Misr University of Science and Technology
College of Engineering Sciences and Technology
Department of Engineering Mechatronics

B. Eng. Final Year Project

Autonomous Self Driving Vehicle

By:
GROUP (4)

NAME	*ID*
Abdelhameed Mohamed Ali Ibrahim	*81087*
Fahad Nazih Ahmed Mohamed Ali	*86357*
David Maged Yousef Botrose	*91489*
Youhana Morcos Elkes Hana	*91615*
Bishoy Makram Youssef Tawadrous	*91629*
Youssef Ahmed Hamed Abdeltwab	*95829*

Supervised By:

Dr. Bahaa M. Nasser
Dr Tarek Mohamed
Ass. Prof. Mohamed Ahmed
Dr. Bikheet Mohamed
Dr Maha Salman

Spring 2024
08/06/2024

DEDICATION

This work is dedicated to my family, whose unwavering love and support have been my guiding light throughout this journey.

To my parents, who instilled in me the values of hard work and perseverance, and who have always encouraged me to pursue my dreams.

Additionally, I extend my heartfelt gratitude to my friends, whose camaraderie and encouragement have been invaluable. specially Eng Moaz Taha, Eng Ziad Shoeib and Eng Omar Ashraf.

Thank you all for your love and encouragement. This work is a testament to your enduring support and belief in me.

Acknowledgment

I would like to express my deepest gratitude to all those who have contributed to the completion of this work.

First and foremost, I wish to thank my supervisors, **Prof. Bahaa Nasser**, **Dr. Bekhet Mohamed**, **Dr.Maha *Salman*** , **Dr.Tarek Abd ELbadia** , **Dr.Mohamed** and **Eng. Waleed El badry** for their invaluable guidance, unwavering support, and insightful feedback throughout the research process. Their expertise and encouragement have been instrumental in shaping the direction and outcome of this study.

Lastly, I am deeply indebted to my family for their unwavering support, patience, and encouragement. Their belief in me has been a constant source of motivation and strength throughout this journey.

To all these individuals and organizations, I extend my heartfelt appreciation and thanks.

DECLARATION

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Bachelor of Science in Mechatronics Engineering is entirely my/our own work, that I/we have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others and to the extent that such work, if any, has been cited and acknowledged within the text of my work.

Signed:

_____

Date: 8/6/2024

Signed:

_____

Date: 8/6/2024

Signed:

_____

Date: 8/6/2024

Signed:

_____

Date: 8/6/2024

Signed:

_____

Date: 8/6/2024

Signed:

_____

Date: 8/6/2024

ABSTRACT

Autonomous vehicles have been invented to increase the safety of transportation users. These vehicles can sense their environment and make decisions without any external aid to produce an optimal route to reach a destination. This type of system can bring a revolution in transportation for differently abled people and help blind people travel independently.

In our project, we will be working on autonomous vehicles that can use Simultaneous Localization and Mapping (SLAM) and Robot Operating System (ROS) in autonomous vehicles with mapping and path planning. Navigation2 stack autonomous vehicles can navigate via different path planning algorithms using the map Previously built, all of these were achieved using sensors such as lidar and Kinect cameras.

The design of our vehicle is simple and made from acrylic material. It has two wheels and one caster wheel. lidar sensor and camera on the top of the vehicle.

Table of contents

List of figures:

List of tables:

1. Chapter I: Introduction and Literature review

1.1 Introduction

We survey research on This self-driving technology mainly deals with the vehicles that can work and move themselves by sensing the movements and environment without any human presence. These self-driving vehicles are also known as autonomous vehicles. A human driver nor any human passenger is needed compulsorily to operate this vehicle. It totally works on the operations done through sensors, remote controlled operations, artificial intelligence. Thus, making it a product of mechatronics field. A Vehicle that has functionality of autonomous, it means it will aware by itself and has capability to choose by itself. For example, you say “drive me to work”, instead to follow your instruction it will be start to drive at another point, no it will not happen, an autonomous vehicle will follow the driver instructions and will reach the destination as driver want. Sometime, we use words self-driven and autonomous interchangeably but there is minor difference between these two. For self-driving vehicle, driver should always be present with in the vehicle, even system can drive by itself, but still there should always be driver to take control of vehicle. We divide vehicles into various categories or levels. So self-driving vehicles come under the conditional and high driving automation level. Self-driving vehicles combines a variety of sensors to sense and connect to the nearest sensors, such as radar, LiDAR, sensors and GPS.

1.2 History of Autonomous Vehicles and Literature review

The idea and design of the first automated car was generated centuries before the first car was even made. In 1500, Leonardo Da Vinci created a cart that could move back and forth without needing to be pushed or pulled. It had springs that provided the power and steering which was set in advance so that the cart could move ahead on a predetermined path.

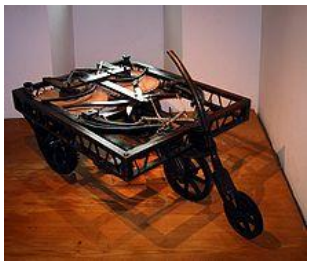


Figure 1 Leonardo da Vinci automobile invention

The first modern step toward automated cars began in 1925. That year, inventor Francis Houdiana, gave a demonstration of a radio-controlled car that could run freely on the streets of Manhattan without the requirement of any drivers. It could be turned on and off, shift gears, sound the horn, and move externally with the help of a radio. But the project couldn't go far because the operator lost control twice and crashed into another vehicle.

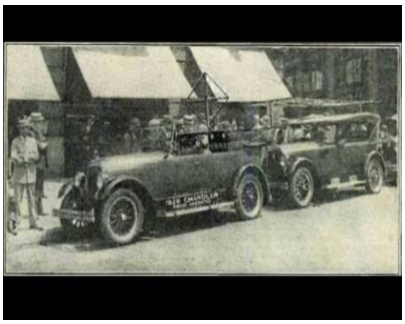


Figure 2. The RCA radio-controlled car

At the 1939's World's Fair, General Motors brought out the first self-driving car model. The car's design showed that it could be guided by a radio-controlled electromagnetic field which controlled the magnetized metal spikes embedded in the roadway. The model turned into reality in 1958. It contained a sensor that could detect surrounding objects and the path to some extent via electrical flow. The sensor could detect the current flow through a wire embedded on the road and send signals to the steering wheel to move left or right.

The next few years from then were filled with hype for space exploration. During that period, researchers were thinking about how to create vehicles that can roam by itself on the moon's surface. In 1961, James Adams created a robotic car named the "Stanford Cart" with a camera implanted in it that could follow designated lines on the ground. Using a camera to create an autonomous vehicle is considered a vital event on this journey.

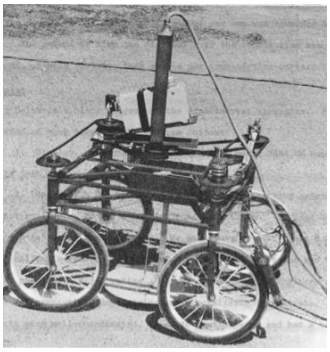


Figure 3.Stanford Cart with cable, 1961

Japanese scientists came up with an improvised idea in 1977, they implanted a camera that could take images of the road and process that data to move on the road. This technology led us to the world's first auto-driven passenger vehicle which could run at a max speed of 20 miles per hour.

At the end of the last century, Carnegie Mellon University integrated the neural networking method with camera implanted self-driving technology that evolved the image processing and steering control systems. They made a car, called the NavLab 5, and took it to the road for testing. Although the speed and braking were controlled by Carnegie Mellon researchers, the car autonomously traveled 2,797 miles from Pittsburgh to San Diego.

Figure 4.The Navlab 5, a 1990 Pontiac Trans Sport

In the early 2000s, the US Department of Research Arm (DARPA) sponsored further research in automotive cars. As a result, four fully self-driving cars were able to successfully cross the 60-mile-long test drive urban route by 2007. In mid-2010, major car manufacturers such as Ford, Mercedes-Benz, BMW, and the renowned ride-sharing company Uber, became interested in working with self-driving technology. However, in 2020, Uber announced the closure of its autonomous car operations due to safety, lawsuits, and financial losses.

Figure 5.Uber autonomous car

Self-driving Car Research's from Google:

Google is the most famous in the autonomous domain. Autonomous car technology research by Google began in 2005. In 2008, Google AV with Pribot from Levandowsku was able to deliver pizzas across the California Bay Bridge to San Francisco-Oakland. In 2009, Google's autonomous vehicle project was led by Sebastian Thrun, who also led the Stanford University team and winning the 2005 DARPA Grand Challenge with his "Stanley" car. Google's autonomous car has become an independent company called Waymo. Waymo's autonomous car uses a RADAR to detect distant objects and their speed, a LIDAR to create a detailed map of the world around the car, and high-resolution cameras to acquire visual information, such as the red or green traffic signal.

Figure 6.The Google's autonomous car

Self-driving Car Research's from Vislab:

In July 2013, an autonomous car developed by the Artificial Vision and Intelligent Systems Laboratory (VisLab) of the Parma University, Italy, drove around the old district of the city of Parma without any human participation. The autonomous car successfully passed the roundabout, the single two-lane, recognized the pedestrians crossing the road, recognized the traffic lights, and so on. In 2010, a driverless van made the longest journey approximately 13,000 kilometers within three months, which began in Italy and ended in China. In 2013, a driverless test program called PROUD-Car Test was organized by Vislab. It can be seen that a car with no one in the driver's seat can move successfully on a mixed traffic route (rural, highway and urban) open to public traffic.

Figure 7.VisLab

Self-driving Car Research's from Tesla.

In 2015, Tesla Autopilot introduced: freeway driving, in ramp to off-ramp and Tesla summon released. In 2016, all Tesla models built with appropriate hardware for SAE level 5 capabilities: steering and acceleration/ deceleration, monitor environment, fallback performance, capability in all driving modes. Tesla plans 90% autonomous cars for public which is expected to have an 'autopilot' feature which would make the '90% autonomous' travel possible.

Figure 8 .TESLA

1.3 Difference between Self-Driving Cars and Regular Cars:

Table 1 Self-driving Cars VS. Regular Cars

Regular Cars	Self-Driving Cars
Human-driven cars monitored by humans sitting behind the steering wheel.	Basic levels of automation used are electronic stability control (ESC), blind spot detection, antilock brake systems (ABS), etc
Driver has the control over driving but involves some kind of automation.	Use their own sensing systems and software programs to navigate the roads.
Human driven cars account for more road fatalities because of human drive error.	Designed to eliminate human drive error, reducing the epidemic of traffic incidents and fatalities.
Basic levels of automation used are electronic stability control (ESC), blind spot detection, antilock brake systems (ABS), etc	Use sensors and software programs, along with Al to help the car navigate the roads and avoid obstacles.

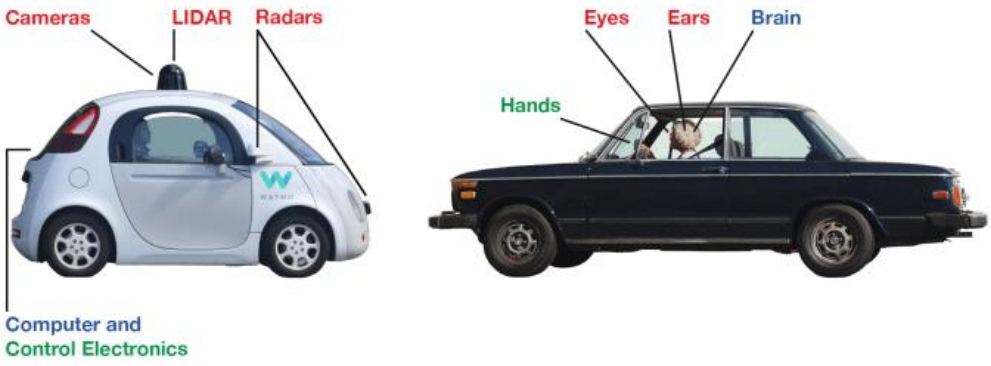


Figure 9. Self-Driving Cars and Regular Cars

1.4 The advantages of self-drive car:

- **Less coasts:** In 2021, U.S. annual vehicular fatality rate was 42,915; 94% of crashes are due to human error. AVs have the potential to remove/reduce human error and decrease deaths. AVs have the potential to reduce crashes by 90%, potentially saving approximately $190 billion per year.

- **More Safety:** Accidents are often caused by driver fatigue, lack of attention or incorrect behavior. This means that almost 99% of all accidents are due to human error. With the elimination of the driver as a source of error and increasing sophistication of systems (sensors, cameras, and AI system), driving can be made more efficient and the accident rate can be reduced. In addition, autonomous vehicles have a lower reaction time and thus shorten braking and starting times.

- **More Time and Comfort:** Depending on the level of the autonomous vehicle, drivers can sit back and relax, take short breaks and devote their time to other things. In the best case, with a level 5 vehicle, you are just a passenger, while the means of transport reliably takes you to your destination.
- help old aged and visually blinded people who are not able to take advantages of vehicle properly. These automated vehicles help them to provide safe and timely journey

1.5 Autonomous vehicle

Levels of Autonomous Vehicles:

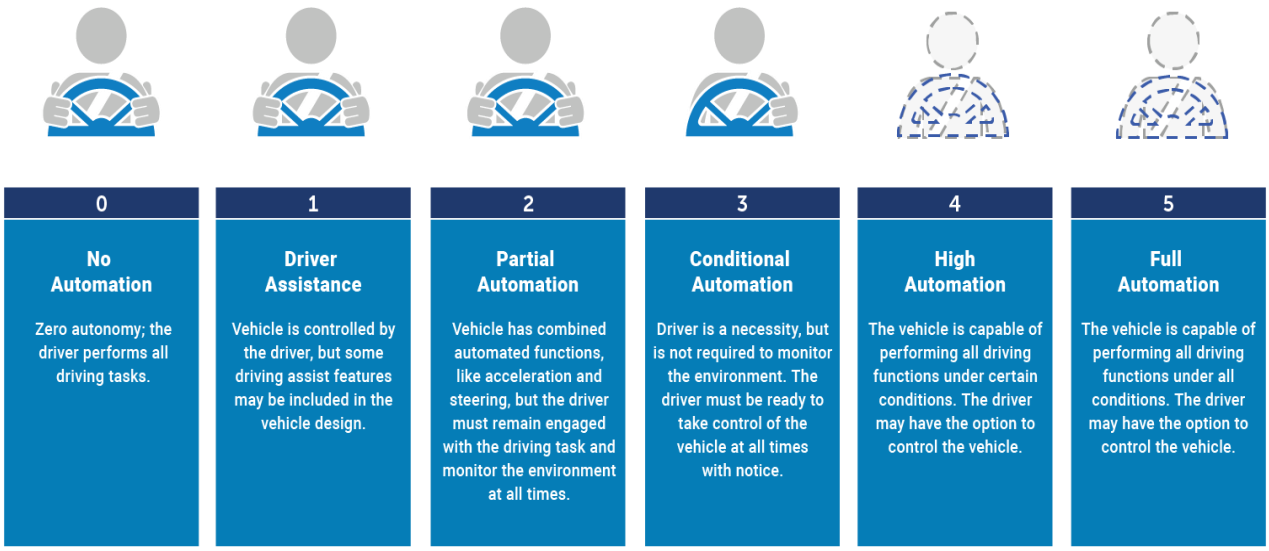


Figure 10. Levels of Autonomous Vehicles

1.6 Development of Autonomous Vehicles:

AV research started in the 1980s when universities began working on two types of AVs: one that required roadway infrastructure and one that did not. The U.S. Defense Advanced Research Projects Agency (DARPA) has held “grand challenges” testing the performance of AVs on a 150-mile off-road course. No vehicles successfully finished the 2004 Grand Challenge, but five completed the course in 2005. In 2007, six teams finished the third DARPA challenge, which consisted of a 60-mile course navigating an urban environment obeying normal traffic laws. In 2015, the University of Michigan built Mcity, the first testing facility built for autonomous vehicles. Research is conducted there into the safety, efficiency, accessibility, and commercial viability of AVs. Unmanned aircraft systems (UAS), or drones, are being deployed for commercial ventures such as last-mile package delivery, medical supply transportation, and inspection of critical infrastructure.

