy. If the cell contains an obstacle, the probability value will be close to 1, and if the cell is not occupied, the probability value will be close to 0.

"Probability road map (PRM) is a network graph of paths that may be present in a map determined by free and unimpeded space" [6]. This map takes random samples from the map and each sample is verified if it's in an empty cell or occupied cell. After the samples are taken, the local planning is made, and after that, each plan will be connected, based on the nearest. The probability road map has two stages: the construction stage and the second stage determine the shortest path using the Dijkstra Algorithm.

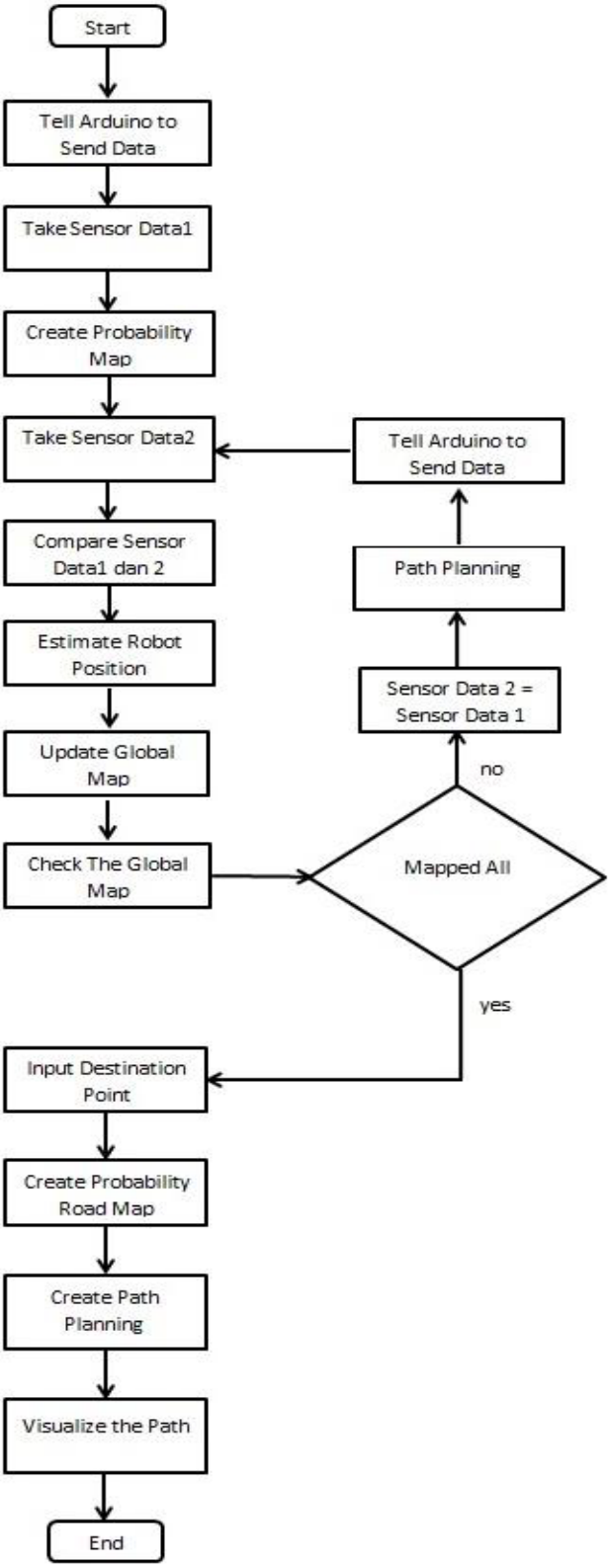


Fig. 13 Flowchart [6]

The hardware design of their project is presented in the following figure.

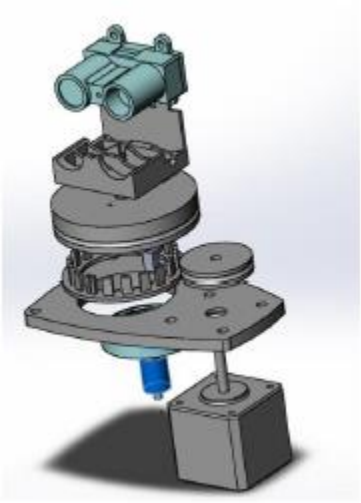


Fig. 14 Hardware design [6]

The lidar used in the project is the LIDAR-Lite V3, from Garmin. The conversion from distance and angle data in the Cartesian form is made with the following formula:

- Convert degree to radian

$$\theta_{rad} = \theta_{deg} \cdot \frac{\pi}{180} \tag{4}$$

where θ_{rad} denotes angle in radian and θ_{deg} is angle value in degree.

- Convert polar to Cartesian

$$x = rho \cdot \sin(\theta_{rad}) \tag{5}$$

$$y = rho \cdot \cos(\theta_{rad}) \tag{6}$$

where x is the position in the x-axis and y position in the y-axis.

Inside the project, several tests are made to show the efficiency of the lidar sensor. As shown in the below table, the deviation value is increasing in the same time with the distance value; if the measured distance is smaller, the deviation is small, but if the measure distance has a higher value, the deviation is quite big (for example for 2000 cm, the deviation is 9.6 cm). According to figure 15, it could be observed that the reading of the lidar sensor is close to the actual distance.

Table 3 Difference between actual measurement and sensor measurement [6]

Actual distance measurement (cm)	Average 30 Times Distance Measurements By Sensor (cm)	Error Average Distance Measurements by Sensor (%)	Standard Deviation of Distance Measurement by Sensor (cm)
15	16	0.0667	1.05
16	16.73	0.0458	1.53
17	17.13	0.0078	1.59
18	18.30	0.0167	1.95
19	19.50	0.0263	1.69
20	20.50	0.0250	1.94
30	33.27	0.1089	3.18
75	74.93	0.0009	1.57
90	90.87	0.0096	1.38
100	103.80	0.0380	2.11
150	155.17	0.0345	2.12
300	308.17	0.0272	3.51
600	603.37	0.0056	3.77
1200	1218.77	0.0156	10.11
2000	2008.87	0.0044	9.62

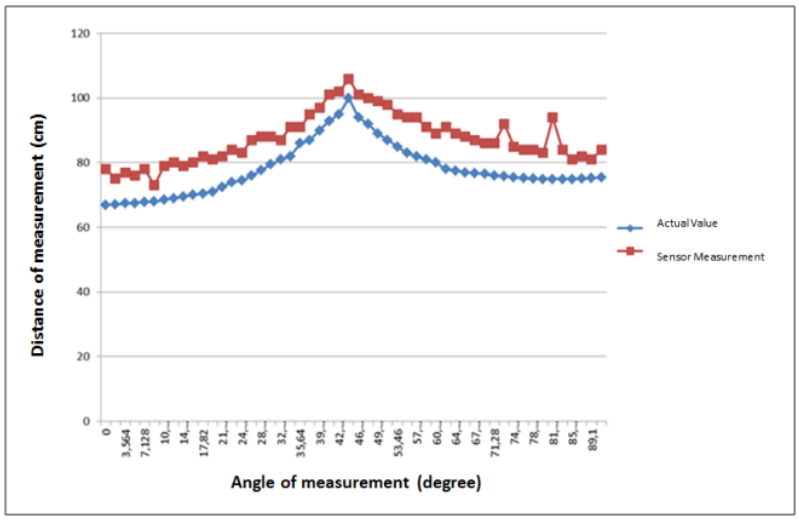


Fig. 15 Comparison between the actual values and read values [6]

The next two tests are made on measuring distance on different surface colors, respectively on the different surface material. On the color test, it can be seen that each different measurement

color har different deviation values, so the color can influence the mapping process. On the other hand, the surface material also influences the measured values.

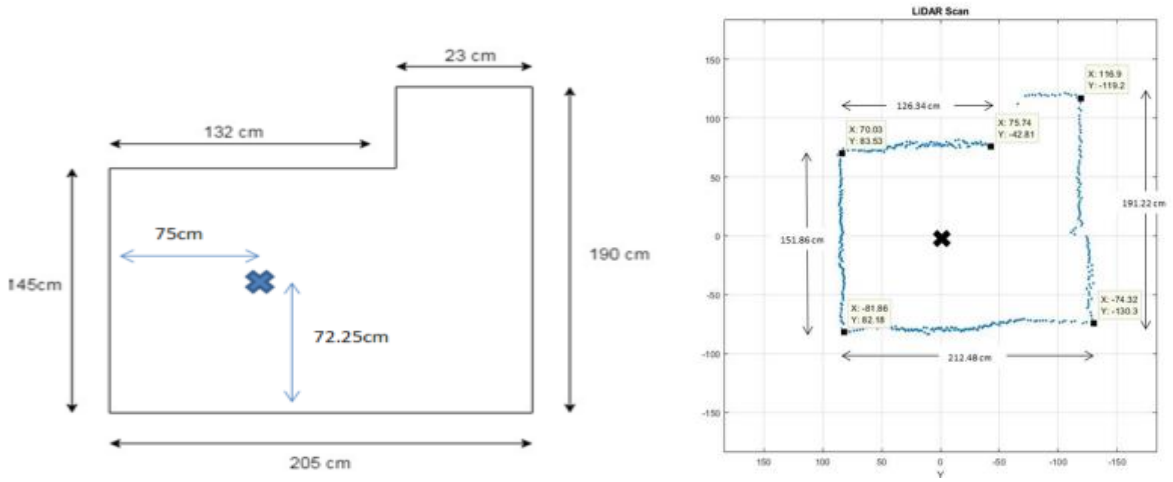
Table 4 Distance measurement on Color [6]

No.	Distance Measurement on Color				
	Red	Green	Blue	White	Black
1	103	104	104	101	88
2	103	105	106	103	99
3	105	105	105	102	104
4	114	105	105	102	105
5	110	104	103	105	96
6	102	103	105	104	101
7	102	106	104	102	102
8	105	103	103	103	98
9	103	104	107	104	106
10	102	104	106	101	103
11	105	105	107	105	106
12	104	103	101	102	104
13	103	104	103	104	101
14	105	103	105	102	96
15	102	104	104	101	102
16	104	106	103	101	98
17	103	104	103	100	99
18	102	103	101	101	102
19	106	103	102	102	100
20	105	103	102	103	98
Average (cm)	104.4	104.05	104.05	102.4	100.4
Standard deviation (cm)	2.96	0.99	1.67	1.43	4.22

Table 5 Distance measurement on Material[6]

No.	Distance Measurement on Material				
	Concrete	Plastic	Wood	Iron	Cardboard
1	81	87	84	102	101
2	104	97	102	104	109
3	103	106	101	103	110
4	105	100	103	105	117
5	106	103	102	103	118
6	102	96	104	103	110
7	100	101	101	109	111
8	100	101	103	105	108
9	107	100	102	106	106
10	106	98	104	105	113
11	101	101	103	103	109
12	100	101	104	95	110
13	105	101	99	105	110
14	105	101	99	105	120
15	106	115	100	104	106
16	100	111	101	106	114
17	102	117	103	105	116
18	97	105	102	109	107
19	81	105	101	105	111
20	107	102	100	107	115
Average (cm)	100.71	102.42	101.09	104.6	113.5
Standard deviation (cm)	7.39	6.62	4.24	2.99	4.75

The next test is the visualization of the lidar sensor scan result values. The purpose of this test is to see a comparison between the real environment and the result of lidar values. The result is very close to the original map, but also present some “variation of sensor readout error” (where the line is not straight) [6].

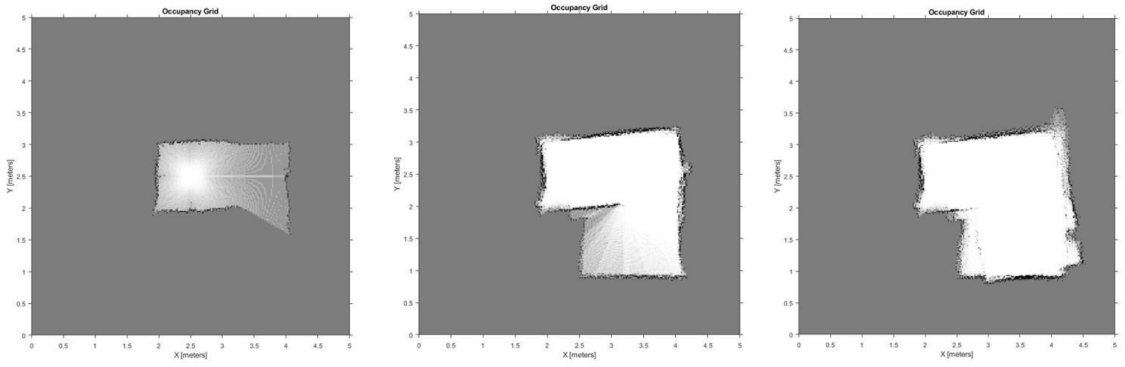


(a) (b)

Fig. 16 Comparison between actual room dimension (a) and sensor results (b) [6]

The last test makes a comparison between the map after 1 step of scanning, 10 steps of scanning, and after 20 steps of scanning. In this test can be seen that the error on the visualization of the map is caused by reading the current orientation of the agent in the environment. “Because the position and orientation estimation based on current and previous scan data, the resulting error is getting bigger and cause the mapping result the less good can be seen from the visualization result which is getting away from the actual shape at every step of mapping step” [6].

The tests result shown that Lidar Lite V3 has different standard deviation values, depending on the distance, color, material, and angle, a deviation that will lead to mapping errors [6].



(a) (b) (c)

Fig. 17 1 step of scanning (a), 10 steps of scanning (b) and 20 steps of scanning (c)

2.3 Mobile Robotics Mapping using RP Lidar Scanner

Another interesting project in this field is called "Mobile Robotics Mapping using RP Lidar Scanner"[10] which was done by 4 students from the Perlis University of Malaysia. The project consists of using a low-cost LIDAR sensor to track the objects in an indoor environment. After retrieving the distance information from the sensor, 2 algorithms are being used to get rid of false values and also to successfully map the environment data (such as fixed objects, etc.).

The LIDAR sensor is called RP LIDAR and is a low-cost LIDAR sensor developed by the Robopeak company. The sensor can scan the environment at a distance of about 6 meters with 360 degrees. Each round of scanning consists of 360 sampling data. The sensor is visible in figure 17.

Fig. 18 RP LIDAR [10]

The physical part consists of chassis, tires, a motor, a driver, a computer(for data processing), a power supply, and a wireless transmitter. An image of the robot is displayed in figure 18.

Fig. 19 Hardware components [10]

Since the project does not have a really big complexity, the scenarios of the program consist of the following:

- The robot starts at the initial point(have 0,0 values in the x-y coordinate system).
- The robot starts moving and after the sensor detects an object it stops moving.
- The robot does a 360-degree scan and saves the data received.
- Depending on the area, the robot moved to a specific direction(forward, to the right, to the left, backward).

These steps are being repeated until all the points in the room are being calculated, so that, by the end of the repetition process, we will have all the information regarding the environment.

In order to get rid of the unnecessary data, 2 algorithms are being used: the raw filter algorithm and the smooth method.

- The raw filter is an algorithm that removes invalid data. "This is not the best solution for image pre-processing, but, it gives simple computation, fast and very practical to a simple mapping application". Since the maximum value is 6m, then all data is mapped to be from 0-6m, discarding all the other data which is above/below this interval.
- Another algorithm that is being used is the smooth algorithm. This will ensure that some patterns will be captured from the existing data. "In this study, the smooth filter performs the small correction using the moving average method where the angle results become the input." A graphical comparison of the row data and the data after the smoothing process is being displayed in figure 20.

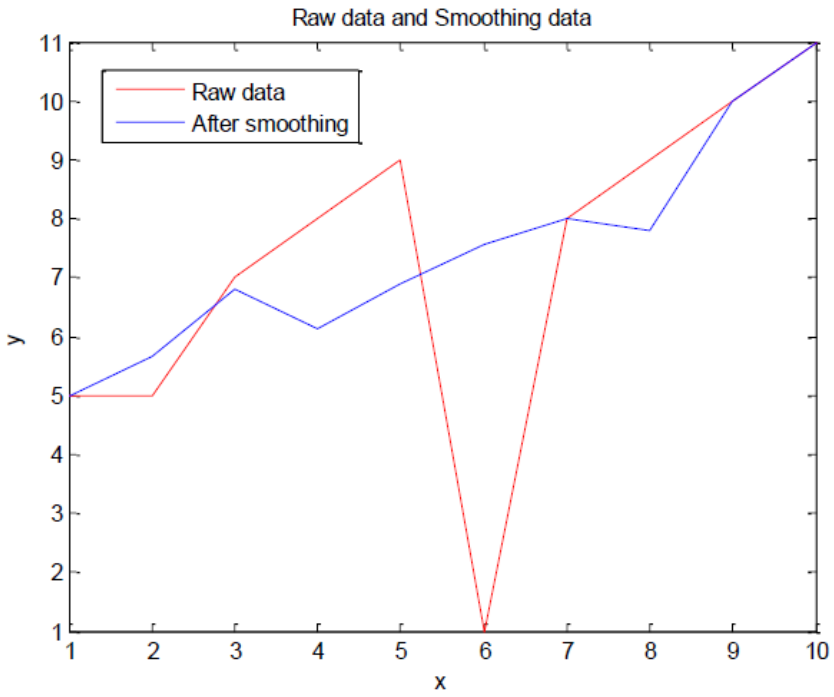


Fig. 20 Comparison between raw data and smooth data [10]

"Based on the results, this study shows that the use of the RP Lidar is impressive in terms of its accuracy of scanning. Most of the trials get 90% above the scanning accuracy" [10]. Since the experiment was not run in a particular or standard procedure, some errors were being made, considering the fact that the agent could not stop at a particular point.

3 APPLICATION

3.1 Formulation of the project theme

In order to apply all information and techniques presented earlier, I developed an autonomous robot, which uses a low-cost lidar, to create a 2D and 3D map, and also avoid obstacles. This kind of application is framed in the automotive application, having large applicability, including in autonomous cars or even in-home appliance applications, like a smart vacuum cleaner.

This paper gathers several projects in a single one, projects presented earlier in this paper. At this moment, lidar sensors are used in a lot of projects and represent one of the most used sensors for detecting the distance, because of their speed and very high precision. However, lidar sensors are not used as much in small applications, because of their high cost, so it is used predominantly in big applications.

The motivation of creating this robot, with a lidar sensor, is to show the capabilities of this sensor, together with Matlab and Arduino, and also to present one of the most widespread usages of this kind of sensor.

This paper is divided into two parts: the hardware part and the software part.

3.2 Hardware part

This part will present all hardware components used at robot creation, and will also include all explanation of functionalities and methods used.

The following components were used:

1. Robot chassis
2. Motor shield
3. Lidar sensor
4. Bluetooth module
5. Batteries
6. Arduino
7. Basic components: breadboard, buttons, logic level converter, etc.

3.2.1 Robot chassis and motor shield

The robot chassis includes 4 wheels with 4 independent DC motors, controlled by a motor shield. The purpose of the motor shield is to control the speed and the direction of the motors, using direct current control. This is a power amplifier, which takes power from Arduino or batteries and sends it to the motors.

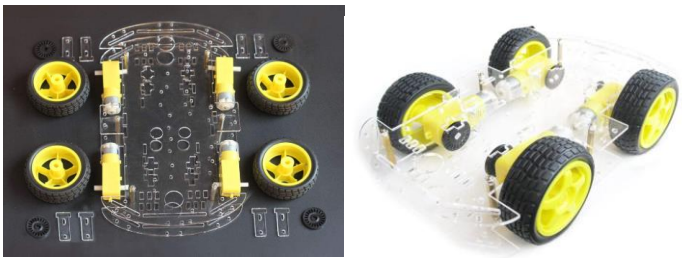


Fig. 21 Robot chassis [34]

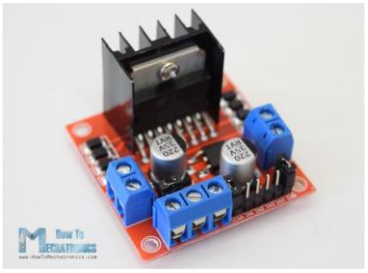
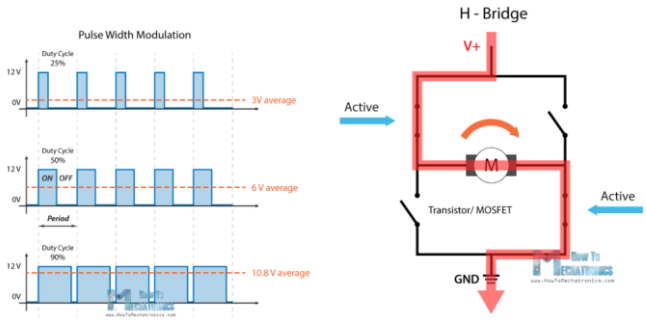


Fig. 22 Motor shield L298N [35]

In order to control the speed of the motors, the motor shield uses PWM signals, and for controlling the motor direction it will inverse the direction of the current flow through the motor.



(a) (b)

Fig. 23 PWM control (a) and direction control (b) [35]

3.2.2 LIDAR

The lidar sensor used in the project is a low cost unidirectional ranging lidar sensor, specially created for Arduino boards, called “Benewake TF Mini micro lidar”. The sensor range varies from 30 centimeters to 12 meters, with long-range and high-precision of 1 cm and very low power consumption.

This sensor principle is based on TOF, namely, Time of Flight principle [13]. This principle is a method for measuring the distance between an object and a sensor, based on the time difference between the emission of a signal and its return to the signal, after being reflected by an object, as shown in the following picture.

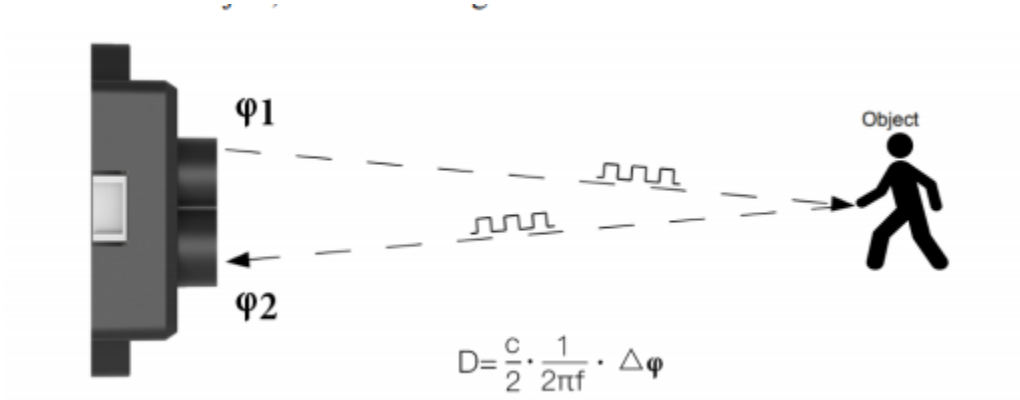


Fig.24 TOF principle [13]

As described before, the sensor operation range is from 30 cm to 1m, with an accuracy of ±4cm, if the measured distance is between 0.3 cm and 6 m, and a ±6cm, if the measured range is above 6 m. The measurement range of the sensor is being influenced by the environment illumination intensity and also by the reflectivity of any detected object [13]. According to TF mini product manual, there are 5 areas of measurement ranges, as follow: [13]

1. The first area is the blind area of the sensor, between 0 cm and 30 cm, in which the data read by the sensor are unreliable.
2. The second range is represented by the extreme condition, which is between 0.3 m and 3m. This is referring to outdoor open fields, where illumination intensity is about 100klux, and detection of a black target, where the percentage of reflection is 10%.
3. The third range represents the range for a white target on normal light conditions, at approximatively 70klux illumination intensity, which includes also the seconde range and is between 0.3 m and 7 m.

4. The forth range is represented by the indoor environment, or the range where the ambient light is very low, and consist between 0.3 m to 12m.
5. The last range represents the "minimum side length of effective detection for TFmini at different distances. For the data to be reliable, the side length of the detection object must be equal or more than the minimum side length" [13]. This minimum side length depends on FOV of sensor – FOV is basically the smaller value between the receiving and the transmitting angle, and has the following formula:

$$d = 2 \cdot D \cdot \tan(\beta) \tag{7}$$

,where d is the minimum side length of effective detection, D is detection range and β is half of the value of the receiving angle of the sensor, 1.15°.

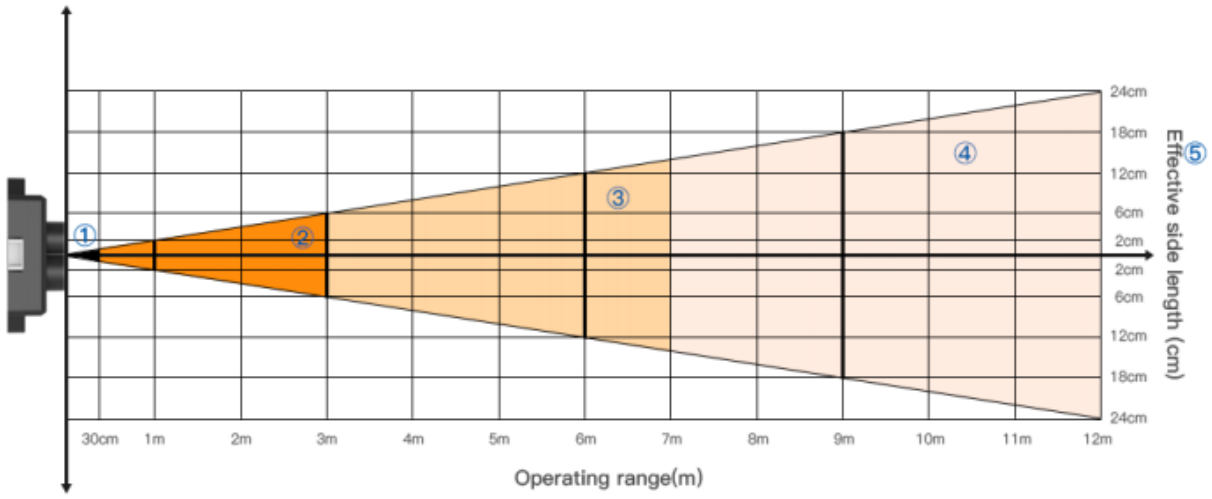


Fig. 25 Measurements area [13]

Electrical characteristic:

Tabel 6 Electrical characteristic [13]

Description	Parameter value
Power supply voltage	5V
Average current	≤120mA
Peak current	800mA
Average power	≤0.6W
Communication level	LVTTL (3.3V)

Data communication protocol

The sensor used the serial port communication protocol, as shown in table 7. The sensor output came in two formats: the standard data output format and the Pixhawk data format.

Tabel 7 Communication protocol [13]

Communication interface	UART
Default baud rate	115200
Data bit	8
Stop bit	1
Parity check	None

The first one has each data package in 9 bytes, which include the header of the package, distance information, signal strength information, distance mode, and data check byte (CRC). The format of this data is hexadecimal, and is detailed in the following picture:

Tabel 8 Data package [13]

Byte0 -1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x59 59	Dist_L	Dist_H	Strength_L	Strength_H	Mode	0x00	CheckSum

Data code explanation:

- Byte 0: 0x59 represent the frame header, same for each frame
- Byte 1: 0x59 represent the frame header, same for each frame
- Byte 2: Dist_L represent the lower value of the distance – 8 bits
- Byte 3: Dist_H represent the higher value of the distance – 8 bits
- Byte 4: Strength_L represent the lower value of the strength – 8 bits
- Byte 5: Strength_H represent the higher value of the strength – 8 bits
- Byte 6: Mode represents the distance mode and can be 02 for short distances and 07 for long distances, which is automatically switchable by default.
- Byte 7: Not used, 00 by default
- Byte 8: Checksum represents the cumulative sum of the numbers of the first 8 bytes – 8 bits.

1. Description of default Output data

- Dist – distance value which is taken from the sensor, with cm as the default unit. The value of the distance is interpreted into a decimal value, in the range of 0-1200 (0 cm-1200 cm). The distance value is influenced by Strength value: if strength is smaller than 20, the distance value will be considered as not reliable, and the output of distance will be FFFF; if strength is bigger than and the actual value is bigger than 12 meters, the output of distance will be 1200(cm).
- Strength – signal strength, having default value in the range of 0-3000. Signal strength is influenced by the distances measured but also by the reflectivity, as follows: if the measured distance has a higher value, the signal strength will be smaller; if the value of reflectivity is small, signal strength value will small.
- Mode – it is used to indicate the distance mode of the product. The sensor has two modes, namely, 02 and 07, which represent short-distance operating mode and long-distance operating mode. The switch between these modes is made automatically based on the value of the distance; this will also influence signal strength value.

2. ***Pixhawk data format*** is the format of the character string and its unit in meters. For instance, for a distance of 2.74m, the output string will be 2.74, followed by the escape character.

3.2.3 HC-05 Bluetooth Module

"Bluetooth technology is a high speed low powered wireless technology link, which is used to connect multiple devices like phones or other portable types of equipment" [18]. Bluetooth uses UHF radio waves, from 2.402 GHz to 2.480 GHz [19]. The IEEE standardized Bluetooth as IEEE 802.15.1 [19].

Bluetooth technology is used for short distances, typically up to 10 meters. It allows the connection of 8 devices simultaneously, and for each device, it will offer a unique 48 address (according to IEEE 802 standard).

Bluetooth network consists of a Personal Area Network which containing 2 devices (minimum), usually a master and up to 7 slaves [18]. The operation mode is defined by the following scenario: the master device will send a radio a message asking for a response from one particular slave (using slave addresses); then the slave responds and synchronizes their frequency with the master device [18].

As every communication method, Bluetooth has some specifications, defined by "Bluetooth Core Specification Working Group – CSWG) [1], which can be split into two categories: core specification and profile specification [18].

- Core specification – this includes the Bluetooth protocol stack and requirements – used for testing Bluetooth based products. This specification can be divided into 5 layers: [18]
    - Radio – include frequency, modulation and power specification
    - Baseband layer – including the definition of the channel used: physical or logical; link types; definition of packet format; transmit and receive timing; it defined channel control and device address.
    - LMP – Link Manager Protocol (LMP) – this establishes the procedure for link set up.
    - Logical Link Control and Adaptation Protocol (L2CAP) – this layer adapts the upper-layer to be the baseband layer.
    - Service Discovery Protocol (SDP) – layer used to allow querying Bluetooth devices.

Bluetooth architecture

Bluetooth architecture is mentioned as scattered ad-hoc topology. This topology uses a small cell called Piconet, which includes a collection of devices connected in an ad-hoc manner [17]. Bluetooth devices can be divided into four types[17]:

- Master(M)
- Slave(S)
- Stand By(SB) – Stand by device is waiting to join the Bluetooth network meanwhile it saves its MAC address in the network.
- Parked/hold(P) Parked device is waiting to join the Bluetooth network later meanwhile and after that release its MAC address.

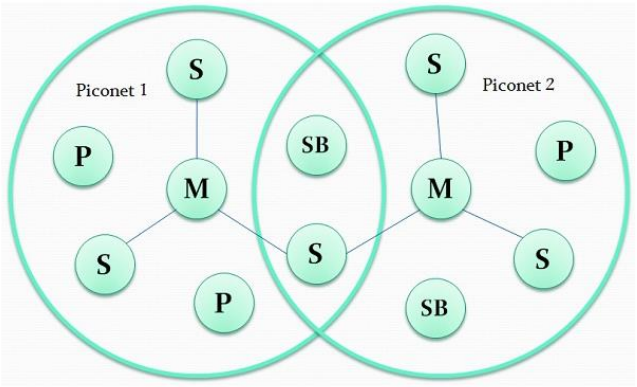


Fig. 26 Bluetooth states [17]

Physical connection

The physical connection between a Bluetooth device and a nominal antenna power is realized via FHSS – frequency-hopping spread spectrum modem. The power of the nominal antenna establishes the coverage of the Bluetooth signal: if the power is 0 dB the coverage will be 10 meters and if the power is 20dB the coverage will be 100 meters.