1.7 Autonomous Vehicle Technologies:

AVs use combinations of technologies and sensors to sense the roadway, other vehicles, and objects on and along the roadway.

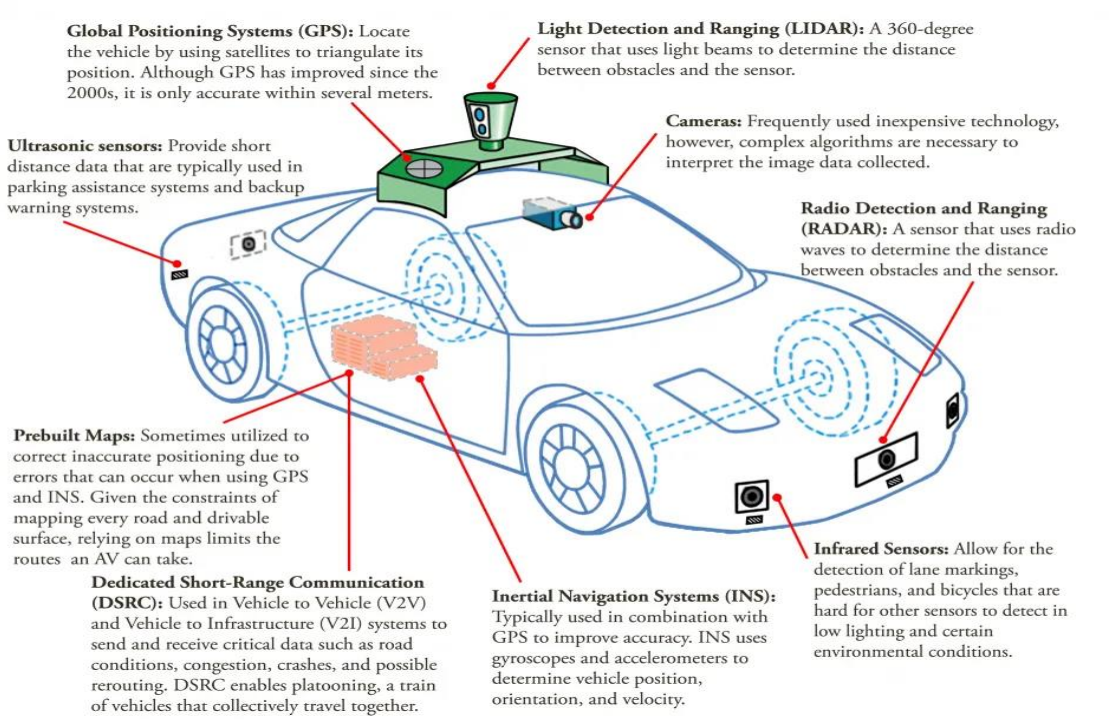


Figure 11.Autonomous Vehicle Technologies

1.8 Working of Automated Vehicle:

1. For any intelligent vehicle, first step is always to know where they are currently in the world. For that very basic things need to know local coordinates, road boundaries and intersections, it is local map. Two types of maps may be used real time map or stored map. In real time map making, lots of components are use like rndf, gps, camera, lidar, radar etc.
2. Sensor, short range proximity radar, ultrasonic sensors may be used to identify the objects near to vehicle for collision avoidance.
3. For tracking nearby traffic light, some school lane signal or any other signal, track other vehicle can be done by video cameras.
4. To measure distances, detect road edges, identify lane marking, Lidar (light detection and ranging) can be used.

5. Finally, the software part of the automated vehicle system takes all these sensory inputs and analyses all that data and provide usable information to vehicle's actuators and send controlled instruction to it.

1.9 CHALLENGES:

Testing is already going on various automated vehicles all around the world but still it is not widely available in marked due to various challenges in this concept. These challenges are not just technical but also related to environment also.

- **Lidar:** To look the surrounding of car to make an automated system, there is need of lot of Lidar and Radar Sensors. These sensors are very expensive and also its processing is totally different to see the object and understands that as compare to emergency breaking a control function of cars.
- **Complex software System**: There is need of large amount of complex code to implement an automated system. This system takes lot of input from sensor from human and from other information sources and then algorithm applied on this large amount of information to take a right decision. For this complex system, there is need of high-end processor also.
- **Traffic Conditions and Laws:** For implementing any automated system, firstly algorithm should be capable to understand our traffic light system and our government laws. System should be aware that what is the rule of traffic light and from how much distance, it should sense the lights.
- **Weather Conditions**: What happens when an autonomous car drives in heavy precipitation? If there's a layer of snow on the road, lane dividers disappear. How will the cameras and sensors track lane markings if the markings are obscured by water, oil, ice, or debris?[6]

1.10 Design considerations

- Design considerations for an autonomous vehicle (AV) are crucial to ensuring safety, efficiency, and user experience. Here are some key considerations:
- Sensing and Perception Systems: AVs rely on a variety of sensors, including cameras, LiDAR, radar, and ultrasonic sensors, to perceive the environment. These sensors must be robust, accurate, and capable of providing real-time data to the vehicle's control system.
- Data Fusion and Processing: AVs process large amounts of sensor data to understand their surroundings and make driving decisions. Efficient algorithms for data fusion, object detection, classification, and localization are essential.
- Control and Decision-Making Algorithms: AVs require sophisticated algorithms to interpret sensor data, plan trajectories, and make driving decisions in real-time. These algorithms should prioritize safety while considering factors such as traffic laws, road conditions, and potential hazards.
- Human-Machine Interface (HMI): The HMI should be intuitive and user-friendly, providing passengers with relevant information about the AV's status, route, and upcoming maneuvers. Clear communication of the vehicle's intentions is crucial for passenger trust and comfort.
- Vehicle Design and Integration: AV technology needs to be seamlessly integrated into the vehicle's design, including considerations of weight distribution, and structural integrity. AV components should be compact, durable, and capable of withstanding harsh environmental conditions.
- Mapping and Localization: High-definition maps and accurate localization are essential for AVs to navigate complex environments with precision. Mapping technology must be continuously updated to reflect changes in road infrastructure, construction zones, and traffic patterns.

1.11 Conclusion:

Since Self Driving Car is the major upgradation in automatable industry in future, this project focuses on bring changes in road safety and commuting and significantly reduce accidents and human errors through continuous learning by the system. This project will be an important step in transporting differently abled people and blind people can drive independently

2. Chapter II: Mechanical design

2.1 Introduction

Mechanical design is the process of designing components, parts, products, and systems of mechanical nature. The primary objective of mechanical design is to ensure that the final product or system is safe, reliable, efficient, and cost-effective. We will go to select the material and do calculations.

2.2 Material:

Many different materials are used in the manufacture of autonomous vehicles body and the most used materials nowadays are steel, aluminum, stainless steel, and acrylic. We chose acrylic material for the car body.

2.2.1 The Advantages of Acrylic:

- Easy to Fabricate and Shape
- light weight and ideal for precision machining
- Highly impact resistance
- Easy to Maintain
- Good dimensional stability
- Highly resistant to different chemicals
- Highly resistant to variations in temperature

2.2.2 properties of Acrylic:

Table 2 properties of Acrylic

Property	Value
Technical Name	Acrylic (PMMA)
Chemical Formula	(C5H8O2) n
Melt Temperature	130°C (266°F)
Typical Injection Mold Temperature	79-107°C (175-225°F)
Heat Deflection Temperature (HDT)	95°C (203°F) at 0.46 MPa (66 PSI)
Tensile Strength	70 Mpa (9400 PSI)
Flexural Strength	90 Mpa (13000 PSI)
Specific Gravity	1.18
Shrink Rate	0.2 – 1% (.002 - .01 in/in)
Relative Density	1.19 g/cm^3
Minimum Service Temperature	-40ºC
Maximum Service Temperature	80ºC
Light Transmission	92%
Refractive Index	1.49
Water Absorption	0.20%
Yield strength	60 Mpa

2.3 Mechanical parts:

Table 3 Mechanical Parts

Part name	Number	Mass (Kg)
Vehicle cover	1	0.7
Vehicle body	1	2.6
Motors	2	0.4
Wheels	2	0.2
Caster wheel	1	0.2
Lidar	1	0.3
Battery	1	2.5
screw	26	0.1
Total Mass		7

2.4 Motor Sizing:

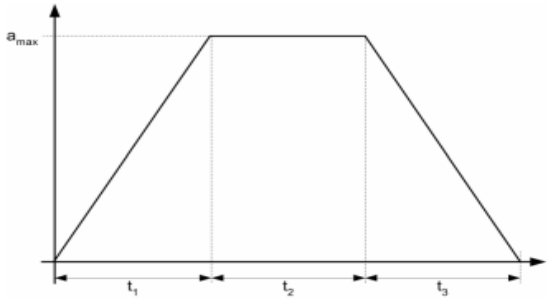
For the DC Motors:

Total weight of car = 7 KG = 68.67 N

Assuming the equal distribution of load = 68.67/3= 22.89 *N* for each wheel.

The load weight for one motor = 22.89 N

2.4.1 Calculate angular acceleration:



t_1 Time of Increasing Acceleration

t_2 Time of Constant Acceleration

t_3 Time of Decreasing Acceleration

Figure 12Trapezoidal curve of acceleration with respected time

Assume t_o: the positioning time = 4 sec

calculate the acceleration (deceleration) time at roughly 25% of the positioning time.

$$\mathbf{t_1} = \mathbf{t_3} = 0.25 \times t_o = 0.25 \times 4 = 1 \text{ sec}$$

N: no. of Revolution = 210 rpm

ω_o: initial angular velocity = 0

ω_{max}: maximum angular velocity

α_{max}: maximum angular acceleration

$$\boldsymbol{\omega_{max}} = \frac{2\times\pi\times N}{60} = \frac{2\times\pi\times 210}{60} = 22 \text{ rad/sec} \qquad \text{eq.2.1}$$

$$\boldsymbol{\alpha_{max}} = \frac{\omega_{max}-\omega_o}{t1} = \frac{22-0}{1} = 22 \text{ rad/sec}^2 \qquad \text{eq.2.2}$$

2.4.2 To Find the Torque of motor:

a. Calculating the rolling resistance:

$$RR = GVW \times C_{rr} \qquad \text{eq.2.3}$$

RR = Rolling Resistance
GVW = Gross Vehicle Weight = 68.67
Crr = Co-efficient of Rolling Resistance (concrete: fair = 0.015)

$$RR = 68.67 \times 0.015 = 1.03 \text{ N}$$

b. Calculating the grade resistance:

$$GR = GVW * \sin\theta \qquad \text{eq.2.4}$$

GR = Grade Resistance
θ = Grade or inclination angle. (assume Θ = 20 degrees)

$$GR = 68.67 \times \sin(20) = 23.48 \text{ N}$$

c. Calculating the friction force:

- **For static:**

$$F_{rs} = m \times g \times \mu_s \qquad \text{eq.2.5}$$

Where:
F_{rs}: Static $Force\ to\ move\ object\ at\ constant\ speed$
μ_s: Static Coefficient of friction
at $\boldsymbol{\mu_s} = 0.4$ so $F_{rs} = 2.33 \times 9.81 \times 0.4 = 9\ N$

- **For kinetic:**

$$F_{rk} = m \times g \times \mu_K \qquad \text{eq.2.6}$$

Where:

F_{rk}: Kinetic *Force to move object at constant speed*.

μ_k: Kinetic Coefficient of friction.

at $\mu_k = 0.3$ so $F_{rk} = 2.33 \times 9.81 \times 0.3 = 6.8\ N$

d. Calculating the required acceleration:

- **For static:**

$$a = \frac{F_{rs}}{m_e} \qquad \text{eq.2.7}$$

Where:

a: maximum acceleration static motion (start motion)

m_e: car mass = 7 Kg

F_{rs}: Max friction force at static = 9 N

$$\text{so} \quad a = \frac{9}{7} = 1.28\ m/s^2$$

- **For Kinetic:**

$$a = \frac{F_{rk}}{m_e}$$

Where:

a: maximum acceleration at kinetic motion

m_e: car mass = 7 Kg

F_{mk}: Max friction force at kinetic = 6.8 N

$$\text{So} \quad a = \frac{6.8}{7} = 0.97\ m/s^2$$

e. Calculating the acceleration force:

$$FA = m \times a \qquad \text{eq.2.8}$$

FA = Acceleration force

m = mass of the vehicle

a = required acceleration

$$FA = 7 \times 1.28 = 8.96\ N$$

f. Finding the total tractive effort:

$$TTE = RR + GR + FA \qquad \text{eq.2.9}$$

TTE = Total tractive effort

$$TTE = 1.03 + 23.48 + 8.96 = 33.5\ N$$

g. Torque required on the drive wheel:

$$T = R_f \times TTE \times R_{wheel} \qquad \text{eq.2.10}$$

T = Motor torque

R_f = Friction factor that account for frictional losses between bearings, axles etc. (assume R_f=0.3)

R_{wheel} = radius of drive wheel = 0.065 m

$$T = 0.3 \times 33.5 \times 0.065 = 0.653 \text{ N.m} = 653 \text{ N.mm} = 6.6 \text{ Kg.cm}$$

2.4.3 Calculating the motor power:

$$\text{MOTOR POWER} = \frac{2 \times \pi \times N \times T}{60} \qquad \text{eq.2.11}$$

T: motor torque

N: number of revolutions = 210 rpm

$$\text{MOTOR POWER} = \frac{2 \times \pi \times 210 \times 0.653}{60} = 14.36 \text{ watt}$$

1 *hp* = 746 watt

$$\text{MOTOR POWER} = 0.019 \text{ hp}$$

2.4.4 The Shaft:

The Material used for the shaft is steel Grade 45C8

Yield strength= 350 Mpa

Take the factor of safety = 3

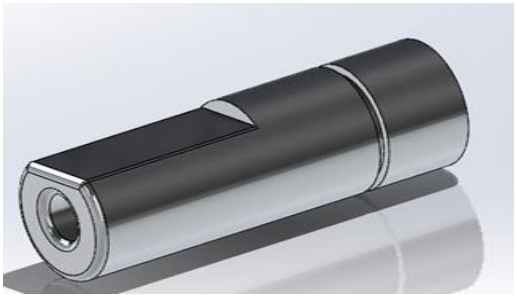


Figure 13 shaft

1. **Find allowable shear and bending stress:**

τ = allowable Shear stress induced due to twisting moment.

σ_b = allowable Bending stress (tensile or compressive) induced due to bending moment

$$\tau = \frac{\sigma_y \times 0.5}{f.s} = \frac{350 \times 0.5}{3} = 58.33\ Mpa$$

$$\sigma_b = \frac{\sigma_y}{f.s} = \frac{350}{3} = 116.67\ Mpa$$

2. **Find the Bending moment:**

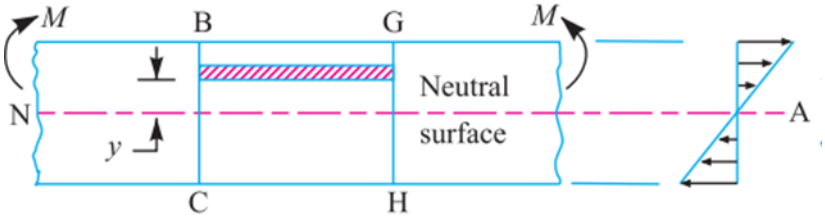


Figure 14 Bending moments

Assuming the equal distribution of load, each shaft carries load = 22.89 N.

M: Bending moment

W: weight

L: length between center of bearing and center of rim = 15 mm

Weight = 22.89 N

M = W × L = 22.89 × 15 = 343.35 N. mm

3. To find the diameter of shaft:

Table 4 recommended values for Km and Kt

Nature of load	K_m	K_t
1. Stationary shafts		
(*a*) Gradually applied load	1.0	1.0
(*b*) Suddenly applied load	1.5 to 2.0	1.5 to 2.0
2. Rotating shafts		
(*a*) Gradually applied or steady load	1.5	1.0
(*b*) Suddenly applied load with minor shocks only	1.5 to 2.0	1.5 to 2.0
(*c*) Suddenly applied load with heavy shocks	2.0 to 3.0	1.5 to 3.0

Km: Combined shock and fatigue factor for bending = 1.5

Kt: Combined shock and fatigue factor for torsion = 1

τ: allowable Shear stress induced due to twisting moment.