Bluetooth vs wireless technology

The main differences between the Bluetooth network and wireless network are defined in the following table[17]:

Table 9 Comparison between Bluetooth and wireless

	BLUETOOTH	WIFI
Bandwidth	Low	High
Range	10 meters	100 meters
Security	Less secure	Security features are better
Power consumption	Low	High
Frequency range	2.4 GHz and 2.483 GHz	2.4 GHz and 5 GHz
Flexibility	A limited number of users	A large number of users
Modulation techniques	GFSK (Gaussian frequency-shift keying)	OFDM (Orthogonal frequency division multiplexing) and QAM (Quadrature Amplitude Modulation)

Bluetooth HC-05

Bluetooth HC-05 is a Serial Port Protocol Bluetooth module, which uses serial communication (UART) which allows a very easy interface with the controller or with a PC [15]. This module has two modes: order response work and automatic connection work mode [16].

In the automatic connection mode, it will use the default way to transmit data automatically. In the order-response work mode, the command AT can be sent by the user to the module to set the control parameters. The switch between those modes can be done by controlling the module PIN – PIO11 input level.

This module can be used in multiple applications, like wireless communication between two microcontrollers, communication with PCs, data logging application, wireless robot, or home automation [20].

The default baud rate is 9600, with the default communication as “slave” and the default mode as the automatic connection mode. The typical sensitivity is -80dBm and the transmit power is up to +4dBm. As hardware specification, the HC-05 module has a power operation voltage between 4V and 6V, operating current of 30mA, with an integrated antenna and edge connector. The module follows the IEEE 802.15.1 standardized protocol, using FHSS (Frequency-Hopping Spread Spectrum) The module security is based on a password, which has the default value as “1234” or “0000”.

Pin description

The HC-05 Bluetooth module is equipped with 6 pins: ENABLE, VCC, GND, TXD&RDX, STATE and button switch, one led, and one button.

- Enable pin is used to toggle between automatic connection mode and order response mode (AT). To enter connection mode, the pin value should be set to low. The default value of the pin is low.
- VCC pin is used to supply the module. This pin should be connected to +5V voltage.
- GND pin is used to connect the module to the system ground.
- TX transmitter pin is used to transmit serial data outputted by the Bluetooth module.
- RX receiver pin is used to receive serial data broadcasted via Bluetooth module.\

- The state pin is connected to an onboard LED and is regularly used to check if Bluetooth is working properly.
- LED indicates the Bluetooth status and has 3 possible functionalities:
    - Blink once in 2 seconds which means that the module is in “command mode”.
    - Repeated blinking which means that the module Is waiting for a connection in “connection mode”.
    - Blink twice in 1 second which means that the module is connected successfully in “connection mode”.
- The button controls the enable pin, to switch from “connection mode” to “AT mode”. [19]

3.2.4 Batteries

Since Arduino is offered only 5V as output voltage, three LiPo batteries are used to supply the whole circuit, including Arduino. Those 3 LiPo batteries offer an 11.1 supply voltage

Fig. 27 LiPo batteries [36]

3.2.5 Arduino MEGA

Arduino MEGA is a microcontroller from the ATmega family, based on ATmega2560 microchip, with a lot of pins and functionalities [29], who is one of the most popular development board, including small project but also big project, all due to its high capability.

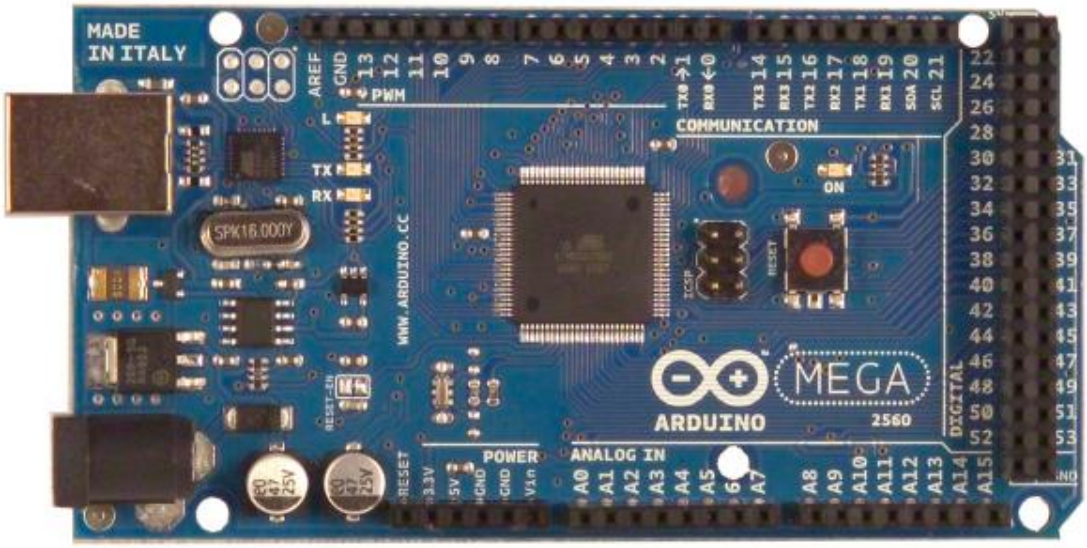


Fig. 28 Arduino MEGA [29]

Arduino Mega has a lot of features to offer, which include a lot of pins (analog and digital), multiple hardware serial ports, and a 16 MHz crystal oscillator. To all these features is added a flash memory of 256 KB for storing the code, from which 8 KB are used by the bootloader, an SRAM of 8KB, and an EEPROM of 4 KB.

Arduino Mega pinout

Arduino Mega inputs and outputs can be categorized in the following pins: [29]

- 54 Digital pins – which can be used to read/write digital signals.
- 16 Analog pins – which can be used to read/write analog signals.
- AREF – which can be used to change the reference voltage for the analog pins.
- Reset – which can be used to reset the microcontroller.

Some of the digital pins can have some extra functions:

- 14 PWM pins - which can be used to generate PWM signals.
- 4 Hardware serial ports – which could be used to transmit and receive TTL serial data. Serial0 of the microcontroller is also connected to the corresponding pins of the ATmega USB-to-TTL Serial chip.

- External interrupts – which could be used to trigger an interrupt on a rising or falling edge, a low value, or a change in value on the corresponding pin.
- SPI – which can be used for SPI communication.
- LED – which can be used to trigger the build-in led on the controller.
- I2C – which can be used for I2C communication.

Analog pins provide 10 bits of resolution and by default the pins measure 5V. As an exception, this value can be changed using the AREF pin and analogReference() function [30].

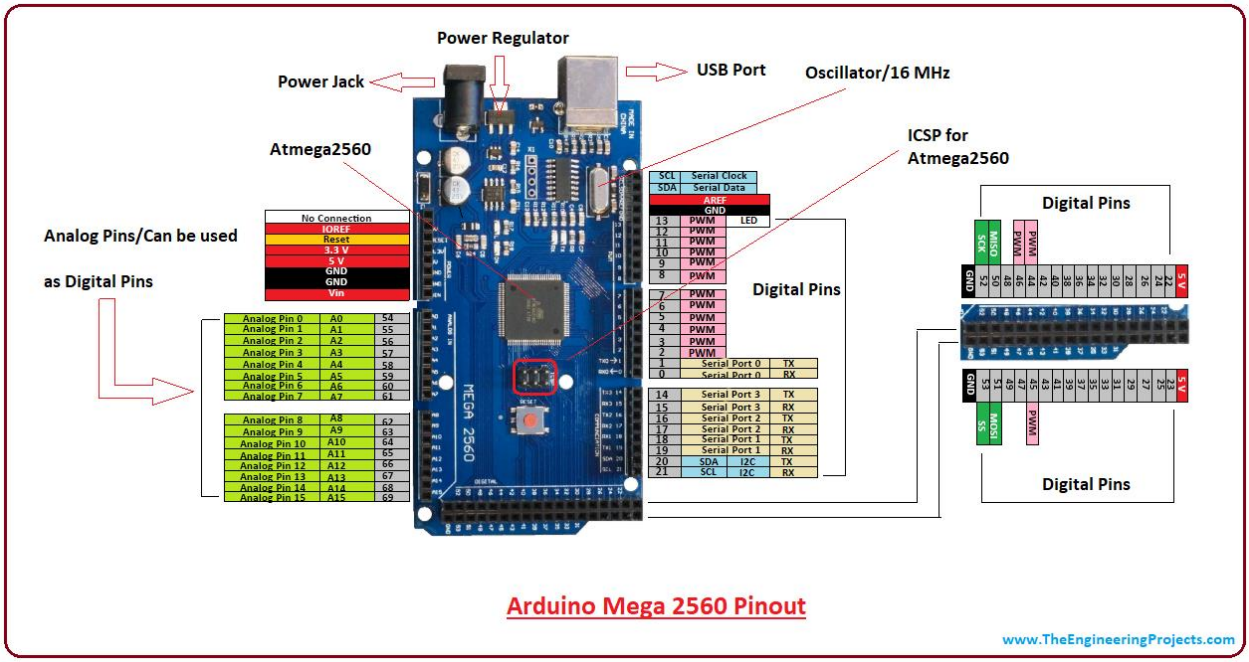


Fig. 29 Arduino MEGA pinout [29]

Arduino Mega specification [29]:

Table 10 Arduino Mega specification [29]

Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA

Arduino Mega programming

Arduino Mega can be programmed using the Arduino IDE(Integrated Development Environment) software. The chip on Arduino Mega came with a pre burned bootloader, which will allow the user to upload the software on the board without using an external hardware programmer. The communication uses the original STK500 protocol [30].

Arduino IDE is a cross-platform application, written in Java and has its origins in a development tool called “Processing” and project “Wiring”. This application uses a simplified version of C++ language, which increases the easiness of programming but still kipping the possibility to develop large and complex applications.

This IDE uses the concept of “Sketches”, being a standard place to save software code. The basic structure of an application includes two parts: the setup part, which is called only once, in the initialization phase, where all pins and communication must be initialized, and the second part which is executed repeatedly until the Arduino board voltage is disabled.

Fig. 30 Arduino IDE

3.2.6 Servo motors

Servo motors are electrical devices that can rotate or move pars of a machine with high efficiency and great precision. “A servo motor is a rotary actuator that allows for precise control of angular position”[37]. The servo motor used for this project is SG 90, which is a small servo motor that can rotate approximately 180 degrees.

Fig. 31 SG-90 Servo motor [37]

3.2.7 Basic components

In order to assemble all components and connect them, several basic components were used, like button, breadboard, logic level converter, lidar support, etc.

- Button - the main purpose of the button is to enable or disable the supply voltage for the whole circuit.
- Breadboard – it connects all components using wires.
- Logic level converter – it is used to convert the 5V voltage from Arduino to 3.3V for UART pins.

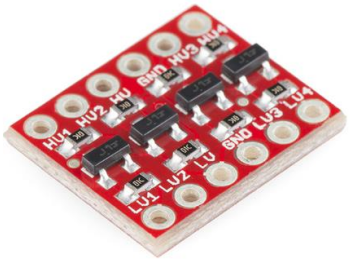


Fig. 32 Logic level converter [38]

- Lidar support – this is a support for a camera, and it adapted for the lidar sensor; this also includes 2 special spots for servo motors.

Fig. 33 Lidar support [39]