σ_b: allowable Bending stress (tensile or compressive) induced due to bending moment

T_e : The equivalent twisting moment

M_e: The equivalent bending moment

$T_e = \sqrt{(Km \times M)^2 + (Kt \times T)^2} = \sqrt{(1.5 \times 343.35)^2 + (1 \times 653)^2} = 831.6\ N.mm$ eq.2.12

$T_e = \frac{\pi}{16} \times \tau \times d^3 = 831.6mm$ ***so d = 4 mm***

$M_e = \frac{1}{2} \times [(Km \times M) + Te] = \frac{1}{2} \times [(1.5 \times 343.35) + 831.6] = 673.3\ N.mm$ eq.2.13

$M_e = \frac{\pi}{32} \times \sigma_b \times d^3 = 673.3mm$ **so, d = 3.8 mm**

So, we will choose d = 4 mm

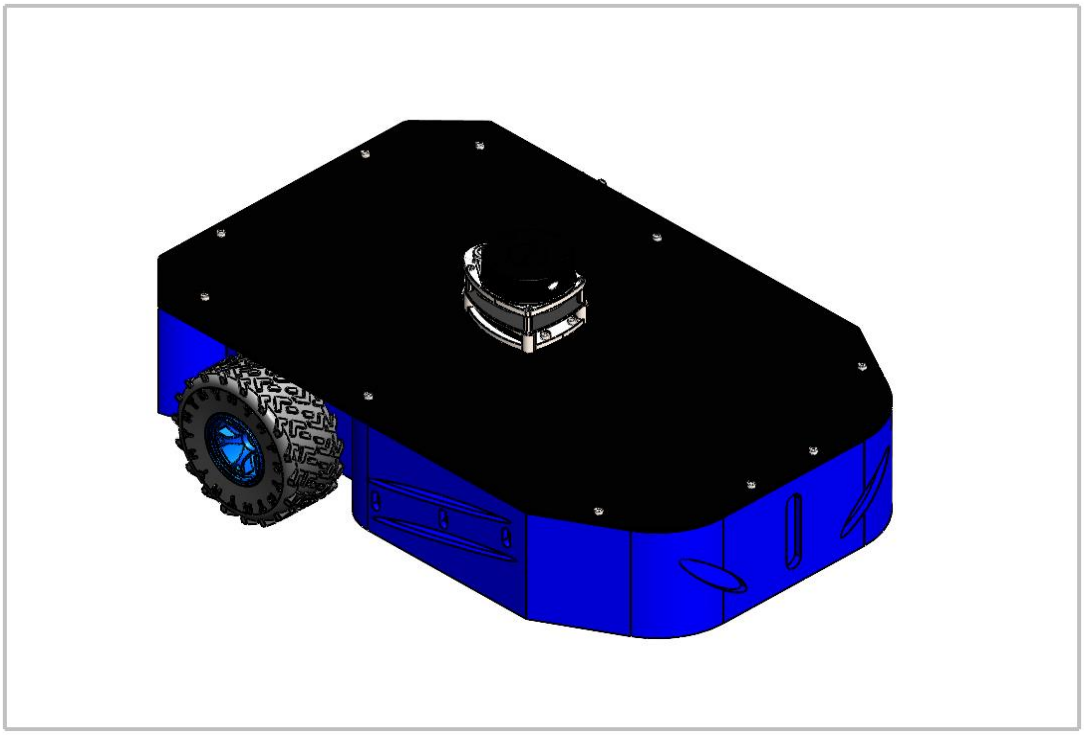
2.4.5 The motor available in the market:

(Dc Gear Motor with Magnetic Linear Encoder 25GA370) motor data sheet

- Nominal voltage: 12 V
- Free-run speed at 12 V: 210 RPM
- Free-run current at 12 V: 46 mA
- Stall current at 12 V: 2 A
- Stall torque at 12 V: 7.8 kg.cm
- Shaft diameter is 4 mm
- Weight: 200 g

Figure 15 Dc Gear Motor with Magnetic Linear Encoder 25GA370

2.5 SolidWorks design:



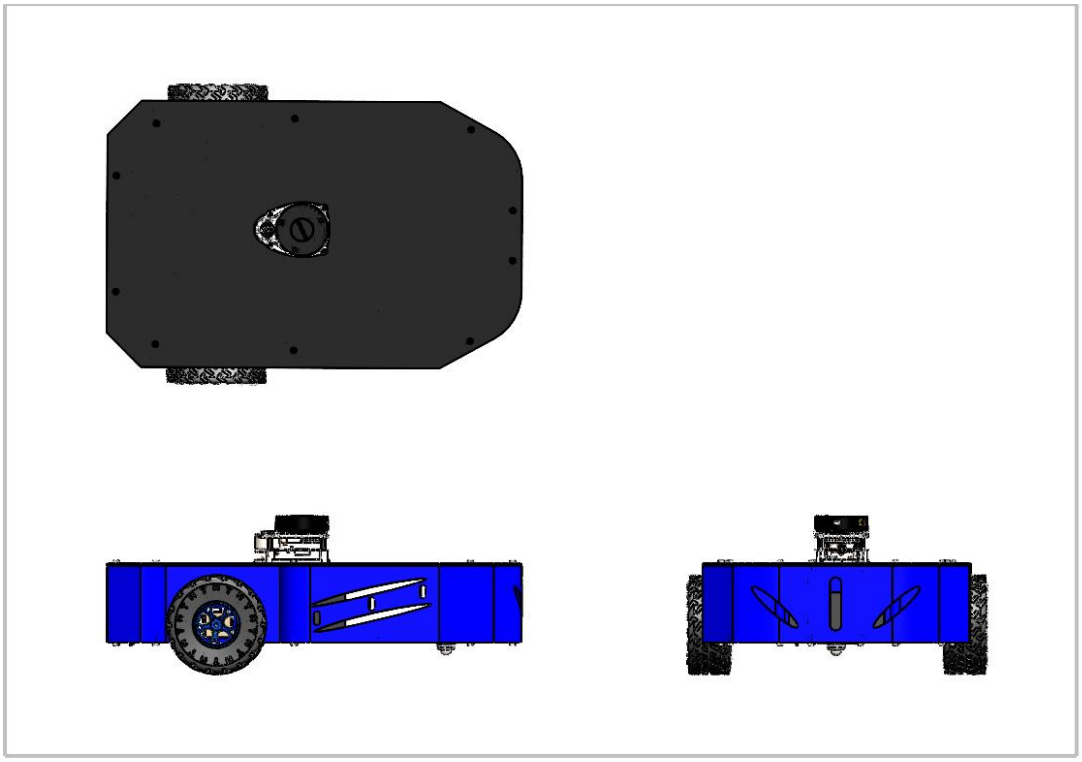


Figure 16. Vehicle

2.5.1 The SolidWorks Parts:

1. The Cover:

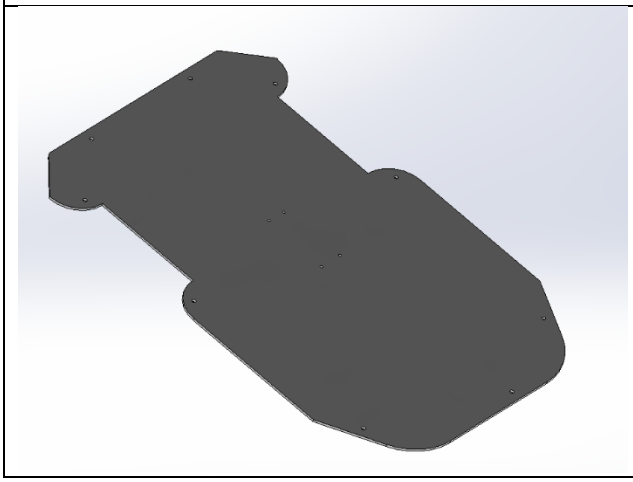
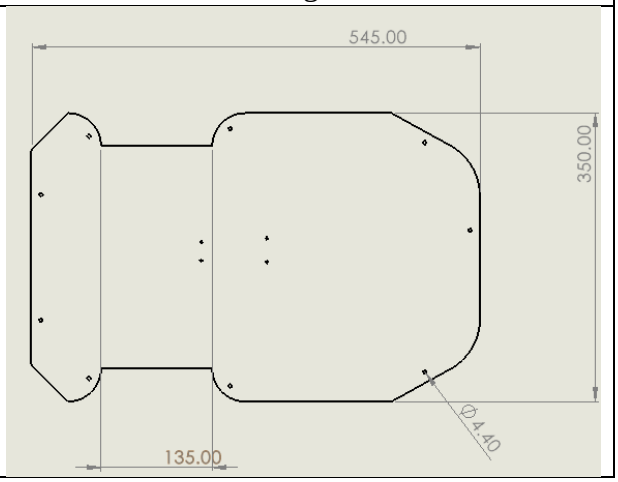
3D View	Drawing Sheet
	

Figure 18.Vehicle Cover

2. The Body:

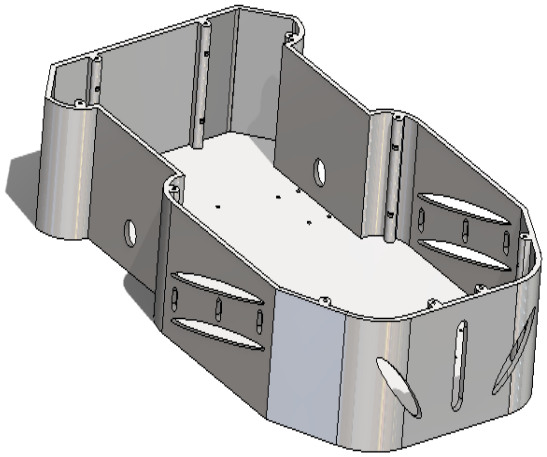
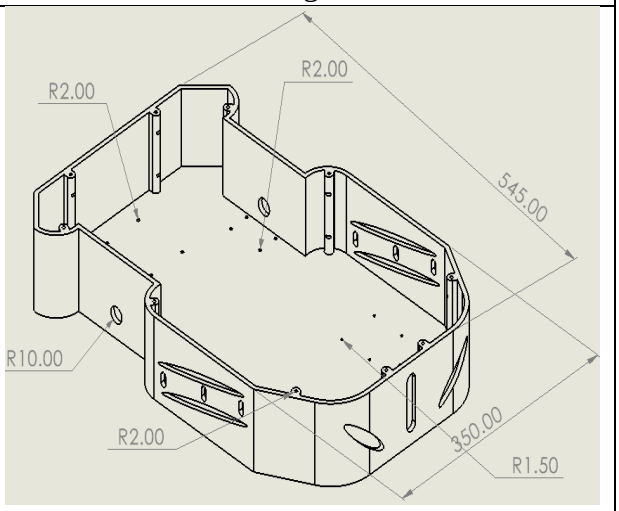
3D View	Drawing Sheet
	

Figure 19. Vehicle Body

3. The Wheel:

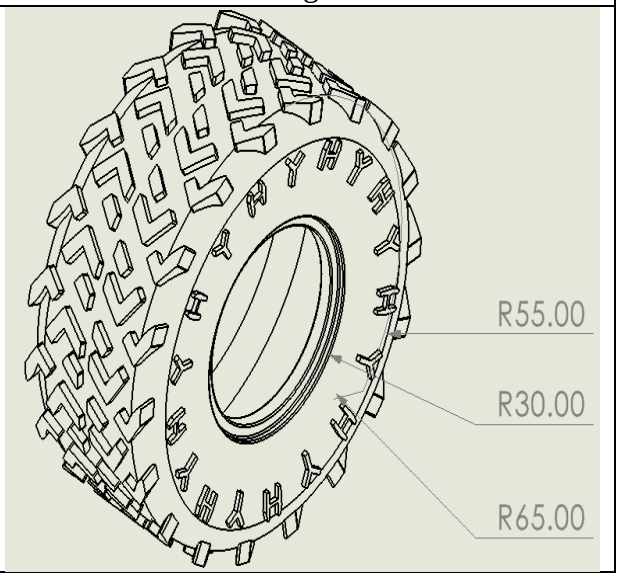
3D View	Drawing Sheet
	

Figure 20. Wheel

4. The Rim:

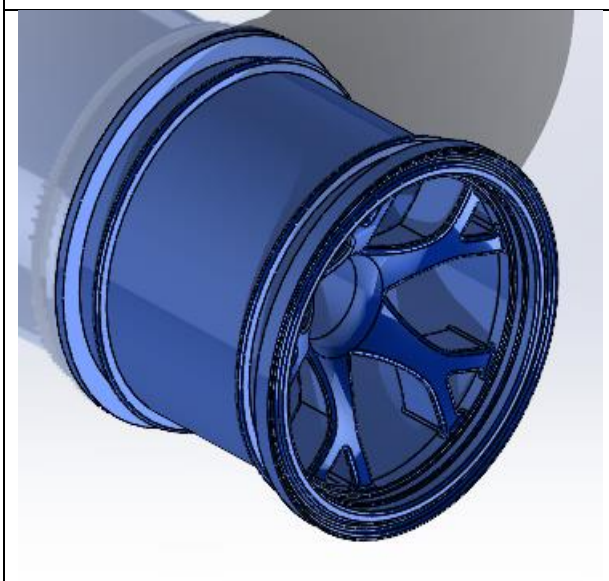
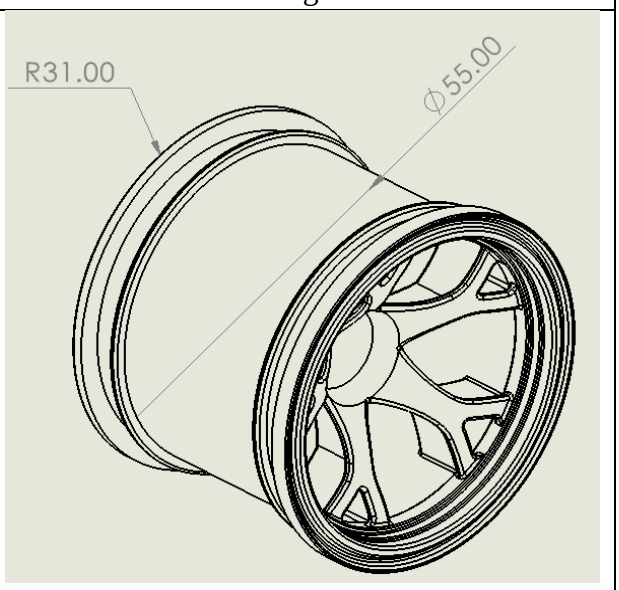
3D View	Drawing Sheet
	

Figure 21. Rim

5. The Ball Caster Wheel:

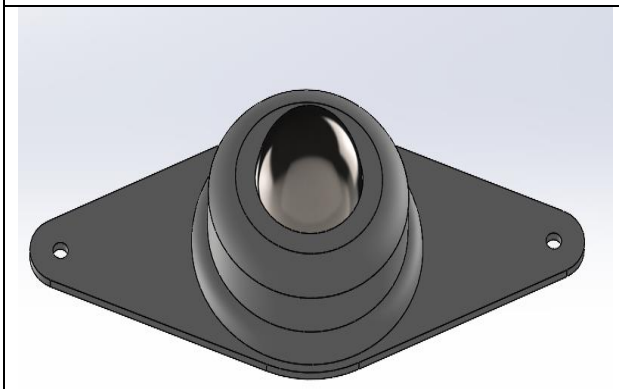
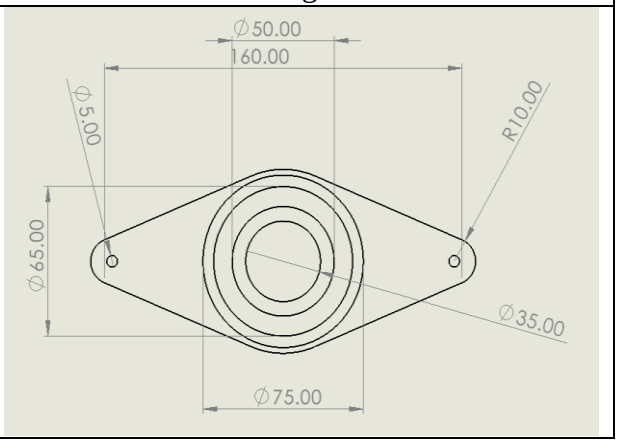
3D View	Drawing Sheet
	