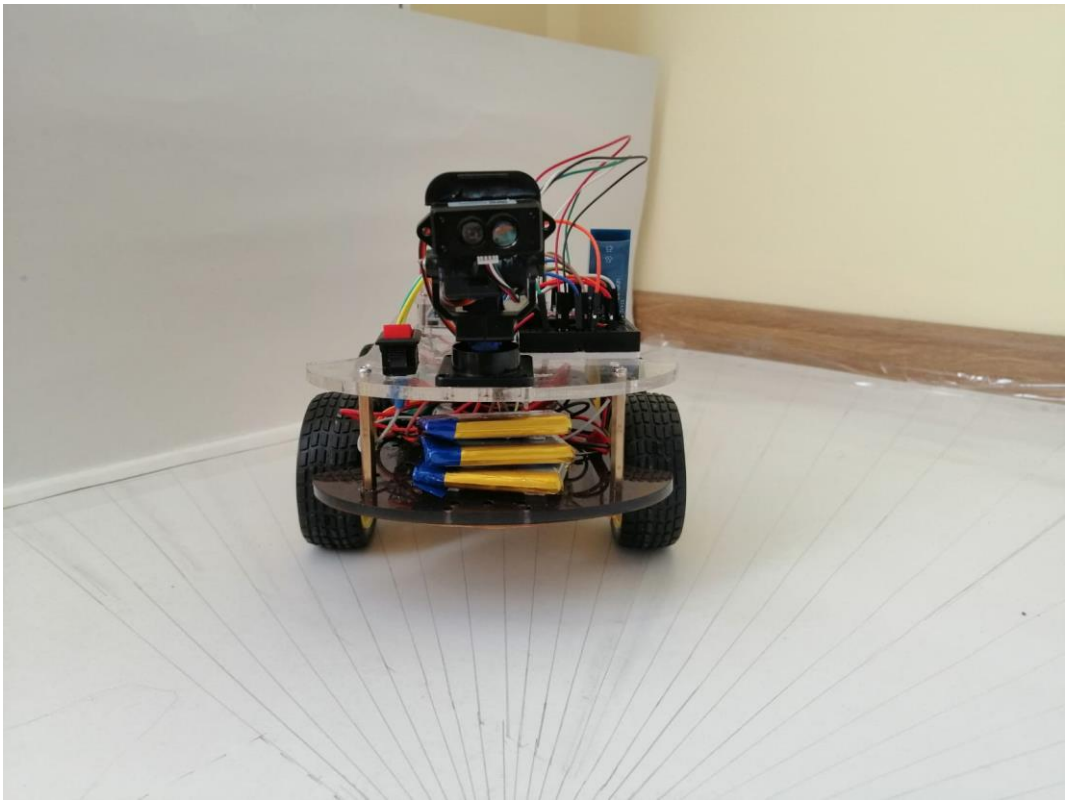


Fig. 34 Robot front view

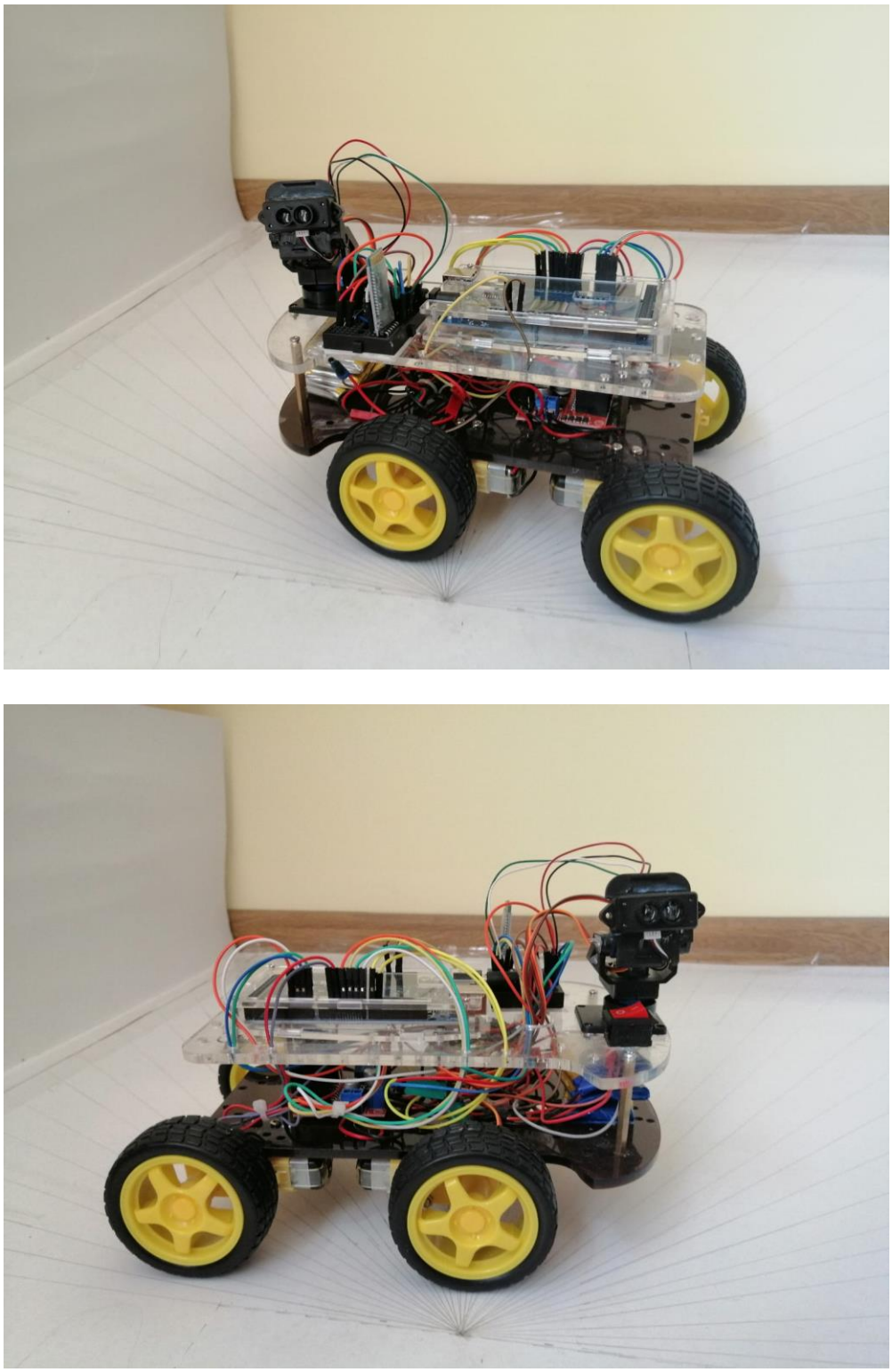


Fig. 35 Robot side view

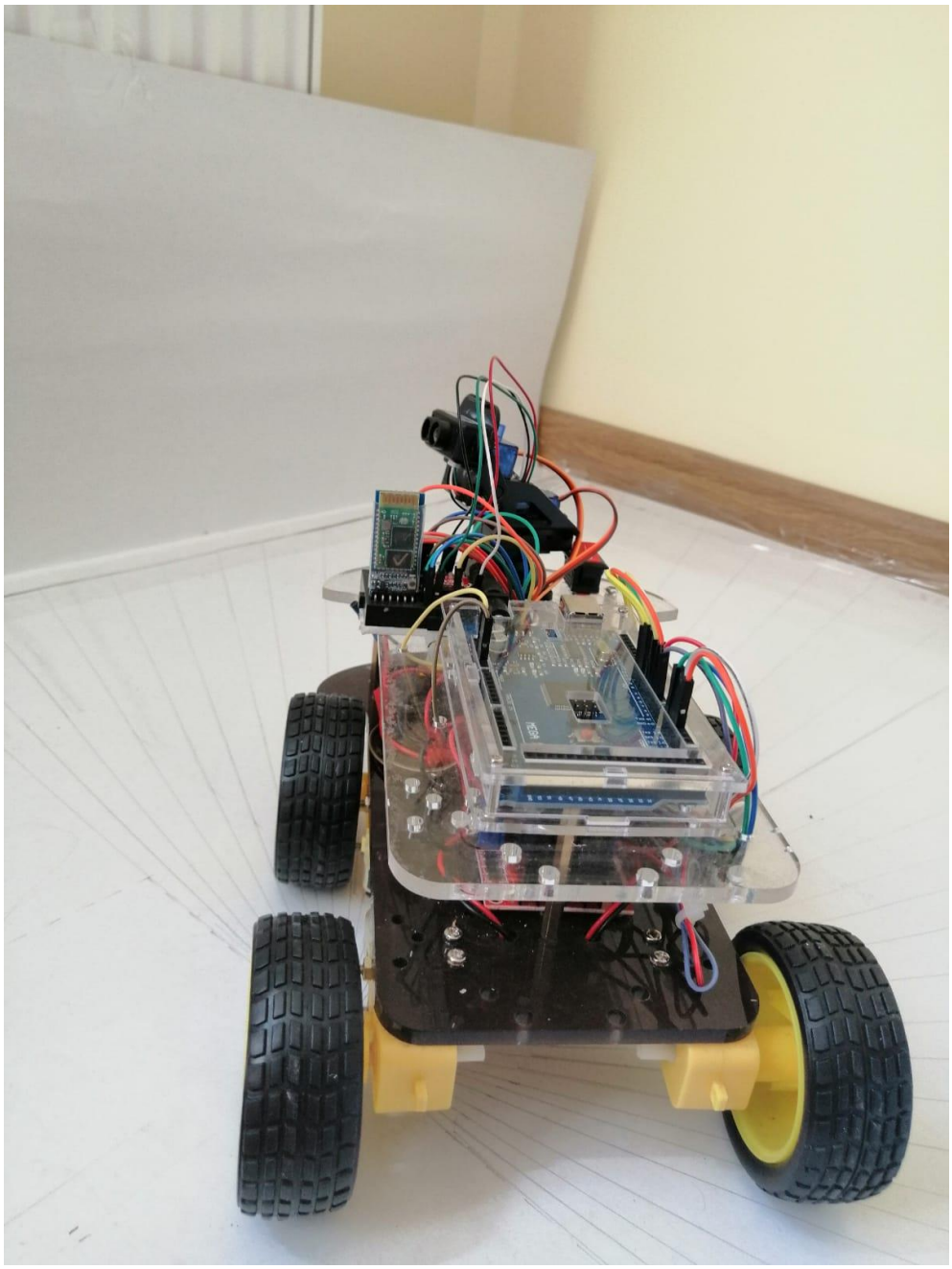


Fig. 36 Robot back view

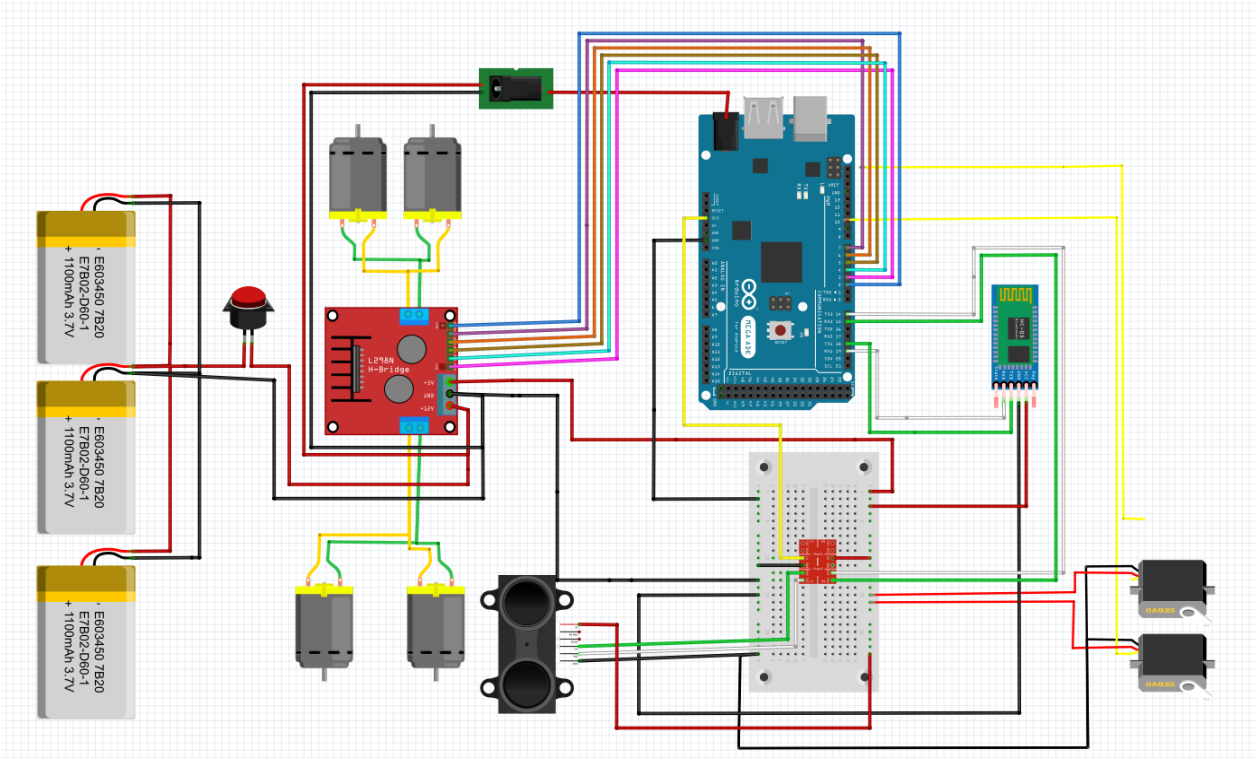


Fig. 37 Robot hardware connection

3.3 Software implementation

3.3.1 MATLAB

Matlab is "a high-level language programming environment, with an interactive view and which can be used for numerical computation visualization and programming" [32]. MATLAB, short from MATrix LABoratory [33] is designed for quick and easy calculation, data analysis, develop algorithms, and create models and applications. Matlab was originally written to have easy access to matrices and arrays, developed by the LINPACK – linear system package and EISPACK – eigensystem package projects. Still, Matlab has become a very used programming tool, because of its modern environment, which includes sophisticated data structures, debugging tools, and build-in editing and also support OOP – object-oriented programming.

The big advantage of MATLAB, in comparison with C language or other conventional computer languages, is the ability to solve technical problems. The basic element which makes Matlab better than the conventional language is the ability to use an array without specifying its dimension.

The Matlab system can be divided into five main parts[33]:

- *Development environment* – this is the graphical user interfaces, which is divided into multiple sub-windows:
    - The command windows
    - The workspace
    - The current directory
    - The editor windows
    - Browser – is used for viewing help

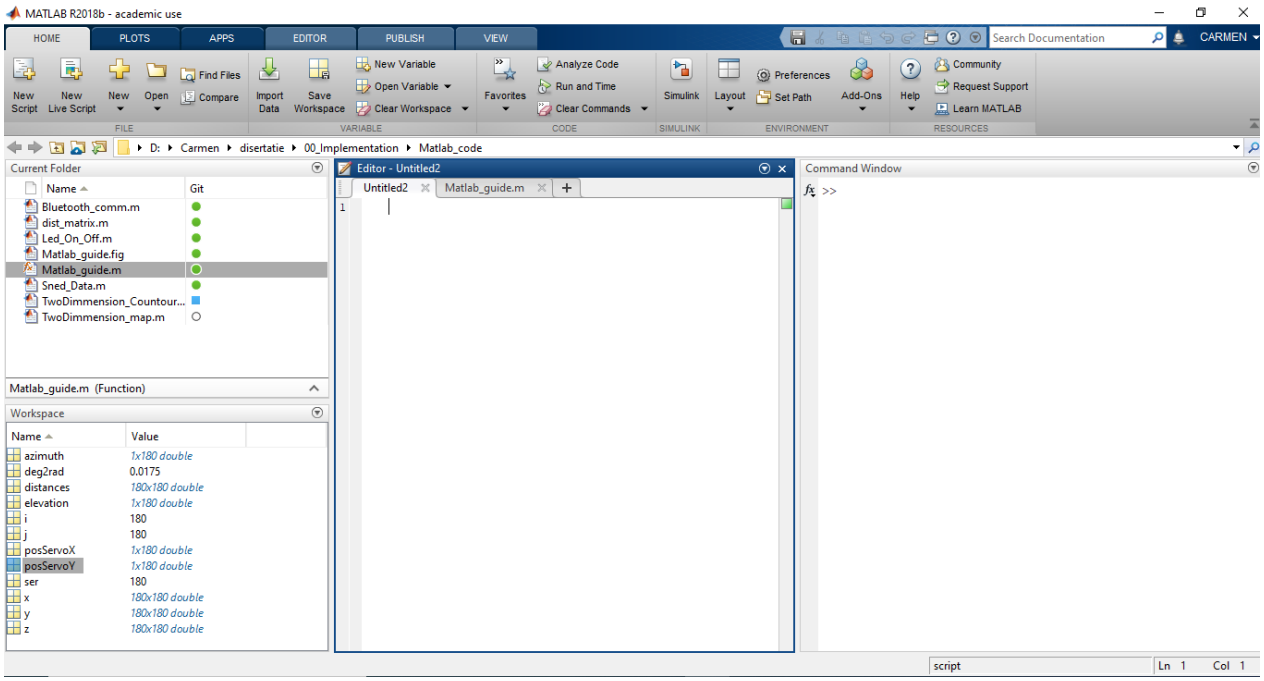


Fig. 38 Matlab interface

- *Matlab Mathematical function library* – this includes elementary mathematic functions, like sum, sine, cosine but also contains complex arithmetic, like matrix inverse, Bessel functions, Fourier transforms, and so on.
- *Matlab language* – this includes the matrix/array language, input/output features, functions, data structures, and object-oriented programming features.
- *Graphics* – this is used for displaying vectors and matrices as graphs, as well as other operations with graphs, like annotating and printing.
- Matlab external interfaces/API – this is a library that allows the user to combine C programs with Matlab programs. These facilitate the interaction between Matlab and C, by calling routine from Matlab.

3.3.2 Application block diagram

For creating the application, the next block diagram is considered:

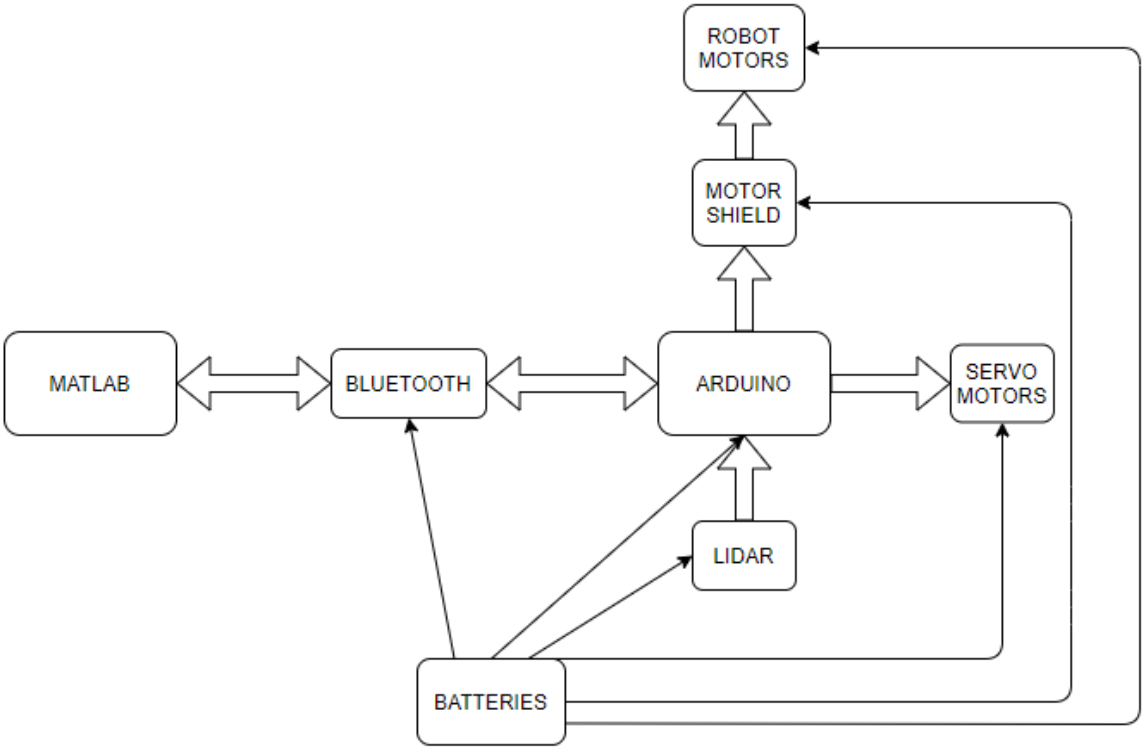


Fig. 39 Application block diagram

The components presented in the diagram are:

- Matlab – which represents the master of the application.
- Arduino – which represents the slave of the application.
- Bluetooth –is responsible for creating the connection between Matlab ad Arduino.
- Lidar sensor – reads information about the environment.
- Servo motors – controls the lidar position.
- Motor shield and robot motors – move the robot to a specific point.
- Batteries – supplies the whole system.