Figure 22. Castor Wheel

6- The Vehicle:

- 3D View:

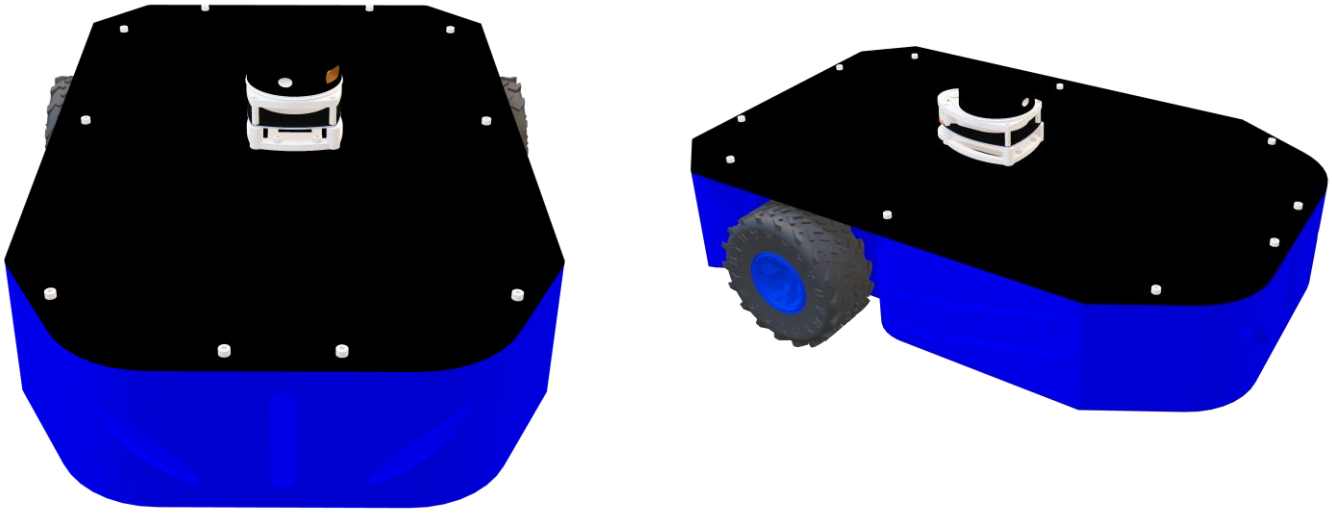
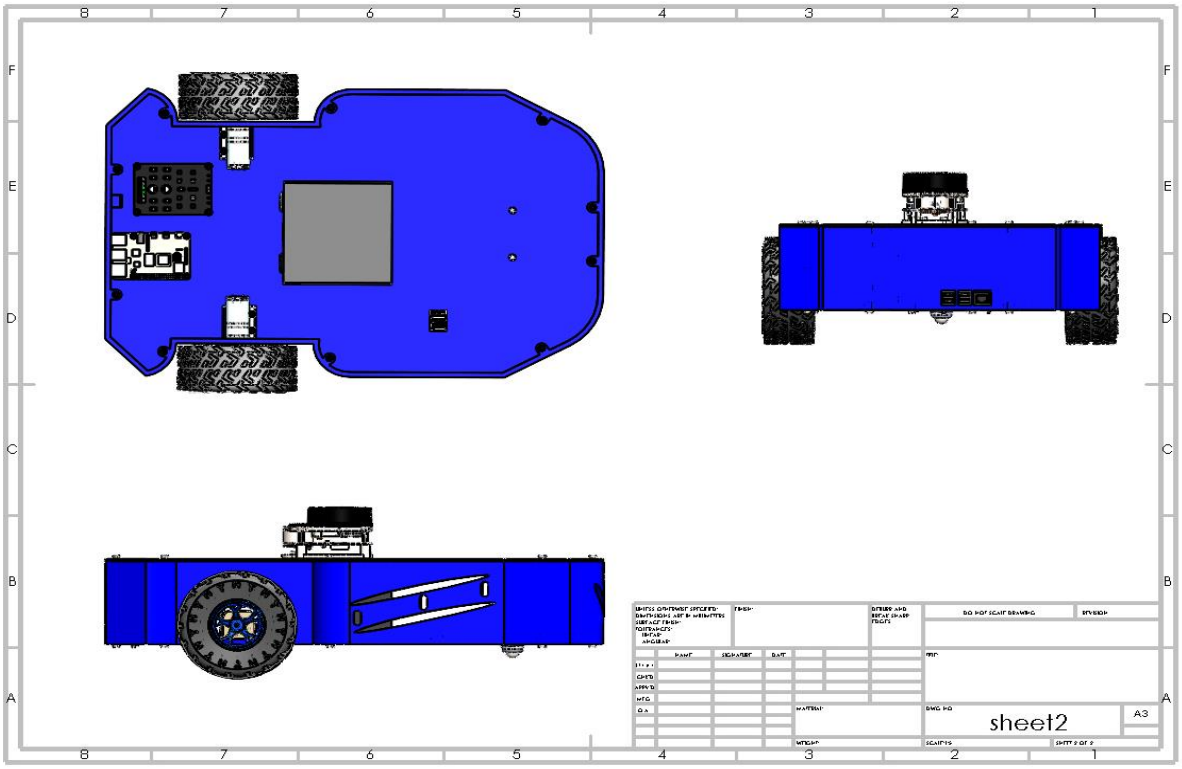
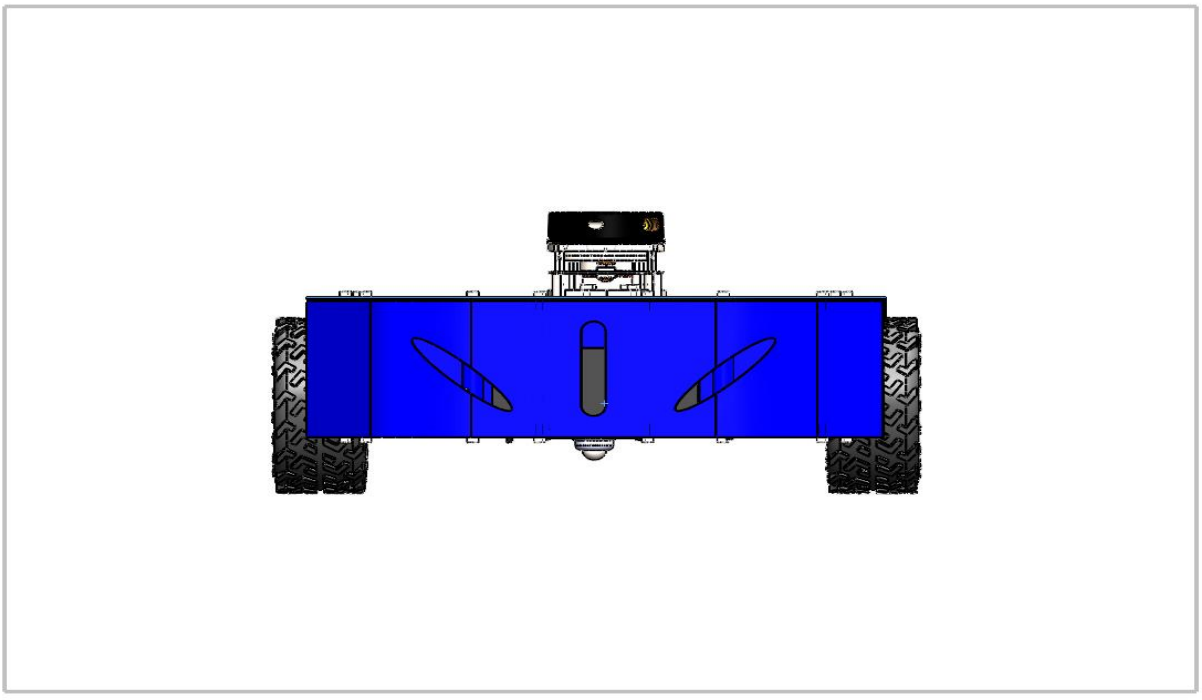


Figure 23. 3D Views

2.5.2 The Drawing Sheet:

Figure 24. Drawing Sheet





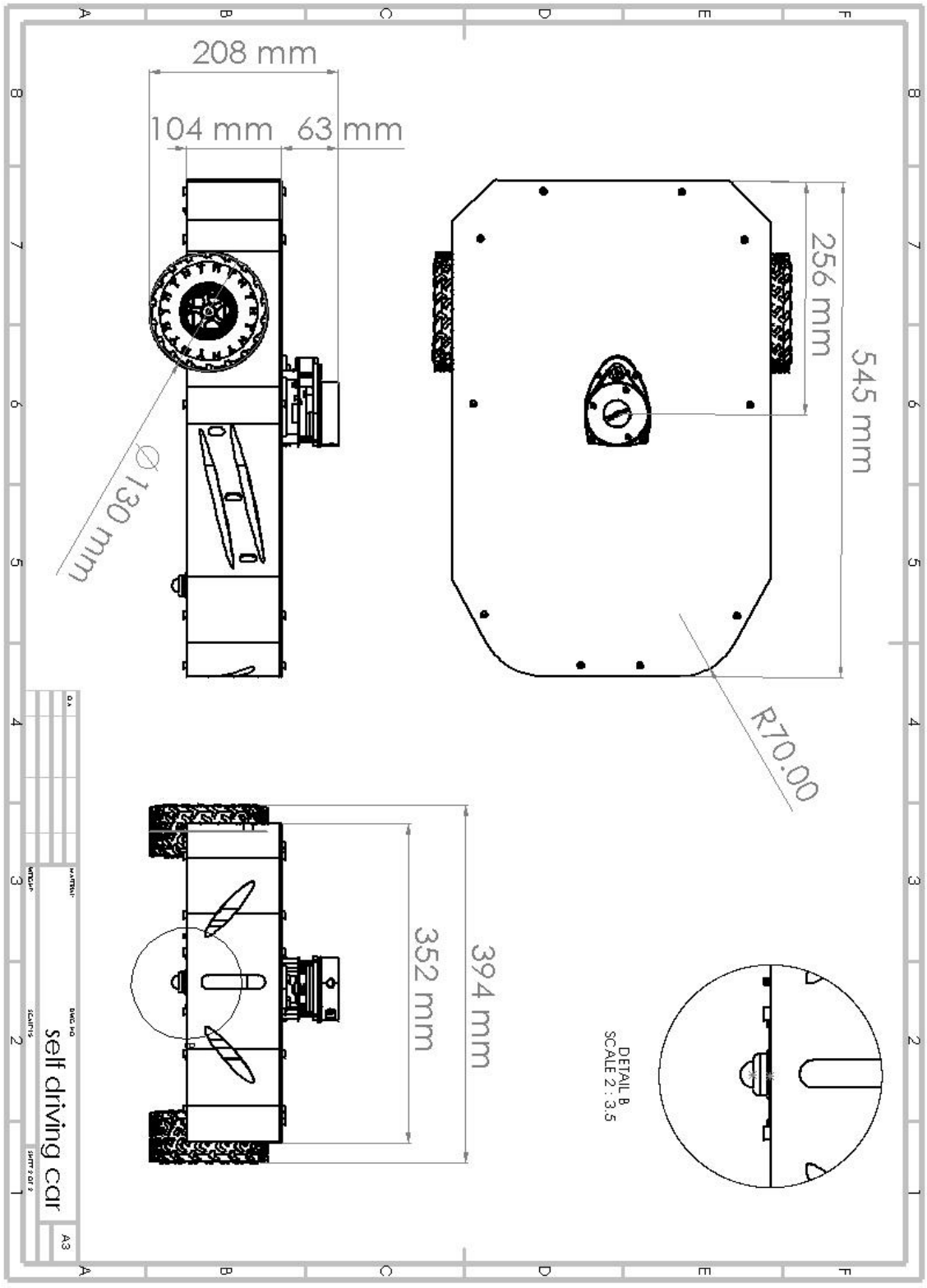
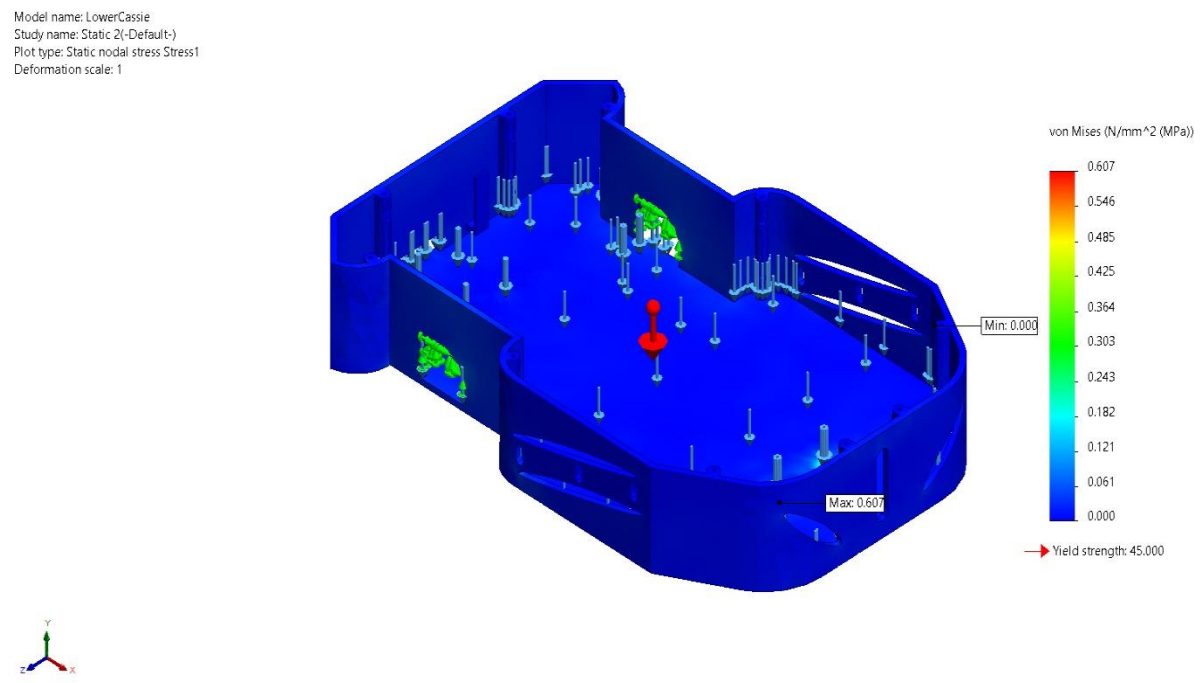