3.3.3 Software application

Software application respects the block diagram presented in fig. 30 and can be divided into the next parts: Matlab application and Arduino application.

Matlab application

Matlab application is created in "Matlab guide", which allows the creation of a graphic application, with a simple user interface.

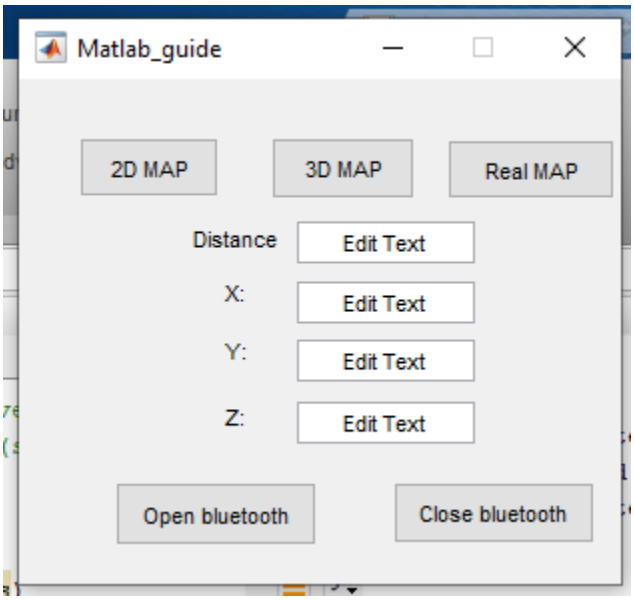


Fig. 40 Matlab guide

This application is responsible for sending a request from Matlab to Arduino, to perform some operations like scanning a two-dimensional map, scanning a three-dimensional map, or moving the robot. This kind of guide application uses "callbacks", which are similar to functions in C or C++ language.

The first callback of the application represents the initialization of the interface, by creating the margins of the interface, and also initialize the buttons, labels, and text boxes. This function is generated by Matlab when the interface is created but also can be modified based on the user needs.

The next callback will include the initialization of the used variables, but also the initialization timer and Bluetooth object. The purpose of the timer is to call Bluetooth callback, or function, at a period of 100 ms, with a fixed rated and an infinite number of function calls. The

Bluetooth object creates a channel for Bluetooth communication, between Matlab and HC-05 device. For creating the Bluetooth object it is used the Bluetooth toolbox provided by Matlab.

Source code for this sequence is presented next:

```
timer_Bluetooth = timer;
timer_Bluetooth.Period = 0.1;
timer_Bluetooth.ExecutionMode = 'fixedRate';
timer_Bluetooth.TasksToExecute = Inf;


bluetooth = Bluetooth('HC-05',1);
try
      fopen(bluetooth);
      set(bluetooth, 'TimeOut', 1);
      disp('Bluetooth opened succesfully');
catch e
      disp('Bluetooth detected an error and could not start');
end
```

If the creation of Bluetooth did not have any problems or issues, the application will continue with its exaction, otherwise, it will return an error message and will exit. After the initialization is done, the application will wait for a request from the user, which can choose from several buttons:

- "2D Map" – which will create a two-dimension map.
- "3D Map" – which will create a three-dimension map.
- "Real Map" – which will create a map based on the real environment measurements.
- "Open Bluetooth" – which will open the Bluetooth channel.
- "Close Bluetooth" – which will close the Bluetooth channel.

The buttons for opening and closing the Bluetooth channel can be used if the Bluetooth channel had some problems or errors.

For creating the two-dimension map, the first step is to start the timer, which will call the function responsible for reading data from Arduino. The source code for this is:

```
timer_Bluetooth.Timerfcn = @(~,~)Read_data_2D(bluetooth);
start(timer_Bluetooth);
```

After the timer will start, it will call the “Read_data_2D” at a period of 100 ms. This function will send the request to Arduino, and it will wait for the response. After the response is received, Matlab will make some computation with the data and will extract from the received string the distance values and the servo motor position, and will save them in an array.

The code sequence:

```
fprintf(Bluetooth_device, '2D'); %request new value from Arduino
received = fgetl(Bluetooth_device) %read the requested data
w = warning('query','last');
id = w.identifier;
warning('off',id)
if received ~= 0
      disp('Received message')
      distance = extractBetween(received,'D','X')
      distance = str2double(distance)   ;
      x_position = extractBetween(received,'X','D')
      x_position = str2double(x_position);
      x_size = numel(x_position)+ last_size;
      resetable_index = 1; %index for distance array
      for index=last_size:x_size
            if resetable_index >= numel(distance)
                  resetable_index = 1;
            else
                  Matrix_2D(index) = distance(resetable_index);
                  resetable_index = resetable_index+1
            end
      end
      last_size = x_size;
      Matrix_2D %display array
      if x_position(resetable_index-1) >= 179
            disp('Stop timer')
            stop(timer_Bluetooth); %Stop timer after all data are received
      end
end
```

That “Matrix_2D” will be used for computation of the ‘x’ and ‘y’ coordinates, in the previous function. For those, there are used trigonometric function: sinus and cosinus and azimuth method presented in chapter one.

```
azimuth = zeros(1,180);
x = zeros(1,180); %x coordinates
y = zeros(1,180); % y coordinates
deg2rad = pi/180;
for j=1:179 % servo x
   azimuth(j) =  posServoX(j)*deg2rad;
   x(j) = Matrix_2D(j)*cos(azimuth(j));
   y(j) = Matrix_2D(j)*sin(azimuth(j));
end
```

After the computation of coordinates is ready, the map with those coordinates will be created, and data will be saved in a “*.txt” file.

```
figure(1)
plot(x,y)
xlabel('x')
ylabel('y')
title("2D Map")
grid on


fid = fopen('distance.txt', 'w')
if fid == -1
    error('cannot create file')
end
fprintf(fid,'%s %s\n','Distance', 'Servo');
for i=1:179
    fprintf(fid,'%g\t', Matrix_2D(i));
    fprintf(fid,'%g\n', posServoX(i));
end
fclose(fid);
```

For creating the three-dimension map, steps are very similar to the creation of the 2D map, meaning the first step will be to start the timer and request data from Arduino, but the called function will be “ Read_data_3D”, which will have, besides the distance and servo motor position array, another array for the second servo motor.

```
fprintf(Bluetooth_device, '3D'); %request new value from Arduino
received = fgetl(Bluetooth_device) %read the requested data
w = warning('query','last');
id = w.identifier;
warning('off',id)

if received ~= 0
    disp('Received message')
    distance = extractBetween(received,'D','X');
    distance = str2double(distance);
    x_position = extractBetween(received,'X','Y')
    x_position = str2double(x_position)
    y_position = extractBetween(received,'Y','D')
    y_position = str2double(y_position)
    y_size= numel(x_position)+ last_size_y;
    resetable_index = 1; %index for distance array
    for index=last_size_y:y_size
        if resetable_index >= numel(distance)
             resetable_index = 1;
        else
            Matrix_3D(x_position+1,(151-y_position(resetable_index))) =
distance(resetable_index);
            resetable_index = resetable_index+1
        end
    end
end
```

The “Matrix_3D” will be used to calculate ‘x’, ‘y’, and ‘z’ coordinates using the same trigonometric function and also azimuth and elevation method.

```
for i=1:180 % servo x
    for j=1:150 % servo y
       azimuth(i) =  posServoX(i)*deg2rad;
       elevation(i) = posServoY(i)*deg2rad;
       x(i,j) = Matrix_3D(i,j)*sin(elevation(i))*cos(azimuth(i));
       y(i,j) = Matrix_3D(i,j)*sin(elevation(i))*sin(azimuth(i));
       z(i,j) = Matrix_3D(i,j)*cos(elevation(i));
    end
end
```

The “Real MAP” button will calculate ‘x’ and ‘y’ coordinates and will create the map, based on the real measurements of the environment, measurements made using a tape measure, and protractor. The sequence code is similar to code from “2D MAP” unless the distance value array and servo motor position array are given in code, and not read from Bluetooth device. Since the Bluetooth device is not used in this case, the timer is not started.

3.3.3.1 Arduino application

Arduino application is responsible for controlling the lidar sensor, using servo-motor, for controlling the robot motors, and for waiting for a request from Matlab. This application uses four states for the robot:

- "PAUSE" – this state is used when the robot is waiting for a request from Matlab.
- "MOVE" – this state is used when the robot received the move command from Matlab.
- "TWO_D" – this state is used when the robot received the command for scanning the room in two dimensions.
- "THREE_D" – this state is used when the robot received the command for scanning the room in three dimensions.

The first step in the Arduino application represents the initialization of the hardware components: servo motors, serial communication channel, and robot motors. The servo motor initialization is represented by the attachment of the servo motor to the corresponding PWM channel and setting the initial position of those.

```
horizontal_Servo.attach(X_Serv_Pin);
vertical_Servo.attach(Y_Serv_Pin);
horizontal_Servo.write(posServoX); //starts from middle position
vertical_Servo.write(posServoY); //starts from middle position
```

"*X_Serv_Pin"* and "*Y_Serv_Pin"* represent the Arduino pin where the servo motors are connected. The schematic for this code is shown in the following image:

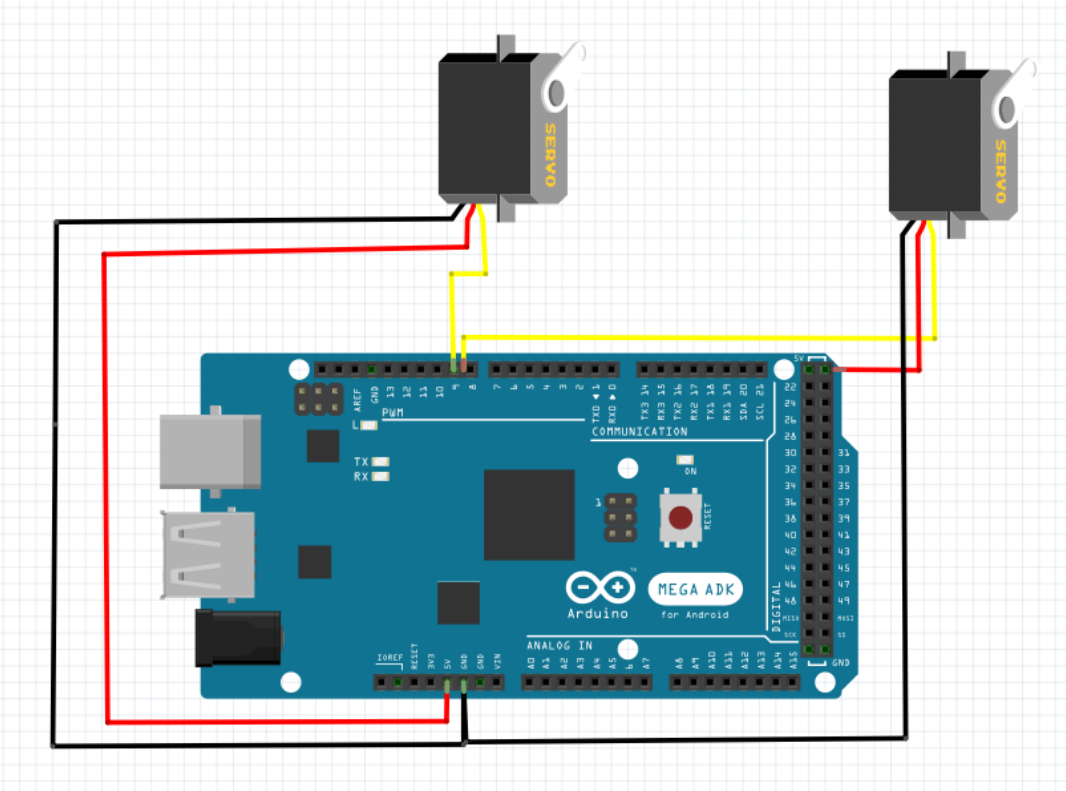


Fig. 41 Servo motors connection to Arduino

The serial communication (UART) is used for connection with the lidar sensor, Bluetooth device, and USB connection to the laptop. The USB connection is used only for testing. In this step, the serial connections are stared at the specified baud rate.

```
Serial.begin(115200); // baud rate of USB serial port

Serial2.begin(9600); // baud rate of Bluetooth

Serial3.begin(115200);  //baud rate of LiDAR sensor
```

The hardware connection of the Lidar sensor and Bluetooth device can be seen in the following images, where it can be seen that the components are connected to the hardware serial of Arduino Mega.