Figure 25 Dimensions

2.5.3 Stress Analysis:

Study Results

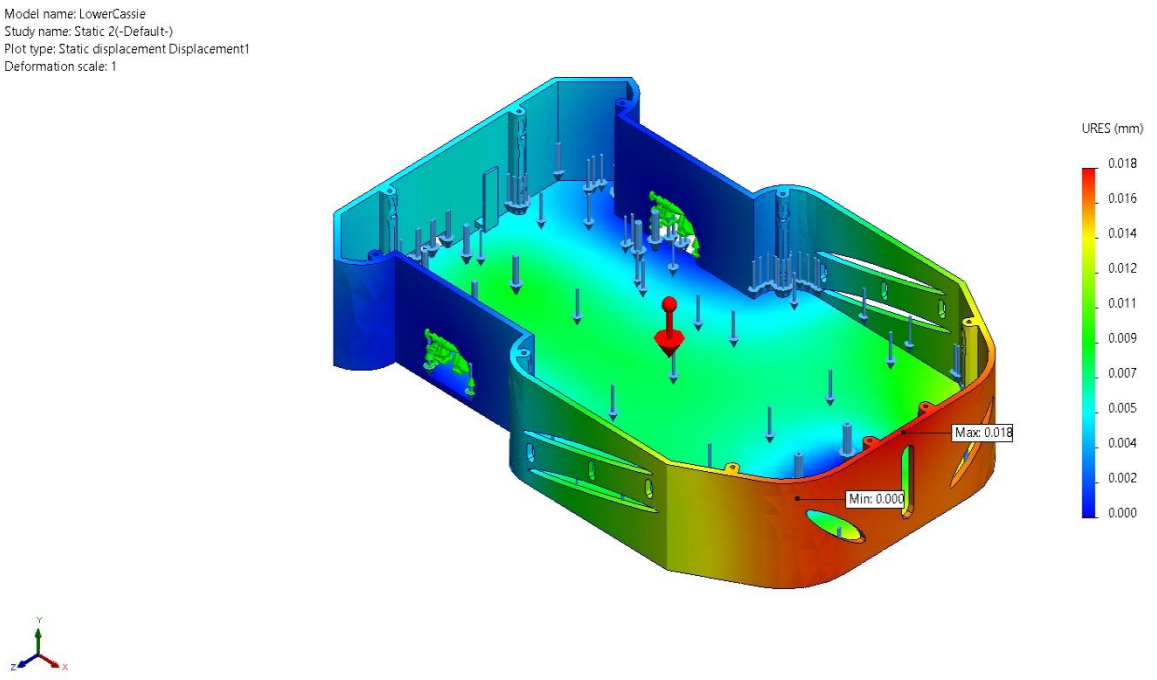
Name	Type	Min	Max
Stress1	VON: von Mises Stress	0.000N/mm^2 (MPa) Node: 6143	0.607N/mm^2 (MPa) Node: 259



LowerCassie-Static 2-Stress-Stress1

Figure 26 Stress Analysis

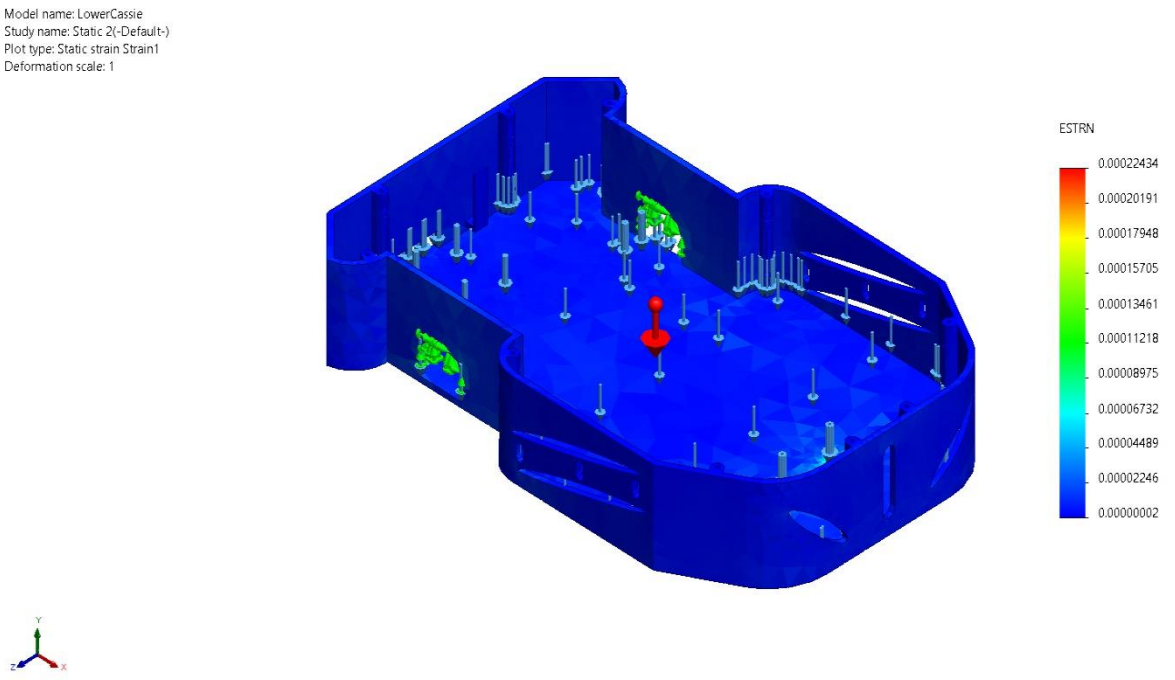
Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0.000mm Node: 258	0.018mm Node: 6786



LowerCassie-Static 2-Displacement-Displacement1

Figure 27 Displacement Analysis

Name	Type	Min	Max
Strain1	ESTRN: Equivalent Strain	0.00000002 Element: 15105	0.00022434 Element: 48541



LowerCassie-Static 2-Strain-Strain1

Figure 28 Strain Analysis

2.6 Kinematics:

Driving a model for the whole robot's motion is a bottom-up process. Each individual wheel contributes to the robot's motion and, at the same time, imposes constraints on robot motion. Wheels are tied together based on robot chassis geometry, and therefore their constraints combine to form constraints on the overall motion of the robot chassis. But the forces and constraints of each wheel must be expressed with respect to a clear and consistent reference frame. This is particularly important in mobile robotics because of its self-contained and mobile nature; a clear mapping between global and local frames of reference is required. We begin by defining these reference frames formally, then using the resulting formalism to annotate the kinematics of individual wheels and whole robots.

Throughout this analysis we model the robot as a rigid body on wheels, operating on a horizontal plane. The total dimensionality of this robot chassis on the plane is three, two for position in the plane and one for orientation along the vertical axis, which is orthogonal to the plane. Of course, there are additional degrees of freedom and flexibility due to the wheel axles, wheel steering joints and wheel castor joints. However, by robot chassis we refer only to the rigid body of the robot, ignoring the joints and degrees of freedom internal to the robot and its wheels.

In order to specify the position of the robot on the plane we establish a relationship between the global reference frame of the plane and the local reference frame of the robot as. The axes and define an arbitrary inertial basis on the plane as the global reference frame from some origin O: {X1, Y1}. To specify the position of the robot, choose a point P on the robot chassis as its position reference point. The basis {XR, YR} defines two axes relative to P on the robot chassis and is thus the robot's local reference frame. The position of P in the global reference frame is specified by coordinates x and y, and the angular difference between the global and local reference frames is given by $\emptyset$. We can describe the pose of the robot as a vector with these three elements. Note the use of the subscript I to clarify the basis of this pose as the global reference frame:

$$\xi_1 = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

To describe robot motion in terms of component motions, it will be necessary to map motion along the axes of the global reference frame to motion along the axes of the robot's local reference frame. Of course, the mapping is a function of the current pose of the robot. This mapping is accomplished using the orthogonal rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix can be used to map motion in the global reference frame $\{X_I, Y_I\}$ to motion in terms of the local reference frame $\{X_R, Y_R\}$. This operation depends on the value of θ:

$$\xi_R = R\left(\frac{\Pi}{2}\right)\xi_I$$

compute the instantaneous rotation matrix R:

$$R\left(\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Given some velocity $(\dot{x}, \dot{y}, \dot{\theta})$ in the global reference frame we can compute the components of motion along this robot's local axes X_R *and* Y_R . In this case, due to the specific angle of the robot, motion along X_R is equal to $\dot{y}$ and motion along Y_R is -$\dot{x}$:

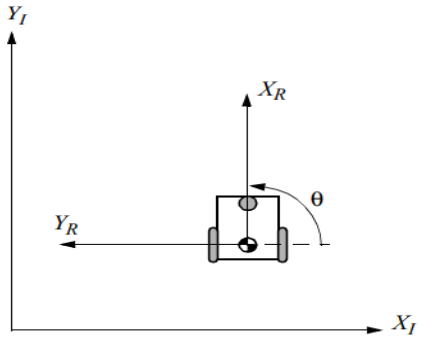


Figure 30 The mobile robot aligned with a global axis

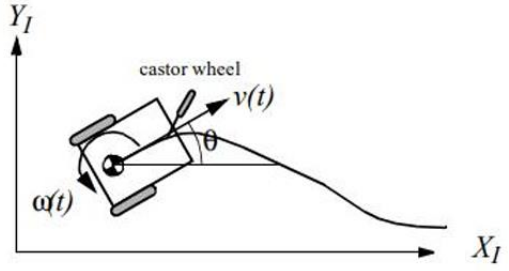


Figure 29 a differential drive robot in its global reference frame

2.6.1 Design of a Two-Wheel drive mobile robot

- Mobile robots

Mobile robots are complex mechatronic autonomous systems characterized by combinations of devices equipped sensors that can operate in a real world or working environment. Among the mobile robots, those that move on tracks or wheels with tires are increasingly used for the execution of special works with a high degree of danger to the human operator, among which are: handling and neutralization of unexploded projectile; the execution of some corridors through the minefields; search of vehicles, trains, aircraft and buildings, followed by the tracing of explosive devices discovered in these vehicles. One important problem for mobile robots is collision avoidance with fixed or mobile obstacles in the robot's workspace. This can be done by several methods. Among the most commonly used navigation methods are: measuring the number of rotations made by the drive wheels, the use of accelerators and gyroscopes, electromagnetic beacons installed in the field, passive or semi-passive signals of an optical or magnetic type.

- Differential steering for mobile robots (Kinematics)

Kinematics is that branch of classical mechanics that deals with the relationships between joint parameters or the relative movement between kinematic elements of a robot and the behavior of the entire robotic system in the workspace or state space. With a differential steering locomotion system, a robot can realize a rotational movement around an axis perpendicular to a point contained by the axis of the two traction wheels. By changing the angular velocities of the two

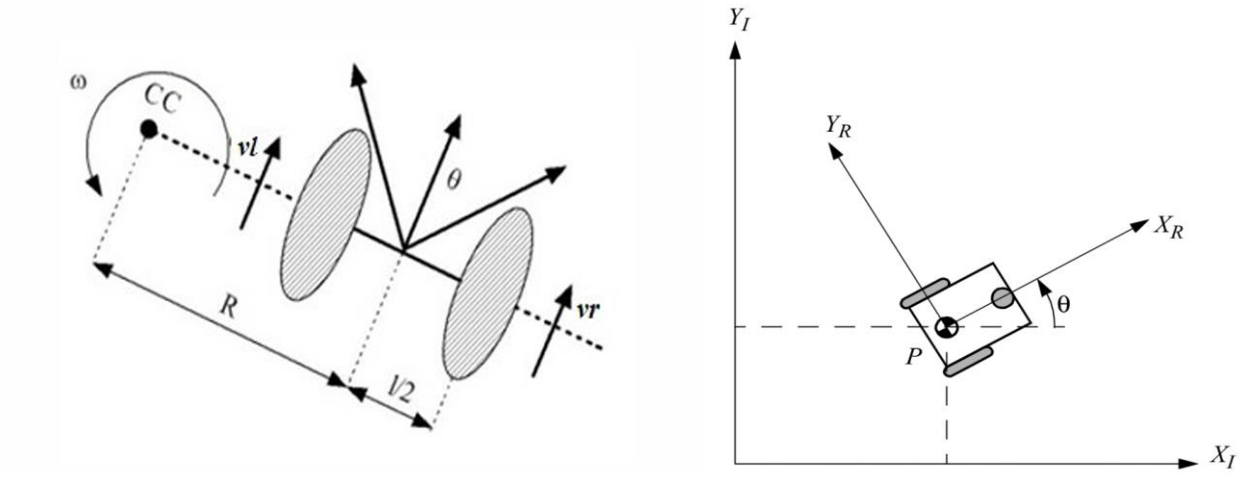


Figure 31 differential steering for mobile robots

traction wheels, the direction of rotation can be changed. The angular velocities of the two wheels should respect the mathematical equation presented below:

$$v_r = \omega \cdot (R + \frac{1}{2}),\ v_1 = \omega \cdot (R - \frac{1}{2}).$$

In the above formula, $v_1$characterizes the angular velocity for the left wheel and vr characterizes the angular velocity for the right wheel. R describes the radius of the circular path travelled by the center of the robot. ω the angular velocity of the wheel, and l the distance between the two traction wheels. The orientation of the robot is described by θ angle. By solving the system of equations one can obtain the solutions presented below:

$$R = [l \cdot (v_r + v_l)] / [2 \cdot (v_r - v_l)],\ \omega = (v_r - v_l)/l$$

If the two angular velocities are equal $v_1 = v_r$. the radius R is infinite. In this case, the robot will move in a straight line. If the two angular velocities have the same value but different direction $v_1 = - v_r$, the distance R is 0, and the robot will rotate around a vertical axis placed in the middle of the I axis. For different values for $v_1$and v_r, the robot will move on a circular path of radius R with respect to the center of curvature CC. Using the right values for the angular velocities $v_1$and v_r the robot could reach a multitude of points other than the starting point. Using the direct kinematics equations one can determine the exact position of the mobile robot:

$$x(t) = 1/2 \cdot \int_{0\text{-}t}[v_r\,(t) + v_1\,(t)] \cdot \cos[\theta(t)] \cdot dt$$
$$y(t) = 1/2 \cdot \int_{0\text{-}t}[v_r\,(t) + v_1\,(t)] \cdot \sin[\theta(t)] \cdot dt$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - xcc \\ y - ycc \\ \theta \end{bmatrix} + \begin{bmatrix} xcc \\ ycc \\ \theta \end{bmatrix}$$

The above equations describe the rotational motion of a robot at a distance R between the middle of the axis of the two wheels and the center of curvature CC with angular velocity ω. By integrating the above equation, starting from a set of initial conditions (x0, y0, θ0), the position of the robot at a time t can be determined, based on the control parameters v_1 (t) and v_2 (t).

workspace a mobile robot must have appropriate actuation of its degrees of freedom. This problem, called motorization, requires further analysis of the forces that must be actively supplied in order to realize the kinematic range of motion available to the robot. In addition to motorization, there is the question of controllability: under what conditions can a mobile robot travel from the initial pose to the goal pose in bounded time? Answering this question requires knowledge both knowledge of the robot kinematics and knowledge of the control systems that can be used to actuate the mobile robot. Mobile robot control is therefore a return to the practical question of designing a real-world control algorithm that can drive the robot from pose to pose using the trajectories demanded for the application.

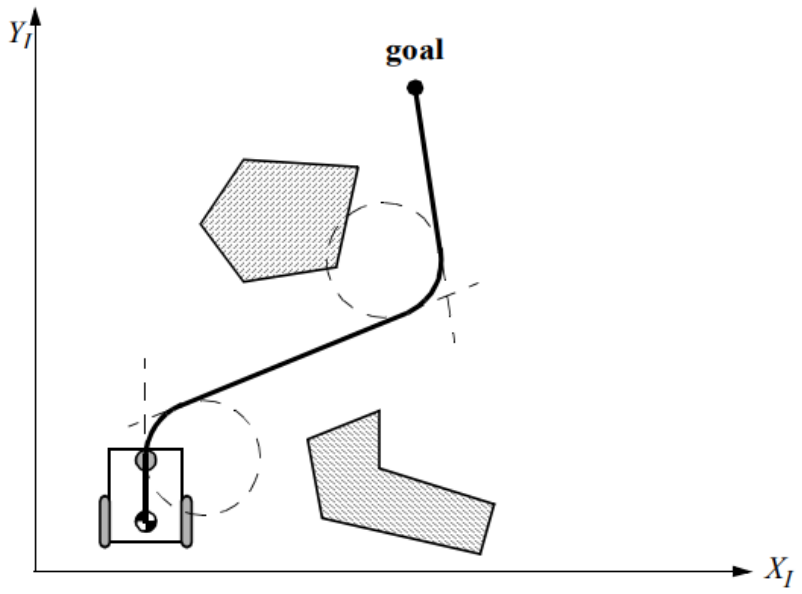


Figure 32 Open loop control of a mobile robot based on straight lines and circular trajectory segments

2.6.2 Open Loop Control (trajectory following):

The objective of a kinematic controller is to follow a trajectory described by its position and/ or velocity profiles as function of time. This is often done by dividing the trajectory (path) in motion segments of clearly defined shape, e.g. straight lines, and segments of a circle. The control problem is thus to pre-compute a smooth trajectory based on line and circle segments which drives the robot from the initial position to the final position. This approach can be regarded as open-loop motion control because the measured robot position is not fed back for velocity or position control. It has several disadvantages:

• It is not at all an easy task to pre-compute a feasible trajectory if all limitations and constraints of the robot's velocities and accelerations have to be considered.