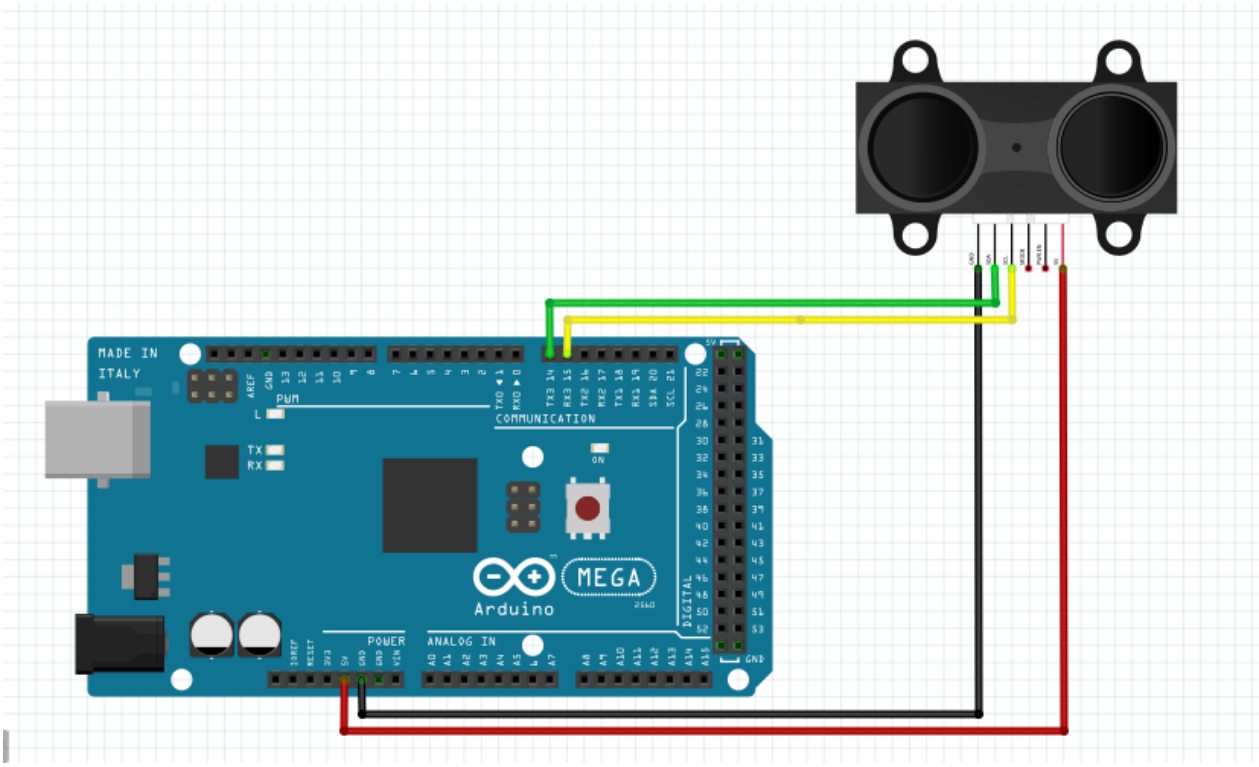


Fig. 42 Lidar sensor connection to Arduino

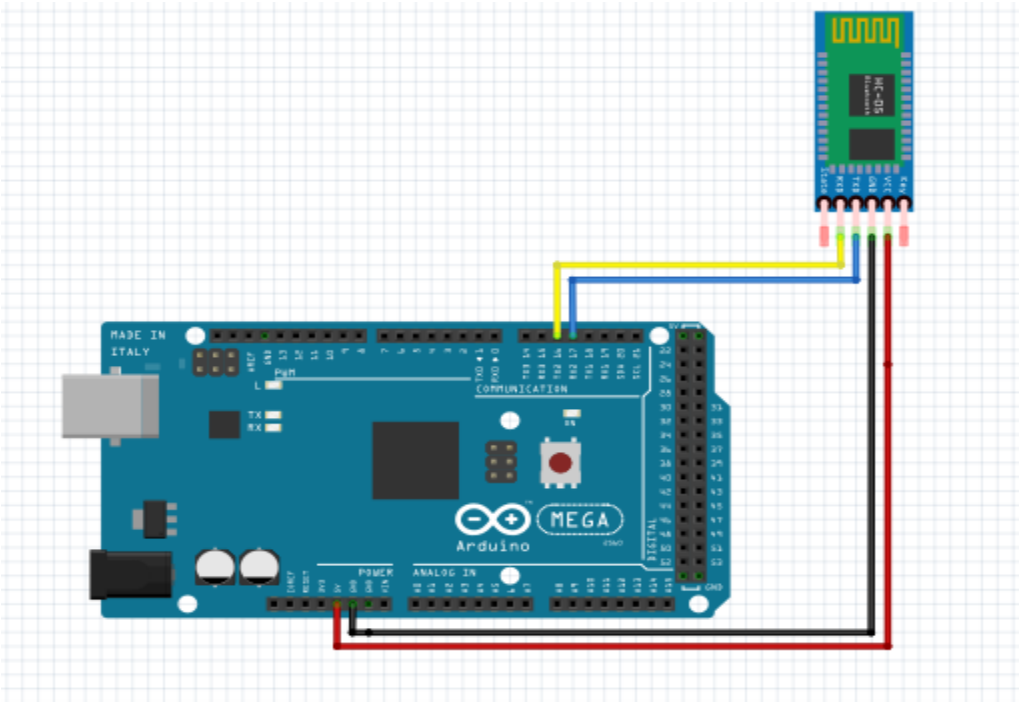


Fig. 43 HC-05 connection to Arduino

Also in the initialization step, the state of the robot is set to “PAUSE” state, until a new request will come from Matlab. Further, Arduino will wait for a request from Bluetooth device, by calling the “Read_bluetooth” function, and will set the corresponding state based on the received message. After the response from Matlab is received, the Arduino will continue its execution with 2D scanning, 3D scanning, or moving, depending on the data received from Matlab. The following flowchart will present Bluetooth implementation.

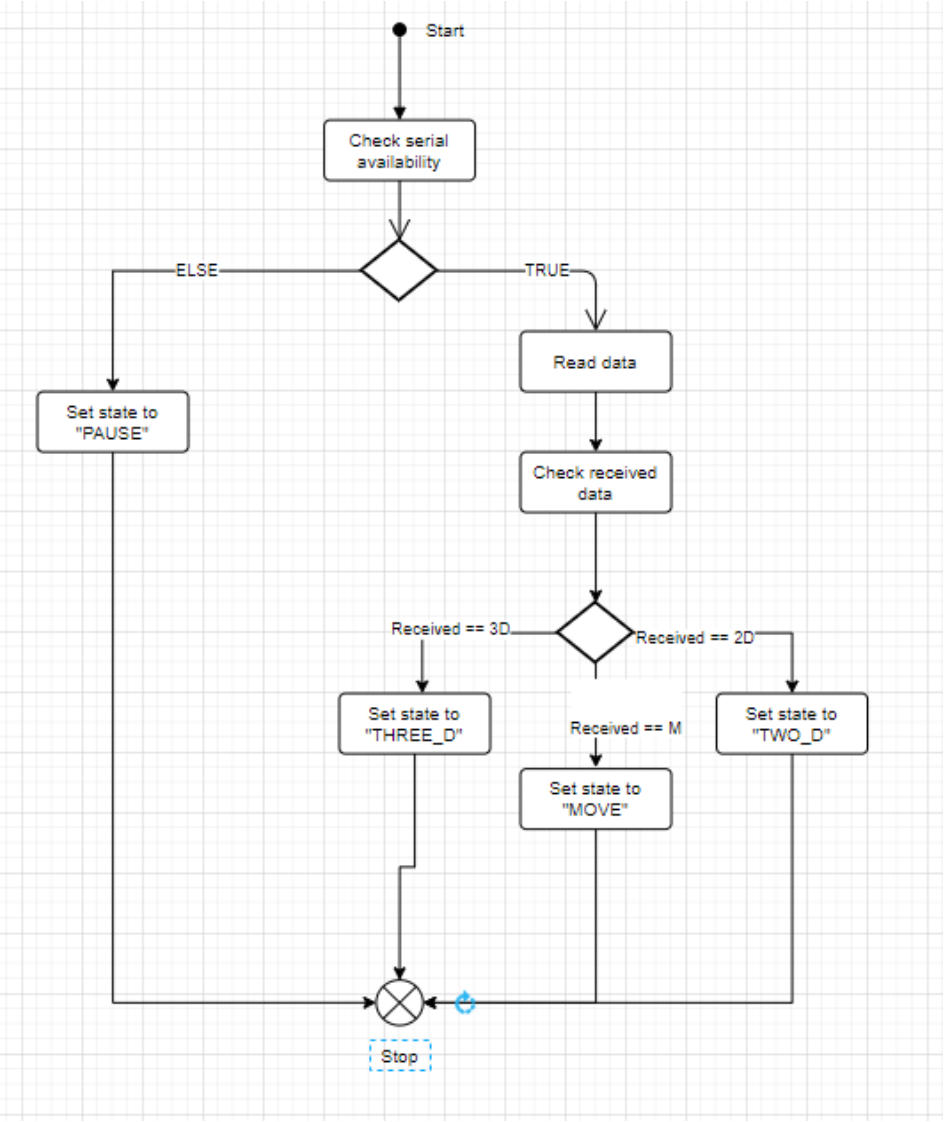


Fig. 44 HC-05 read function flowchart

When Arduino receives a message or request from Matlab, it will start to process it and continue its workflow. Let's assume that the received message from Matlab is requesting the creation of a two-dimension map. In this case, the Arduino will follow the flowchart presented in figure 45.

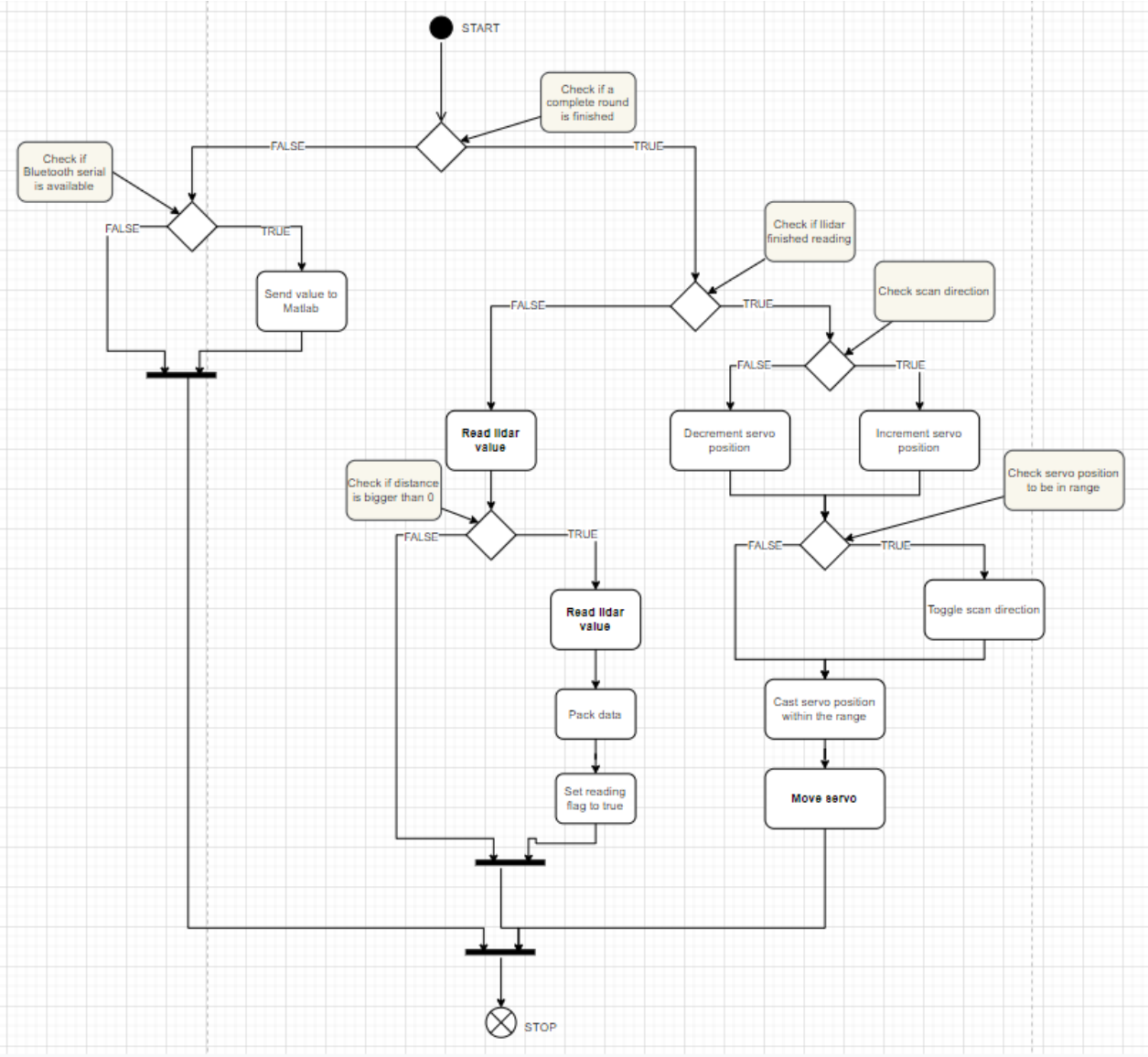


Fig. 45 2D scanning flowchart

2D scanning starts with a security check phase, in order to detect if the servo motor completed a full scanning (from 0 to 180°), to detect if the values were sent to Matlab and also to detect if the lidar finished reading. In the initialization step, all these variables are set so that the servo motor can start moving (the full scan flag is set to '*false'*, the send value flag is set to '*false'*,

and the lidar reading flag is set to '*true'*). After all these checks, the next step will be to check the scanning direction (left or right). This is made because the servo motor can be initialized in any direction, and will start scanning from that direction. The value of the servo motor angle will be incremented according to its direction, and there is also a security check, to keep the servo motor angle value withing its range, to avoid servo damages. This value will be used to move the servo motor, to perform the reading to that position. The code for this section is presented below:

```
if (posServoX != lastPosServoX)
  {
    horizontal_Servo.write(posServoX);
    Serial.print("Angle X: ");
    Serial.println(posServoX);
    lastPosServoX = posServoX;
    movement = true;
    read_flag = false;
  }
```

If the servo motor position is different from the previous position, the servo will move to the new position and the flag for lidar reading will be reset so that the lidar can start reading. The flowchart for lidar reading is shown in figure 46.

In this function, the first step is to check the availability of the serial communication. The values read from the Lidar sensor are packaged in 9 bytes, so we need to create an array to save data from serial wire. After the data are read, the first and second values from the array will be compared with the frame header, to detect if the values are really from the Lidar. If the header is correct, the CRC of the value will be calculated (all remaining bytes, except for the last one) will be summed together and will be compared with the last value from the array. If there are differences between those two, the value read is not valid and the software will try a new reading. The last step of this function is to calculate the distance and strength values.

The code for this function is presented next:

```
static char index = 0; //index is used for accessing Lidar bytes
  char j = 0;
  int CRC = 0;
  static int rx[9];
  if(Serial3.available())
  {
    //Serial.println( "tfmini serial available" );
    receive[i] = Serial3.read();
    if(receive[0] != HEADER) {
      index = 0;
    } else if(index == 1 && receive[1] != HEADER) {
      index = 0;
    } else if(index == 8) {
      for(j = 0; j < 8; j++) { //j representes the number of bytes from Lidar
measurements, without CRC byte
        CRC += receive[j];
      }
      if(receive[8] == (CRC % 256)) {
        *distance = receive[2] + receive[3] * 256;
        *strength = receive[4] + receive[5] * 256;
      }
      index = 0;
    }
 else
    {
      index++;
    }
}
```

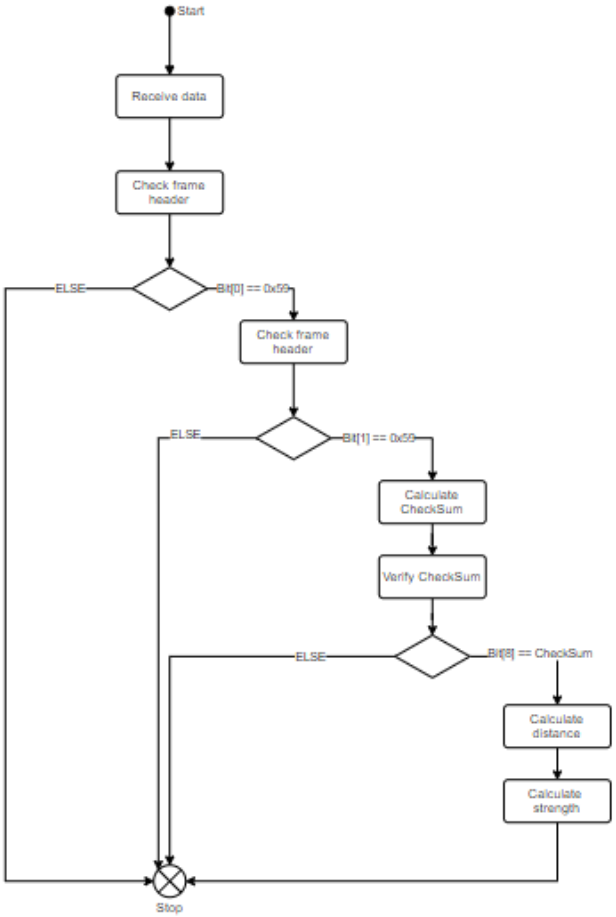


Fig. 46 Lidar reading flowchart

In the second scenario, Arduino will continue its execution with the three-dimensional map creation, of course, based on the input from Matlab. The implementation for this function is very similar to the previous one, except that here, both servo motors: horizontal and vertical are used.

The algorithm starts with the same security checks, only one extra, which will check if the algorithm has finished both scannings: on the x-axis and the y-axis. Arduino will continue as in the previous step until the vertical servo-motor will reach one of its limits. After that, the position of the horizontal servo motor is incremented, and the algorithms start over until the horizontal servo reached its maximum value. The complete flowchart of this algorithm can be seen in figure 46.