• The robot will not automatically adapt or correct the trajectory if dynamic changes of the environment occur.

• The resulting trajectories are usually not smooth, because the transitions from one trajectory segment to another are for most of the commonly used segments (e.g. lines and part of circles) not smooth. This means there is a discontinuity in the robot's acceleration.

2.6.3 Feedback control

A more appropriate approach in motion control of a mobile robot is to use a real state feedback controller. With such a controller the robot's path-planning task is reduced to setting intermediate positions (sub-goals) lying on the requested path. One useful solution for a stabilizing feedback control of differential drive mobile robots is presented in the following section.

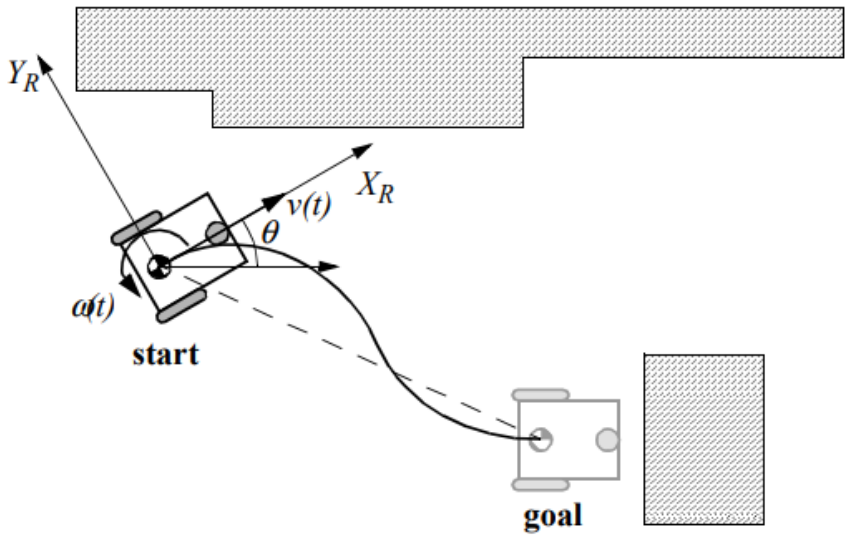


Figure 33 Feedback-control of a mobile robot

2.7 Implementation on MATLAB

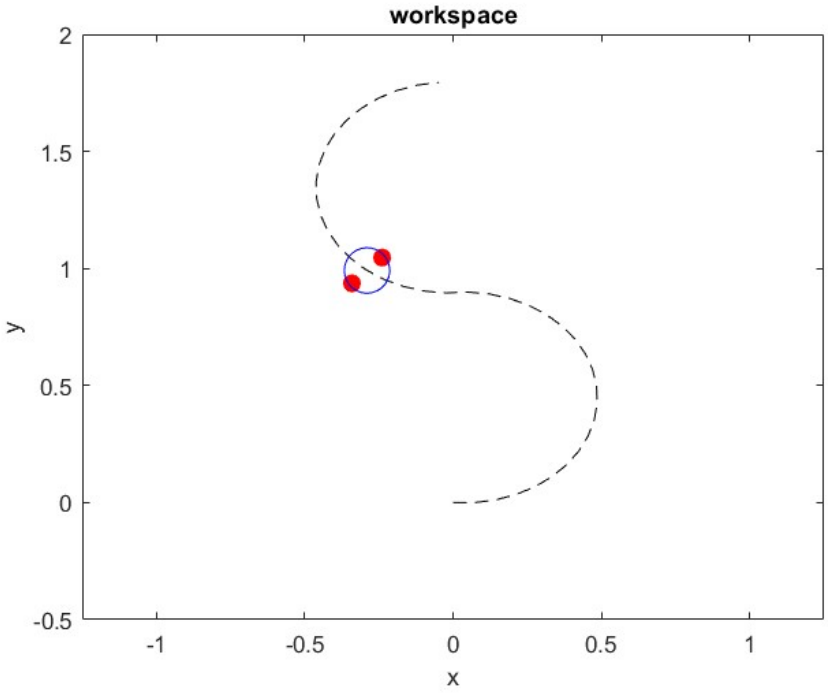


Figure 34 Implementation of two wheeled robot on MATLAB

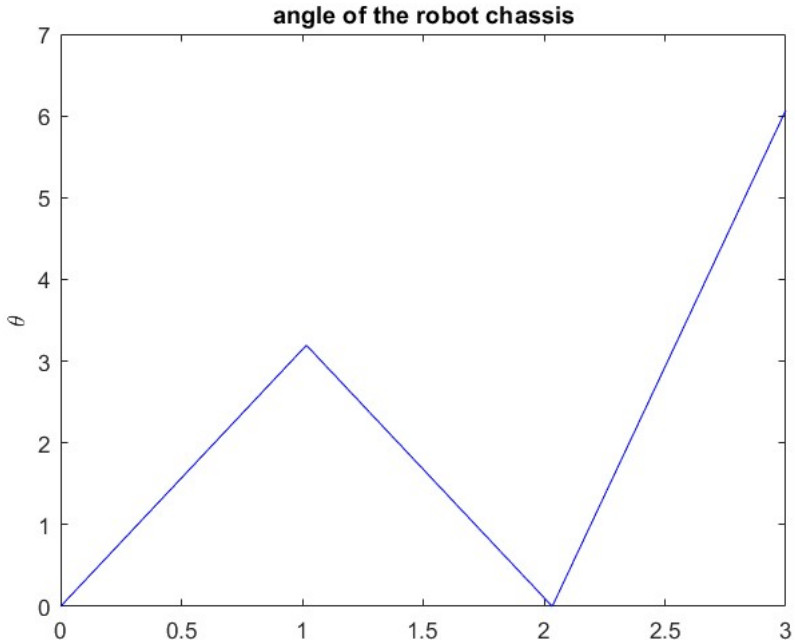


Figure 35 Angle of the robot chassis

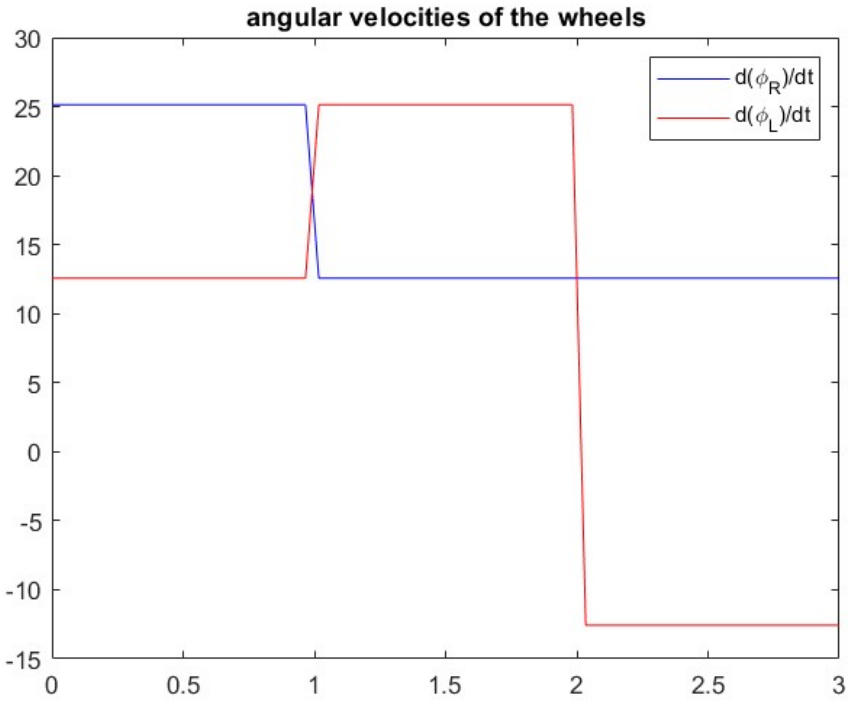


Figure 36 Angular velocities of the wheels

2.8 Conclusion:

Mechanical design includes various aspects such as material selection, mechanical calculations, and integration of mechanical parts.

- Material selection is a critical step in mechanical design because it directly affects the performance, durability, and cost-effectiveness of the final product.

- Mechanical calculations are essential to ensure the structural integrity and functionality of mechanical systems. These calculations help determine the appropriate dimensions, tolerances and safety factors required for each part, ensuring that the system can withstand expected forces and perform reliably throughout its life cycle.

- Mechanical parts integration involves the design and assembly of individual components to create a functional system. You must ensure that the parts fit together seamlessly, allowing for smooth operation and efficient power transmission. In addition, they focus on improving the arrangement and alignment of components to reduce friction, reduce wear and increase overall system performance.

3 Chapter three: Electrical design

3.6 components

3.6.1 Raspberry Pi 4 Model B

Key Features and Specifications

1. Processor:

   Broadcom BCM2711, Quad-core Cortex-A72 (ARMv8) 64-bit SoC @ 1.5GHz.

2. Memory:

   4GB LPDDR4-3200 SDRAM.

3. Connectivity:
   - 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac wireless LAN.
   - Bluetooth 5.0/BLE (Bluetooth Low Energy).
   - Gigabit Ethernet.

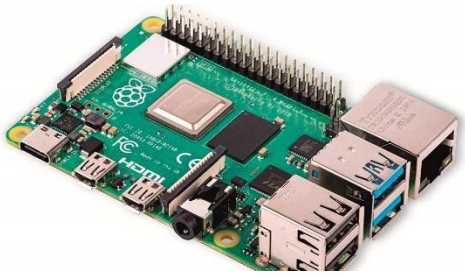


Figure 37. Raspberry pi 4

4. USB Ports:
   - 2 × USB 3.0 ports.
   - 2 × USB 2.0 ports.
5. Video and Audio Output:
   - 2 × micro HDMI ports (supporting up to 4Kp60).
   - 3.5mm 4-pole stereo audio jack and composite video.
6. Multimedia:
   - H.265 (4Kp60 decode).
   - H.264 (1080p60 decode, 1080p30 encode).
   - OpenGL ES 3.0 graphics.
7. Storage:

   microSD card slot for operating system and data storage.

8. GPIO:
   - 40-pin GPIO header, compatible with previous Raspberry Pi models.
   - SPI, I2C, serial, and GPIO pins available for interfacing with external devices.
9. Power:

- USB-C power supply (5V/3A recommended).

10. Operating System Support:
    - Raspbian (now known as Raspberry Pi OS), which is based on Debian.
11. Form Factor:
    - 85.6mm × 56.5mm × 17mm.
12. Cooling:
    - Improved thermal management with a heat spreader and a fan can be added.
13. Other Features:
    - PoE (Power over Ethernet) support via a separate PoE HAT.
    - Dual display support.

a. Comparison to Previous Modules

1. **Processor:**
    - Raspberry Pi 4: Broadcom BCM2711, Quad-core Cortex-A72 (ARMv8) 64-bit SoC @ 1.5GHz.
    - Raspberry Pi 3: Broadcom BCM2837B0, Quad-core Cortex-A53 64-bit SoC @ 1.4GHz.
2. **Memory:**
    - Raspberry Pi 4: 4GB LPDDR4-3200 SDRAM.
    - Raspberry Pi 3: 1GB LPDDR2 SDRAM.
3. **USB Ports:**
    - Raspberry Pi 4: 2 × USB 3.0 ports, 2 × USB 2.0 ports.
    - Raspberry Pi 3: 4 × USB 2.0 ports.
4. **Network Connectivity:**
    - Raspberry Pi 4: Gigabit Ethernet, 802.11b/g/n/ac wireless LAN, Bluetooth 5.0.
    - Raspberry Pi 3: 10/100 Ethernet, 802.11n wireless LAN, Bluetooth 4.2.
5. **Video Output:**

- Raspberry Pi 4: 2 × micro HDMI ports (up to 4Kp60).
- Raspberry Pi 3: 1 × HDMI port (up to 1080p).

6. **GPIO:**
    - Both models have a 40-pin GPIO header, but the GPIO on the Raspberry Pi 4 is backward compatible with previous models.
7. **Storage:**
    - Both models use a microSD card for storage.
8. **Form Factor:**
    - Raspberry Pi 4 is slightly larger than Raspberry Pi 3 (85.6mm × 56.5mm vs. 85.6mm × 56.5mm).
9. **Thermal Management:**
    - Raspberry Pi 4 includes improved thermal management with a heat spreader.
10. **Power:**
    - Raspberry Pi 4 uses a USB-C power supply, while Raspberry Pi 3 uses a micro USB power supply.
11. **Dual Display Support:**
    - Raspberry Pi 4 supports dual displays, which is not a native feature of Raspberry Pi 3.

3.6.2 RP LIDAR A1M8 2D

What is LIDAR?

LIDAR is a remote sensing method that uses lasers to measure distances to objects. It stands for "Light Detection and Ranging" and is commonly used in self-driving cars, robotics, and archaeology.

Key Features and Specifications

1. **Range:**
    - Lidar sensors have a specified range within which they can accurately measure distances. The range can vary depending on the specific model.
2. **Accuracy:**
    - The accuracy of distance measurements is a crucial specification for lidar sensors. It is often expressed in millimeters or centimeters.
3. **Resolution:**
    - Lidar sensors provide a level of spatial resolution, indicating the ability to distinguish between closely spaced objects. Higher resolution allows for more detailed mapping.
4. **Scanning Speed:**
    - Lidar sensors often have a scanning mechanism that determines how quickly they can acquire data points. Faster scanning speeds are beneficial in applications where real-time data is crucial.
5. **Field of View (FoV):**
    - The field of view defines the angular range over which the lidar sensor can detect objects. A wider field of view may be advantageous in certain applications.

6. **Data Output Rate:**

    - Lidar sensors generate a considerable amount of data, and the rate at which this data is output can be an important consideration, especially in applications with high data processing requirements.

7. **Wavelength:**

    - The wavelength of the laser used by the lidar sensor can impact its performance in different environmental conditions.

8. **Operating Conditions:**

    - Lidar sensors may have specified operating conditions, including temperature ranges and environmental factors.

9. **Interface:**

    - The interface options available for connecting the lidar sensor to other devices or systems, such as USB, Ethernet, or others.

10. **Size and Weight:**

    - Physical dimensions and weight can be critical factors, especially in applications where space and weight are limited.

a. Comparison: RP Lidar A1 M8 vs RP Lidar A2 M7

RP Lidar A1 M8

- High precision scanning
- 360-degree field of view
- Long range detection
- Fast data acquisition

Figure 38. RPLIDAR

RP Lidar A2 M7

- High precision scanning
- 360-degree field of view
- Improved range detection
- Enhanced data acquisition speed

3.6.3 KINECT CAMERA

What is Kinect 3D Camera?

The Kinect is a 3D camera originally developed for Microsoft's Xbox video game console. It uses infrared technology to track movement and depth perception, making it ideal for gesture-based and body-tracking applications.

Key Features:

1. Depth Sensing:

   The Kinect camera uses time-of-flight technology to create a 3D map of its surroundings, allowing it to measure distances accurately.

2. RGB Camera:

   In addition to depth sensing, the Kinect camera includes a standard RGB camera that captures color information.

3. Microphone Array:

   Kinect features a built-in microphone array that enables audio capture and voice recognition. It can be used for voice commands and communication.

4. Motorized Tilt:

   The Kinect camera has a motorized tilt mechanism that allows it to be adjusted remotely, providing flexibility in capturing the environment.

5. Skeleton Tracking:

   Kinect is capable of skeletal tracking, allowing it to recognize and track the movements of individuals within its field of view.

6. Facial Recognition:

   The camera can perform facial recognition, identifying and tracking faces in its range.

7. Gesture Recognition:

   Kinect supports gesture recognition, enabling users to interact with applications using hand gestures and movements.

8. SDK Support:

   Microsoft provides a Software Development Kit (SDK) for Kinect, allowing developers to create applications and programs that utilize Kinect's capabilities.

Specifications (Kinect for Xbox One):

Figure 39. Kinect camera

1. Depth Sensor:
   - Range: 0.5m to 4.5m.
2. Field of View:
   - Horizontal Field of View: 70 degrees.
   - Vertical Field of View: 60 degrees.
3. Color Camera:
   - 1080p resolution.
4. Frame Rate:
   - Depth and RGB: 30 frames per second (fps).
5. Audio:
   - 4-channel microphone array.

6. Power Supply:
    - Kinect for Xbox One typically requires an external power supply.
7. Connectivity:
    - USB 3.0.
8. SDK Compatibility:
    - Compatible with the Kinect for Windows SDK and other development frameworks.

a. Comparison to Previous Modules

Kinect for Xbox One:

1. Improved Depth Sensing:
    - The Kinect for Xbox One features an improved depth-sensing technology known as time-of-flight, offering better accuracy and a longer sensing range compared to the structured light technology used in the Kinect for Xbox 360.
2. Higher Resolution RGB Camera:
    - The RGB camera in the Kinect for Xbox One has a higher resolution (1080p) compared to the 640x480 resolution of the Kinect for Xbox 360.
3. Enhanced Skeletal Tracking:
    - The skeletal tracking capabilities are enhanced in the Kinect for Xbox One, providing more accurate and detailed tracking of body movements.
4. Facial Recognition:
    - Kinect for Xbox One introduces facial recognition technology, allowing the camera to identify and track individual faces.

5. Improved Microphone Array:
    - The Kinect for Xbox One includes a 4-channel microphone array, providing better audio capture and voice recognition capabilities compared to the Kinect for Xbox 360.
6. Motorized Tilt:
    - Kinect for Xbox One features a motorized tilt mechanism, allowing users to adjust the camera's angle remotely.
7. USB 3.0 Connectivity:
    - The Kinect for Xbox One uses USB 3.0 for connectivity, offering faster data transfer rates compared to the USB 2.0 used by the Kinect for Xbox 360.
8. Discontinuation:
    - Microsoft discontinued the Kinect for Xbox One, signaling a shift away from using Kinect in gaming applications. However, the technology and sensors from Kinect have found applications in various industries beyond gaming.

Kinect for Xbox 360:

1. Structured Light Technology:
    - The Kinect for Xbox 360 uses structured light technology for depth sensing.
2. Lower Resolution RGB Camera:
    - The RGB camera on the Kinect for Xbox 360 has a lower resolution compared to the Kinect for Xbox One.
3. Skeletal Tracking:
    - The Kinect for Xbox 360 introduced skeletal tracking, allowing users to control games and interact with the console using body movements.

4. Fixed Tilt:
    - Unlike the Kinect for Xbox One, the Kinect for Xbox 360 does not have a motorized tilt mechanism for remote adjustments.
5. USB 2.0 Connectivity:
    - The Kinect for Xbox 360 connects via USB 2.0.
6. Original Kinect SDK:
    - The original Kinect for Xbox 360 had an official SDK that allowed developers to create applications using Kinect technology.

3.6.4 Cytron Motor Driver

Key Features and Specifications for cytron motor driver MDD10A

Key Features:

1. Dual-Channel Motor Driver:
    - Capable of independently controlling two brushed DC motors.
2. Voltage Range:
    - Designed to operate with a wide voltage range to accommodate different power supply configurations.
3. Motor Current:
    - Supports a specific maximum current for each motor channel. The current-handling capacity is an important factor in motor driver selection based on the motors being used.
4. PWM Speed Control:
    - Allows for speed control using Pulse Width Modulation (PWM) signals.
5. Direction Control:

- Supports bi-directional control for each motor, allowing forward and reverse movements.

6. Overcurrent Protection:
    - Built-in protection mechanism to prevent damage due to excessive current.
7. Thermal Protection:
    - Features thermal protection to prevent overheating of the motor driver.
8. Motor Brake Function:
    - Provides the option for a motor brake function, allowing the motor to be quickly brought to a stop.
9. Regenerative Braking:
    - Some motor drivers, including the MDD10A, may support regenerative braking, which redirects the energy from the motor back to the power supply.
10. Compact Form Factor:
    - Designed in a compact form factor, suitable for various applications where space is a consideration.

Specifications:

1. Operating Voltage:
    - Specifies the acceptable voltage range for the motor driver.
2. Maximum Motor Current:
    - Indicates the maximum current that each motor channel can handle.
3. PWM Frequency:
    - Specifies the PWM frequency range supported by the motor driver.
4. Input Voltage for Logic:

- The voltage level required for the logic input signals (e.g., PWM, direction control).

5. Current Sensing Voltage:
    - Specifies the voltage level proportional to the motor current that can be used for current sensing.
6. Operating Temperature Range:
    - Indicates the range of temperatures within which the motor driver can operate reliably.
7. Dimensions:
    - Provides the physical dimensions of the motor driver module.
8. Weight:
    - Specifies the weight of the motor driver, which can be relevant for certain applications.

a. comparison with motor driver L298 N

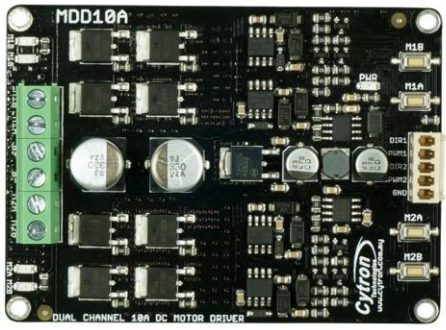


Figure 40. cytron driver

Cytron MDD10A:

1. Channels:
    - MDD10A: Dual-channel motor driver.
2. Voltage Range:
    - MDD10A: Operates with a wide voltage rang
3. Maximum Motor Current:
    - MDD10A: Supports a specific maximum current for each motor channel.
4. PWM Speed Control:
    - MDD10A: Allows for speed control using PWM signals.

5. Overcurrent Protection:
    - MDD10A: Built-in overcurrent protection to prevent damage due to excessive current.
6. Thermal Protection:
    - MDD10A: Features thermal protection to prevent overheating.
7. Motor Brake Function:
    - MDD10A: Provides the option for a motor brake function.
8. Regenerative Braking:
    - MDD10A: May support regenerative braking.

L298N:

1. Channels:
    - L298N: Dual H-bridge motor driver.
2. Voltage Range:
    - L298N: Typically used with a specific voltage range (e.g., 7V to 35V).
3. Maximum Motor Current:
    - L298N: Supports a specific maximum current for each motor channel.
4. PWM Speed Control:
    - L298N: Allows PWM-based speed control.
5. Overcurrent Protection:
    - L298N: Some versions may lack overcurrent protection. External circuitry may be needed for protection.
6. Thermal Protection:

- L298N: Some modules lack thermal protection. Care must be taken to prevent overheating.

7. Motor Brake Function:
    - L298N: Typically includes a motor brake function.
8. Regenerative Braking:
    - L298N: Does not inherently support regenerative braking.

3.6.5 ESP 32

Key Features:

Figure 41. ESP32

1. Dual-Core Processor:
    - Dual-core Tensilica LX6 microprocessors.
2. Wireless Connectivity:
    - Integrated Wi-Fi (802.11 b/g/n) with support for WPA/WPA2 security.
    - Integrated Bluetooth (Bluetooth Classic and Bluetooth Low Energy/BLE).
3. Peripheral Interfaces:
    - GPIO (General Purpose Input/Output) pins for digital and analog input/output.
    - UART (Universal Asynchronous Receiver-Transmitter).
    - I2C (Inter-Integrated Circuit).
    - SPI (Serial Peripheral Interface).
    - ADC (Analog-to-Digital Converter).
4. Memory:
    - Flash memory for program storage.
    - SRAM (Static Random-Access Memory) for variable storage.

a. Comparison ESP32 with other previous

ESP32:

1. Processor:
    - ESP32: Dual-core Tensilica LX6 Xtensa processors.
2. Wireless Connectivity:
    - ESP32: Integrated Wi-Fi (802.11 b/g/n) and Bluetooth (Bluetooth Classic and BLE).
3. Peripheral Interfaces:
    - ESP32: Expanded GPIO, UART, I2C, SPI, ADC, and more.
4. Memory:
    - ESP32: Comes with a larger range of flash memory options, often starting at 4MB.
5. Security:
    - ESP32: Enhanced security features, including hardware-accelerated cryptographic algorithms and secure boot.
6. Number of Cores:
    - ESP32: Dual-core, enabling more complex multitasking and parallel processing.
7. Power Consumption:
    - ESP32: Power consumption varies based on usage but generally efficient in low-power applications.
8. OTA Updates:
    - ESP32: Supports OTA updates with additional features.

ESP8266:

1. Processor:
    - ESP8266: Single-core Tensilica L106 Xtensa processor.
2. Wireless Connectivity:
    - ESP8266: Integrated Wi-Fi (802.11 b/g/n).
3. Peripheral Interfaces:
    - ESP8266: Limited GPIO, UART, I2C, and SPI.
4. Memory:
    - ESP8266: Typically comes with 512KB to 4MB of flash memory.
5. Security:
    - ESP8266: Basic security features.
6. Number of Cores:
    - ESP8266: Single-core.
7. Power Consumption:
    - ESP8266: Known for relatively low power consumption.
8. OTA Updates:
    - ESP8266: Supports Over-the-Air (OTA) updates.

3.7 batteries

3.7.1 lithium-ion batteries

4-cell lithium-ion battery key features and specifications

1. Voltage:

- Nominal Voltage: A 4-cell lithium-ion battery typically has a nominal voltage of 14.8 volts. This is achieved by connecting four individual cells in series, each contributing approximately 3.7 volts.

2. Capacity:

- Nominal Capacity: Measured in milliampere-hours (mAh) or ampere-hours (Ah), the nominal capacity of a 4-cell lithium-ion battery can vary. Common capacities range from 5000mAh to 10,000mAh, indicating the amount of charge the battery can store.

3. Chemistry:

- Lithium-Ion Chemistry: This battery employs lithium-ion chemistry, known for its high energy density and rechargeable nature. It uses lithium cobalt oxide, lithium manganese oxide, or other lithium-based materials in its construction.

4. Cell Configuration:

- Four Cells in Series (4S): The battery configuration involves connecting four individual lithium-ion cells in series. This arrangement allows the battery to achieve the desired voltage output.

5. Current Rating:

- Continuous Discharge Current: This represents the maximum current that the battery can consistently deliver without overheating. For a 4-cell lithium-ion battery, a continuous discharge current of at least 10 amperes or higher is typical.

6. Connector Type:

- Standard Connector: The battery is equipped with a connector suitable for the specific application. The connector type may vary, and it's essential to ensure compatibility with the system it powers.

7. Physical Dimensions:

- Compact Form Factor: The physical dimensions of the 4-cell lithium-ion battery are designed to be compact, balancing the need for high energy density with practical considerations such as weight and size.

8. Weight:

- Lightweight Design: Lithium-ion batteries are known for their relatively low weight, making them suitable for applications where minimizing overall weight is crucial, such as in portable electronics or electric vehicles.

9. Cycle Life:

- Long Cycle Life: Lithium-ion batteries generally offer a long cycle life, capable of undergoing hundreds to thousands of charge-discharge cycles before experiencing a notable reduction in capacity. This durability ensures sustained performance over time.

10. Safety Features:

- Battery Management System (BMS): A critical safety feature, the BMS monitors individual cell voltages, prevents overcharging, over-discharging, and manages temperature. This enhances the safety and longevity of the battery.

11. Charging Specifications:

- Voltage and Current Requirements: The battery's recommended charging voltage and current should be followed to ensure safe and efficient charging. The charging specifications depend on the specific chemistry and design of the lithium-ion cells used.

3.7.2 comparison between lithium-ion battery and other

1. Lead-Acid Batteries:

Energy Density:

When it comes to energy density, lithium-ion batteries outshine lead-acid batteries. The latter are like the reliable workhorses, sturdy but not as efficient in storing energy compared to the sleek lithium-ion counterparts.

Cycle Life:

In the marathon of life cycles, lithium-ion batteries take the lead. Lead-acid batteries may tire out more quickly, especially in deep discharge cycles. It's like comparing a sprinter to a long-distance runner.

Weight:

Picture lead-acid batteries as the heavyweight champions—solid but heavy. Lithium-ion batteries step in as the featherweights, offering the same punch in a lighter package. This makes lithium-ion batteries a knockout choice for weight-sensitive applications like electric vehicles.

2. Nickel-Cadmium (NiCd) Batteries:

Energy Density:

Lithium-ion batteries enter the scene with a higher energy density, stealing the spotlight from the nickel-cadmium performers. Lithium-ion batteries, with their lighter and more compact nature, emerge as the stars of the show.

Cycle Life:

The race in life cycles is a close one. While both lithium-ion and nickel-cadmium batteries have their merits, lithium-ion batteries often take the lead due to their higher energy density and freedom from toxic cadmium.

Memory Effect:

Nickel-cadmium batteries are like the forgetful actors, prone to memory effect. In this theater, lithium-ion batteries play it cool with a lower memory effect, retaining their performance even if not fully discharged before recharging.

3. Nickel-Metal Hydride (NiMH) Batteries:

Energy Density:

Lithium-ion batteries take center stage with a higher energy density, showcasing their ability to pack more power in a smaller and lighter package. NiMH batteries play solid supporting roles but can't match the lithium-ion spotlight.

Cycle Life

: In the longevity saga, lithium-ion batteries emerge as the marathon runners, outlasting NiMH batteries. The latter, while dependable, may find themselves overshadowed in the long run.

Memory Effect:

NiMH batteries navigate the memory effect storyline with a milder impact compared to their NiCd relatives. Lithium-ion batteries, as the protagonists, maintain their low memory effect reputation.

4. Solid-State Batteries:

Safety:

Solid-state batteries emerge as the safety crusaders in this narrative. By replacing the liquid electrolyte with a solid counterpart, they significantly reduce the risk of thermal runaway, setting a new standard for safety.

Energy Density:

Solid-state batteries unveil a potential for higher energy density, promising more capacity for the same volume. It's a teaser for a future where energy storage reaches new heights.

Cost:

In the financial plot, solid-state batteries play the role of the high-budget production—more expensive to produce than traditional lithium-ion batteries. It's the price of being on the cutting edge.

3.7.3 why lithium-ion battery

1. High Energy Density:

- Lithium-ion batteries have a high energy density, meaning they can store a significant amount of energy in a relatively small and lightweight package. This is crucial for applications where weight and space are important considerations, such as in autonomous vehicles.

2. Rechargeable:

    - Lithium-ion batteries are rechargeable, allowing for multiple charge-discharge cycles. This makes them cost-effective and suitable for applications where a reliable and rechargeable power source is required.

3. Long Cycle Life:

    - Lithium-ion batteries typically have a good cycle life, capable of undergoing hundreds to thousands of charge-discharge cycles before experiencing a significant reduction in capacity. This makes them durable and suitable for long-term use.

4. Low Self-Discharge Rate:

    - Lithium-ion batteries have a low self-discharge rate, meaning they retain their charge for longer periods when not in use. This is beneficial for applications that may have intermittent use, such as autonomous vehicles.

5. Versatility:

    - Lithium-ion batteries are available in various shapes and sizes, providing flexibility for different device designs. This versatility allows them to be adapted to the specific space constraints of a given application.

6. Wide Range of Applications:

    - Lithium-ion batteries are used in a wide range of applications, from small electronics like smartphones and laptops to larger systems like electric vehicles and robotics. Their versatility makes them suitable for various power needs.

7. Fast Charging Capability:

    - Many lithium-ion batteries support fast charging, allowing for quicker recharge times. This is advantageous for applications where downtime needs to be minimized.