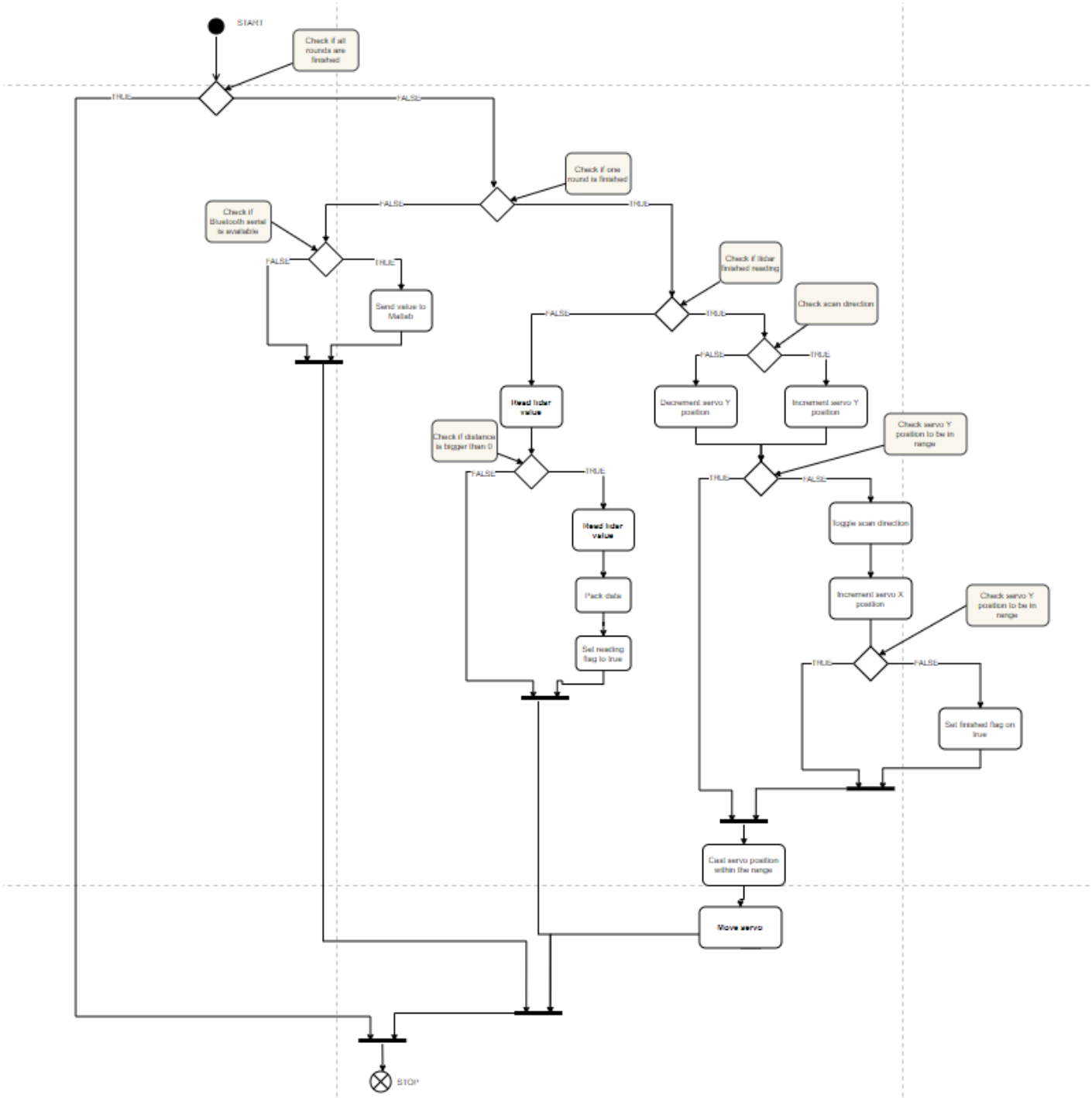


Fig. 47 3D scanning flowchart

4 EXPERIMENTAL RESULTS

4.1 Environment description

In order to create a proper environment, I have decided to create a model map made out of cardboard, having the dimensions presented in table 11, where the map was divided into 36 angle measurements, from 0 to 180 degrees, with a 5-degree step:

Fig. 48 Environment presentation

The position of the map was in a corner of a room, in order to be surrounded by two of the room walls. In this way, the cardboard model was designed only with one “artificial” wall, so that the lidar sensor would not take into consideration the object located in the room.

The main objective in the process of splitting the map into 36 sections was to compare the measurements made by the sensor with the actual values. In this way, we can have a better look at the results and see the performance of the sensor.

For a better understanding of sensor performance, I have decided to create both a two-dimension map and also three-dimension map, using the cardboard model as the map.

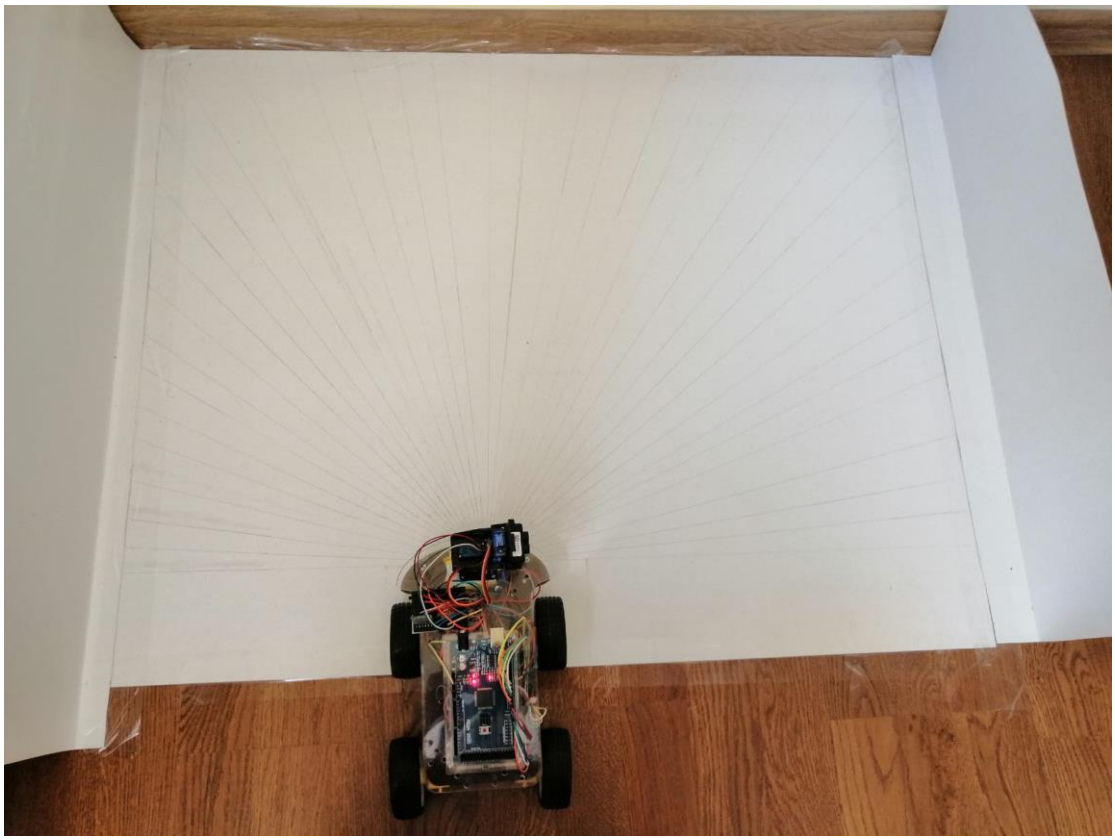


Fig. 49 Robot position on the map

4.2 2D Map

In this case, the measurements were made using the robot located in the middle of the cardboard x-axis, because the actual measurements were taken from that start point. In order to realize the measurements, only the horizontal servo motor was used, which was moving from left to right (or from 0 to 180 degrees with a step of 1 degree). The other servo motor from LiDAR support was positioned in its central point because it is not relevant for 2D measurement. After 5 sets of tests, the following table shows the measurement results.

The results of the measurements concluded with the 2D map created in Matlab, upon which we can see the real-time difference between the maps created using sensor values and map created using actual values.

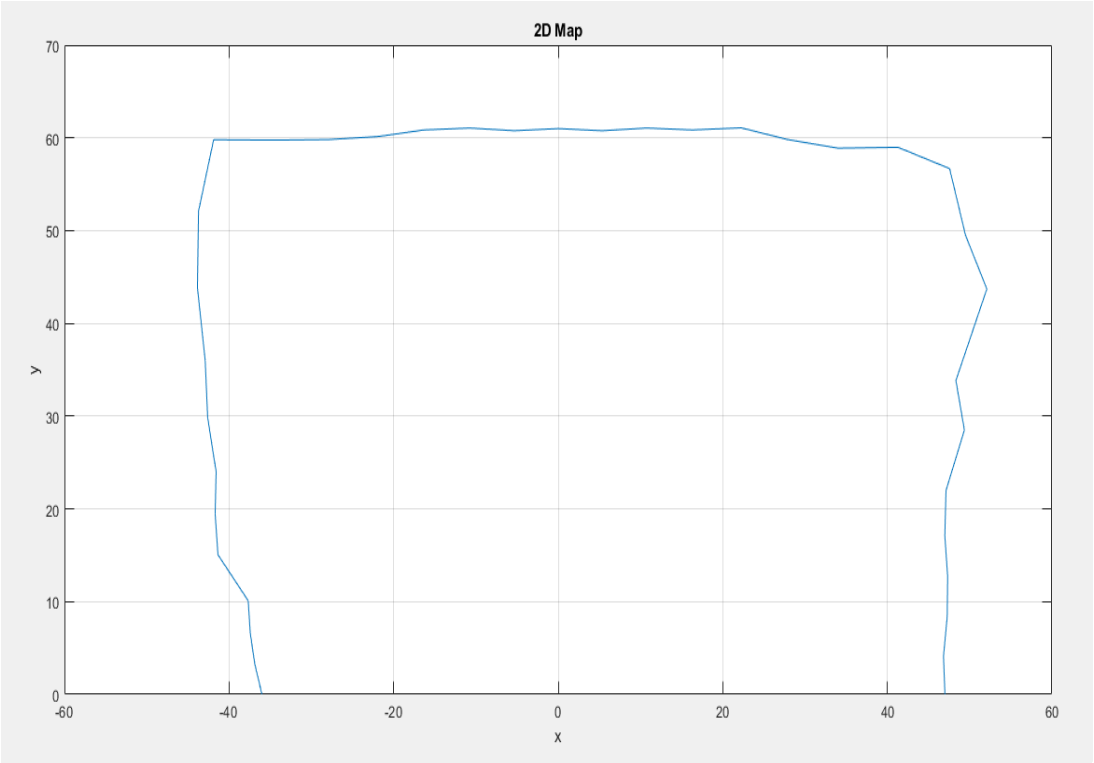
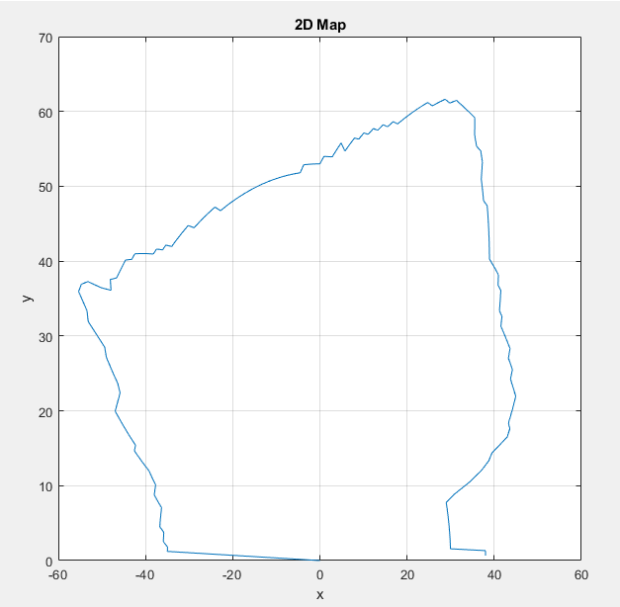
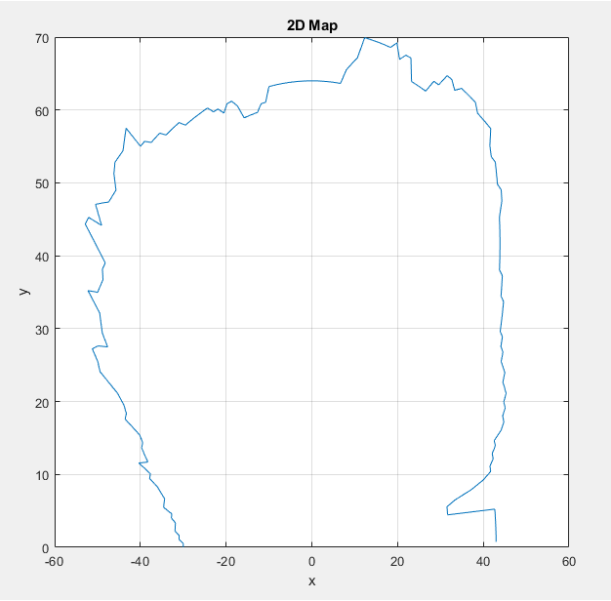


Fig. 50 2D map with actual measured values



(a) Test 1 measurement



(b) Test 2 measurement

Fig. 51 2D map with sensor measure values

Table 11 Measurement values

Servo motor position (°)	**Real distance (cm)**	**Test 1 (cm)**	**Test 2 (cm)**	**Test 3 (cm)**	**Test 4 (cm)**	**Test 5 (cm)**
0	47	43	43	43	45	44
15	49	44	44	46	43	47
30	57	53	53	53	53	53
45	70	62	62	65	67	63
60	68	74	72	74	73	72
75	63	61	70	67	56	65
90	61	52	64	63	52	61
105	63	50	61	61	51	62
120	69	51	66	66	51	66
135	62	59	67	67	60	67
150	48	54	54	54	59	61
165	39	30	30	37	37	30
180	36	30	30	37	30	30

After the measurements, I have decided to study the deviation between the actual measurements of the map and the measurements from the sensor, using the following formula.

$$Average\ error = \frac{Sensor\ measurement - Real\ measurement}{Real\ measurement} \cdot 100\ [\%] \quad (8)$$

As we can observe in the following table, the error is varying from 1% to 15%, depending on external factors as luminosity and reflection of the object, but also on internal factors such as the sensor precision. A more graphical view could be observed in figure

Table 12 Comparision between actual values and sensor values

Actual distance measurement (cm)	Average 10 Times Distance Measurements By Sensor (cm)	Error Average Distance Measurements by Sensor (%)	Standard Deviation of Distance Measurement by Sensor (cm)
47	43.6	7.234	3.4
49	44.9	8.367	4.1
57	53	6.315	3.6
70	63.6	9.142	6.4
68	72	5.882	4
63	64.2	1.904	1.2
61	58.4	4.262	2.6
63	58.9	6.507	4.1
69	59.4	13.913	9.6
62	63.4	2.258	1.4
48	55	14.583	7
39	34.6	11.282	4.4
36	32	11.111	4

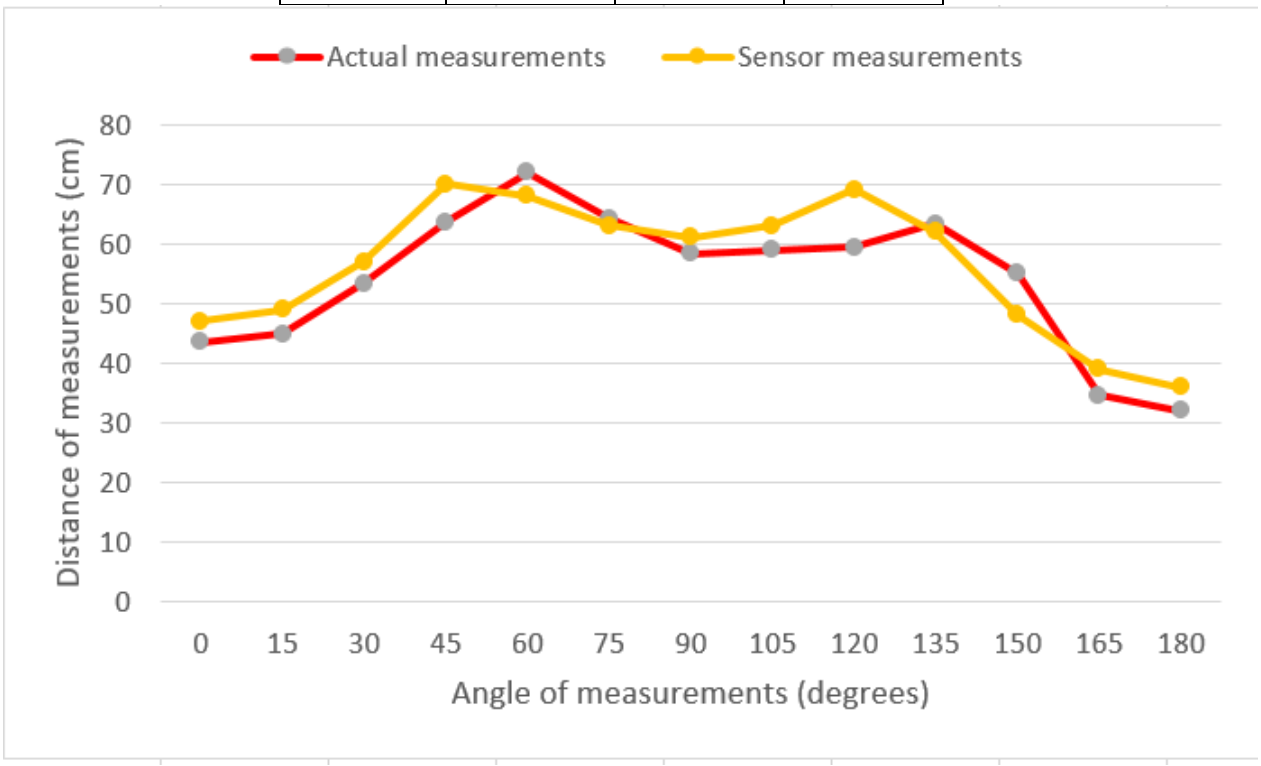


Fig.52 Comparison between actual measurements and sensor measurements

4.3 3D Map

The second case presented in this study is the realization of a three-dimension map, in the same environment, in order to create a more complex and complete picture of the environment. The measurements were made using the same approach as before, in the middle of the cardboard x-axis. In this scenario, both horizontal servo motor and also vertical servomotor were used, which were making movements from left to right (or from to 0 to 180 degree with a step of 1 degree) and also from top to bottom (or from 10 to 150 degree with a step of 1 degree). The 10 and 150-degree values were being used because of the servo motor support, which is not allowing the servo motor to move from minimum to maximum range.

In this particular case, no comparison between the sensor values and the actual values were being made, because it would have taken to much time to take the actual measurements of the environment (to create a real 3D measurement of the room, it would be necessary 140x180 measurement, where 140 means the measurement on the y-axis and 180 measurements on the x-axis).

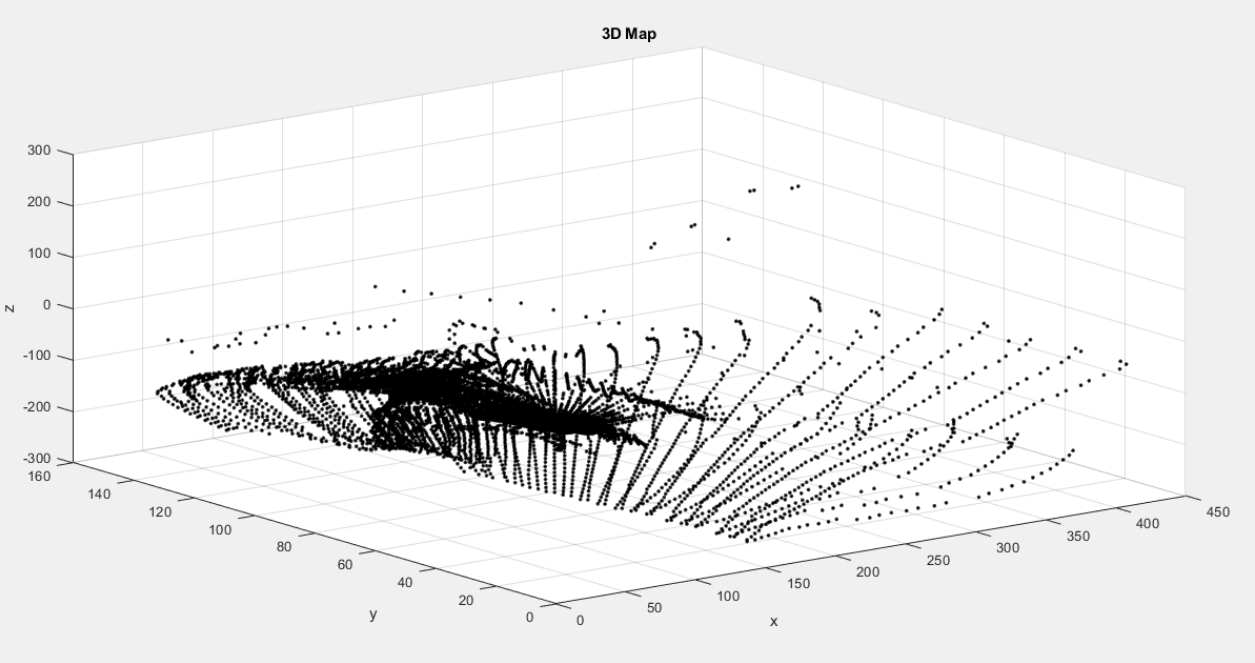


Fig. 53 3D Map

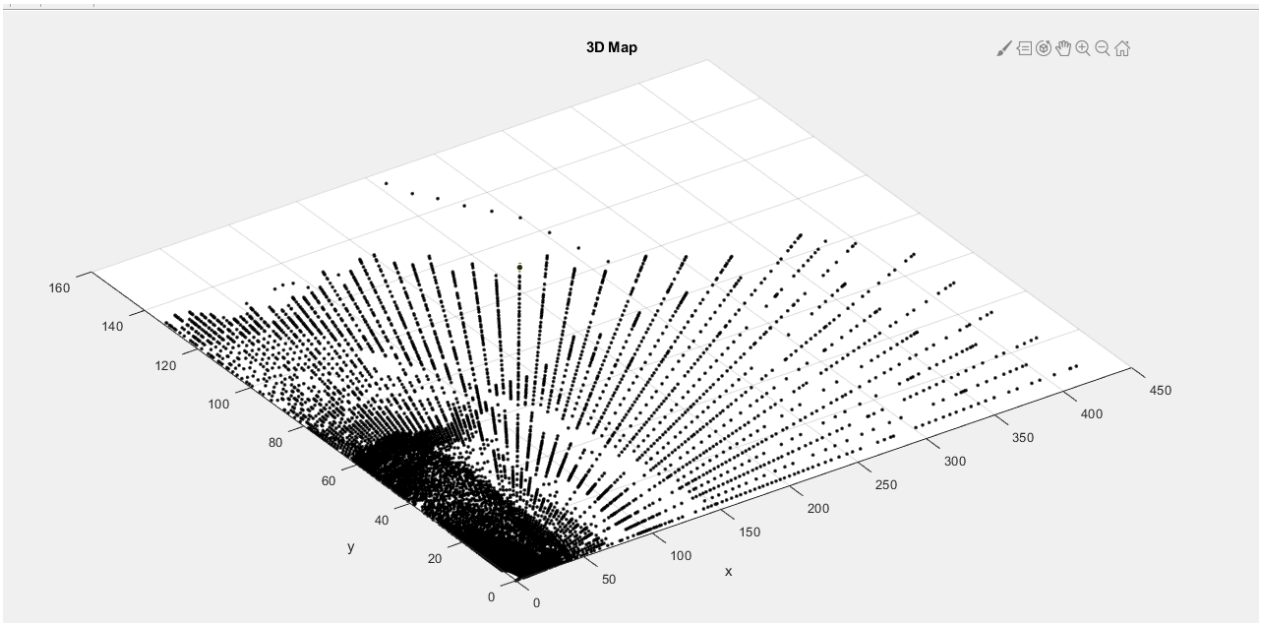


Fig 54 3D Map – up view

5 CONCLUSION

In my opinion, the robotics domain is gaining more and more usage in our lives, due to both technological progress and also the hardware structure upon which relies. This has a major impact on our current society, allowing it to improve our day to day life, by taking over some of the repetitive tasks.

One of the most interesting topics, regarding the robotics area, is the ability to adjust big projects, like robotic satellite or autonomous cars, and the benefit of their technology, which could be used successfully in small projects, such as autonomous vacuum cleaner or recognition agents.

A fundamental part of this kind of agent is the ability to interact with the surrounding environment. Over here we have endless options, including different types of sensors based on their functionality: ultrasonic sensors, infrared sensors, light sensors, etc.

Two of the most practical examples in this field, which are being used nowadays are autonomous vacuum cleaners, which have commercial use and also rescue agents, which have military use.

As a proof of concept, I decided to approach the localization and mapping problem presented in the previous examples, on an autonomous robot, using one of the most commonly used sensors in this field, the lidar sensor.

The approach was to create a 2D and also a 3D map, using two servo motors to control the lidar sensor on ‘x’ and ‘y’ axis. In order to achieve the approach, I used a combination of Arduino microcontroller and Matlab software.

The hardware components were being controller by the Arduino microcontroller and the processing computation was being made in Matlab software, due to its high performance.

The outcome of the experimental results were compared to a real map in order to test the practicality of the approach. As a finding, it was observed that the generated map was partially influenced by the surrounding factors (such as luminosity level, object material, and color,

contrast, etc) and also by the interferences between the other serial communication that was being used by the Bluetooth device.

Another observation made was the fact that the feedback from the servo motor is missing due to the low cost of the servo motor used, which will lead to an incorrect or incomplete map.

One of the most challenging problems was the fact that the Lidar sensor a low priced one, which had repercussions upon both the quality of the output and also upon integrated it with the Arduino microcontroller.

However, some feature improvements can be added, such as adding the possibility fo the robot to go to a specific point and also the improvement of the quality of the generated map, by adding the feedback of the servo motors, allowing us to known the precise angle.

As a final word, I believe that this kind of project is very useful to understand the localization and mapping problem in different environments. However, it is important to keep in mind that the quality of used sensors and microcontrollers are playing a crucial part.

6 REFERENCES

[1] - FAIRFIELD N. – *Localization, Mapping and Planning in 3D Environments,* 2009, The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213

[2] - GUOLAI J., LEI Y., SHAOKUN J., CHAORAN T., XINBO M., YONGSHENG O., *A Simultaneous Localization and Mapping (SLAM) Framework for 2.5D Map Building Based on Low-Cost LiDAR and Vision Fusion,* 2019, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

[3] - MADER B. – *Localization and Mapping with Autonomous Robots,* 2011, Radboud University Nijmegen

[4] - MAHRAMI M., ISLAM MD. N., KARIMI R., *Simultaneous Localization and Mapping: Issues and Approaches,* 2013, ISSN: 2047 – 3338

[5] - MARKOM M., ADOM A., TAN E., ABDUL SHUKOR S., ABDUL RAHIM N., *A mapping mobile robot using RP Lidar scanner*, 2015, School of Mechatronic Engineering,

Malaysia Perlis University

[6] - MAULANA I., RUSDINAR A., PRIRAMADHI R. - *Lidar Application for Mapping and Robot Navigation on Closed Environment,* 2018, Journal of Measurements, Electronics, Communications, and Systems, ISSN: 9772477798025

[7] - RAJMOHAN G. - O*bstacle detection and avoidance methods for autonomous mobile robot,* 2016, ISSN (Online): 2319-7064

[8] - *** *Robotic mapping*, Available from: https://en.wikipedia.org/wiki/Robotic_mapping {2}

[9] - *** *Robotic mapping: A survey*, Available from: http://robots.stanford.edu/papers/thrun.mapping-tr.pdfj

[10] - ***, *Robot navigation*, Available from: https://en.wikipedia.org/wiki/Robot_navigation

[11] - *** *What is Lidar? ,* Available from: https://oceanservice.noaa.gov/facts/lidar.html

[12] - ***, *100 Real-World Applications of LiDAR Technology,* Available from: https://levelfivesupplies.com/100-real-world-applications-of-lidar-technology/

[13] - ***, *Product manual of TFmini*, Available from: https://cdn-shop.adafruit.com/product-files/3978/3978_manual_SJ-PM-TFmini-T-01_A03ProductManual_EN.pdf

[14] -***, *Time of Flight principle,* Available from: https://www.terabee.com/time-of-flight-principle/#:~:text=Time%2Dof%2DFlight%20Principle,being%20reflected%20by%20an%20object

[15] - ***, *How does Bluetooth works*, Available from: https://www.elprocus.com/how-does-bluetooth-work/#:~:text=A%20Bluetooth%20technology%20is%20a,over%20short%20distance%20without%20wires.

[16] -***, *Bluetooth*, Available from: https://en.wikipedia.org/wiki/Bluetooth

[17] - ***, *Difference between Bluetooth and Wifi,* Available from: https://techdifferences.com/difference-between-bluetooth-and-wifi.html

[18] - ***, *HC-05 Bluetooth Module User Manual*, Available from: https://www.gme.cz/data/attachments/dsh.772-148.1.pdf

[19] - ***, AT command set, Available from: http://www.linotux.ch/arduino/HC-0305_serial_module_AT_commamd_set_201104_revised.pdf

[20] - ***, *HC-05 Bluetooth Module*, Available from*:* https://components101.com/wireless/hc-05-bluetooth-module

[21] - ***, *Introduction to SLAM*, Available from: http://www.arreverie.com/blogs/introduction-simultaneous-localisation-and-mapping/

[22] - ***, *Simultaneous localization and mapping*, Available from: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[23] -***, *Occupancy maps,* Available from: https://www.cs.cmu.edu/~16831-f14/notes/F14/16831_lecture06_agiri_dmcconac_kumarsha_nbhakta.pdf

[24] - ***, *Topological Map,* Available from: https://www.researchgate.net/figure/Topological-Mapping-a-Building-map-of-sections-A-with-robot-following-two-different_fig22_269573252

[25] - ***, *Concept map,* Available from: https://en.wikipedia.org/wiki/Concept_map

[26] - ***, *Obstacle detection*, Available from: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-31439-6_52

[27] - ***, *LiDAR: The Use of Light Detection and Ranging Technology*, Available from: https://www.lumitex.com/blog/lidar-technology

[28]- ***,*LIDAR map of New Orleans flooding caused by Hurricane Katrina, 3 September 2005*, Available from: http://www.esa.int/ESA_Multimedia/Images/2012/01/LIDAR_map_of_New_Orleans_flooding_caused_by_Hurricane_Katrina_3_September_2005

[29] - ***, *Arduino Mega 2560 Datasheet*, Available from: https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf

[30] - ***, *Introduction to Arduino Mega 2560*, Available from: https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-mega-2560.html

[31] – Shaohao C., *Introduction to MATLAB*, Available from: https://www.bu.edu/tech/files/2017/01/intro_matlab_2017Spring.pdf

[32] – GERRITSEN M., *A brief introduction to MATLAB*, Available from: https://web.stanford.edu/class/cme001/handouts/matlab.pdf

[33] - ***, *MATLAB The language of technical computing*, Available from*:* https://www.mn.uio.no/astro/english/services/it/help/mathematics/matlab/getstart.pdf

[34] - ***, *Robot chassis*, Available from: https://www.amazon.com/wheel-layer-Chassis-Encoder-Arduino/dp/B06VTP8XBQ/ref=sr_1_8?dchild=1&keywords=robot+chassis&qid=1593335023&sr=8-8

[35] - ***, *Arduino DC Motor Control Tutorial 0 L298N*, Available from: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge

[36] - ***, *LiPo Battery pack*, Available from: https://shop.pimoroni.com/products/lipo-battery-pack?variant=20429081991

[37] - ***, *Servo motor,* Available from: https://www.slideshare.net/komalmehna/servo-motor-with-arduino-sg-90

[38] - ***, *Logic level converter,* Available from: https://www.sparkfun.com/products/12009

[39] - ***, *Camera support,* Available from: https://cleste.ro/suport-camera-cu-servomotor-pan-tilt.html

[40] - ***, *Azimuth and elevation*, Available from: https://whatis.techtarget.com/definition/azimuth-and-elevation

[41] - ***, *Azimuth and elevation,* Available from: https://www.photopills.com/sites/default/files/tutorials/2014/2-azimuth-elevation.jpg

[42] - ***, *Understanding Azimuth and Elevation,* Available from: https://www.photopills.com/articles/understanding-azimuth-and-elevation