3.7.4 calculation

- Battery Capacity (Wh)=Total Power Consumption (W)×Desired Runtime (hours)×Safety Margin

Let's do a calculation with a 50% safety margin for a 1.5-hour runtime:

- Battery Capacity (Wh)=65W×1.5 hour×1.5
- WhBattery Capacity (Wh)=146.25 Wh
- Battery Capacity (Ah)=Battery Capacity (Wh)/Voltage (V)
- Battery Capacity (Ah)=146.25 Wh/12V
- Battery Capacity (Ah)=12.1875 Ah

For this calculation we need battery 12V 15Ah lithium-ion battery 3C

3.8 Electrical Circuits

3.8.1 connection

PCB circuit using EAGLE:

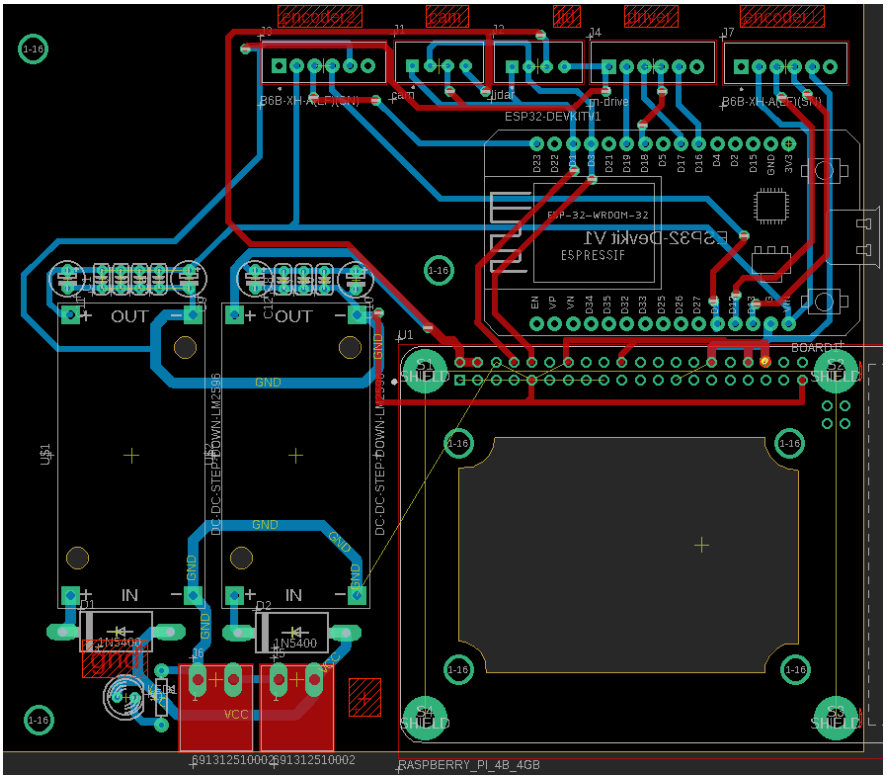


Figure 42. PCP on top

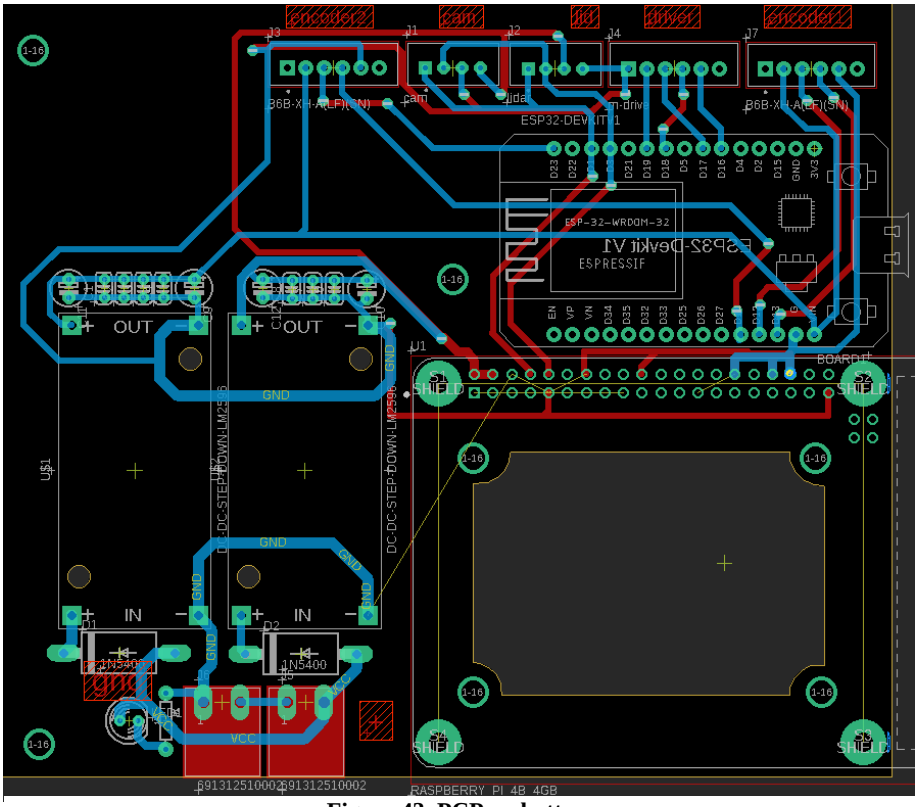


Figure 43. PCP on bottom

3.3.2 Schematic diagram:

Using Autodesk Eagle

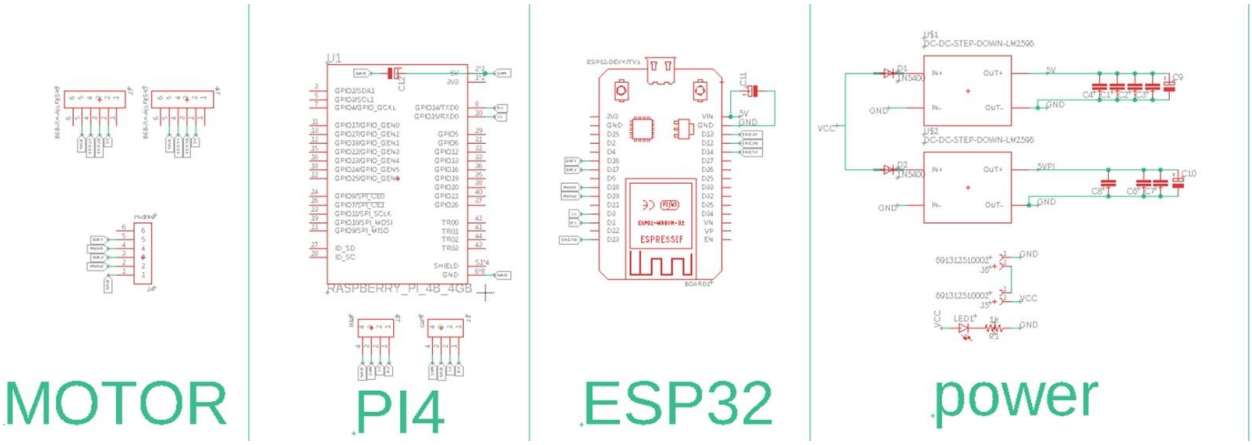


Figure 44. Schematic

3.4 Cost of component

Table 5 Cost of components

Components	Prices	Number
DC Motor GA25-370 with Encoder 2kg 280RPM 12V with Bracket	1000.00 EGP	2
Raspberry pi 4 GB module B	5000.00 EGP	1
LIDAR A1 M8	7000.00 EGP	1
KINECT CAMERA XBOX ONE	2000.00 EGP	1
CYTRON MOTOR DRIVER MDD 10 A	1950.00 EGP	1
ESP 32	500.00 EGP	1
TOTAL	**18450 EGP**	

3.5 Conclusion:

By integrating these components thoughtfully, the system can achieve powerful and versatile functionality. The Raspberry Pi 4 serves as the control hub, coordinating inputs from the LiDAR A1M8 and Kinect Camera for environmental sensing, driving motors through the Cytron Motor Driver MDD10A, and managing wireless communication via the ESP32. A well-designed 12V battery system and a custom PCB ensure efficient power management and reliable operation. Effective thermal management strategies will maintain system stability under various operating conditions. This comprehensive approach results in a robust embedded system suitable for a wide range of applications.

4. Chapter IV: software design and simulation

4.1 Pid control:

In this project the procedure of analyzing and measuring the parameters is spilt into two parts part one the connection of the DC-Motor with the other hardware components and part two the software using a series of programs.

Controlling and testing the system

First we started by uploading the Arduino code on the Arduino UNO chip This code main function is that there are a set of variables that stands for the pins connection on the Arduino where pin number seven used for setting the direction either clock-wise or antilock-wise and pin number two and pin number three are used for the encoder phases phase-A and Phase-B respectively and pin number five as the Motor Enable and the 5v pin is connected to the positive terminal of the encoder and all is connected to a common ground. The H-bridge is connected with pins number seven and is also connected with the motor supplying it with voltage and ground throw two pins. Where pins number seven is responsible for sending the direction of motion from the serial monitor throw the Arduino and then to the motor throw the H-bridge and sending the value of the PWM throw pin number five to the H-bridge then to the motor to control the speed of the motor and it is sent in the form of voltage. Where pin number two and pin number three support interrupt peripheral and pins are digital input and output pins and pin five support PWM.

4.1.1 Modeling using MATLAB

Using this app gives you big benefits with modeling and simulating a big and complex systems by using Simulink we can simulate our system in form of blocks as shown in figure

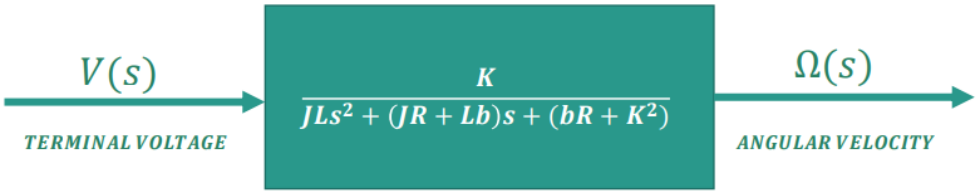


Figure 45 Transfer Function of DC Motor

So, it could be considered the green block as our real motor that enable as interact with it take results and plotting it and this make the optimizing and analysis processes easier and faster

4.1.2 Full control using Arduino:

Using Arduino ide by including a set of libraries (timerone.h ,encoder.h)

Where timerone: is one of the libraries written to take advantage of the 16-bit counter that comes with the Atmega328.

Encoder library: allows the code to integrate with the encoder where the Encoder counts pulses from quadrature encoded signals, which are commonly available from rotary knobs, motor or shaft sensors and other position sensors.

Then we defined the (port serial) to make the code editable.

Then we initialized a set of variables that the code would use as locations representing the connections between the system to draw the pass for the signals that we would sent throw the serial monitor to study the behavior of the motor.

Then there are a set of variables which are (plus count, plus per revolution, time)which are used to calculate the speed in RPM using this equation

SpeedRPM = ((PulseCount / timerInterval) / PulsesPerRevolution) * 60.0

And these values are always updated throw out the program.

And then we assigned the input and output pins where we deduced by the process of trial and error when we sent a high signal to one direction, we must send low to the other direction for the motor to operate

Table 6 Motor Rotation

Dir-1	Dir-2	Motor rotation
0	0	Stop rotating
0	1	Clock-wise
1	0	Anti-clock wise
1	1	Stop rotating

A family of application-defined functions that are called whenever there is data to be read from a serial peripheral. Then we initialized a variable called command to integrate between the system and the serial plotter.

And by using switch case it allowed us to control the starting and stopping of the motor and its speed by changing the PWM and its direction.

Getting measurements:

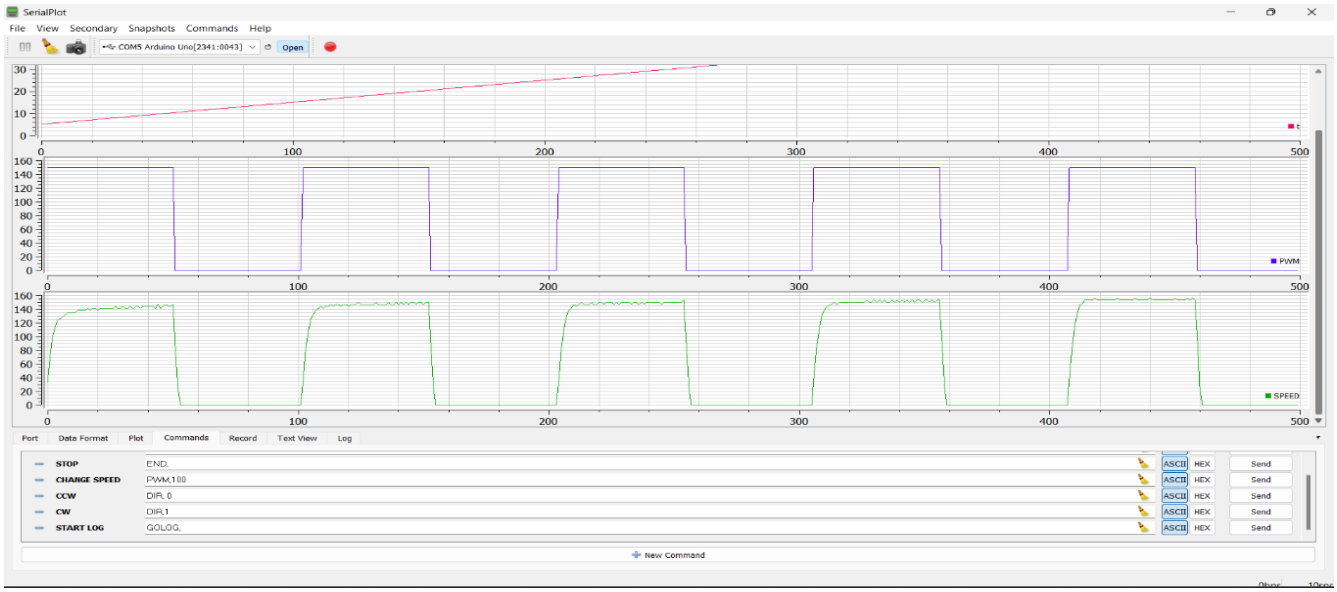


Figure 46 Serial Plot results

4.1.3 Explain the concept of estimating Parameter

When you know nothing for a system Parameter you can estimate their parameters if you Know some of their features which you deduce by analyzing the system and modeling it mathematically or by using MATLAB. One of these important features is the transfer function of the system which we mention in details in simulation 2 partition and we can use anther similar system to get its knowing parameter as initial assumed parameter and by getting a set of measured data from our actual output which is a result for a set of input data then we can use parameter estimator which is an app in Simulink to estimate our real system parameters

4.1.4 Prepare the logging

for this purpose, we control our system in particular way to get this data and using the code mentioned in the previous partition with additional point to fit this way. The way we are talking about is to turn the motor on then off for equal periods of time

and the additional point for the code is to control the number of cycle and the number of samples send by the microcontroller and for this we define some variables to show (the sample count, the switching period, the number of samples, the overall sample count, the overall number of samples) and by using these variables in our code we could manage how long the on period will take and how many of this period will happen by using the following calculations

switchingPeriod = 7; (determine how long the half period time)

numOfSamples = switchingPeriod / timerInterval;

overallNumOfSamples = numOfSamples * switchingPeriod * 2 + numOfSamples;

And the main concept for our program is the interaction between the encoder which represents the feedback of the system and the timer peripheral and the interrupt peripheral so when we attach an interrupt to the pulse count which we get from the encoder we can use it to calculate the RPM and when we pass this RPM to function which is attached to an ISR and use the timer peripheral to control when this ISR will fire So this way we could plot our RPM values which we consider the output set of data and this output set of data changes by the change of the input set of data which are the PWM values which are translated by the H-bridge to voltage and then transmit it to the motor. As we get this data, we could now call them a logging data that we will export in the form of a csv file from serial plotter which we will use in the next step for parameters estimation. The following figures show how the serial plotter show the data.

Figure 47 (PWM) graph

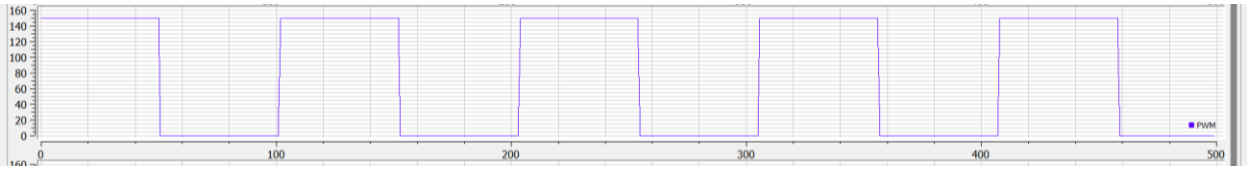


Figure 48 (speed) graph

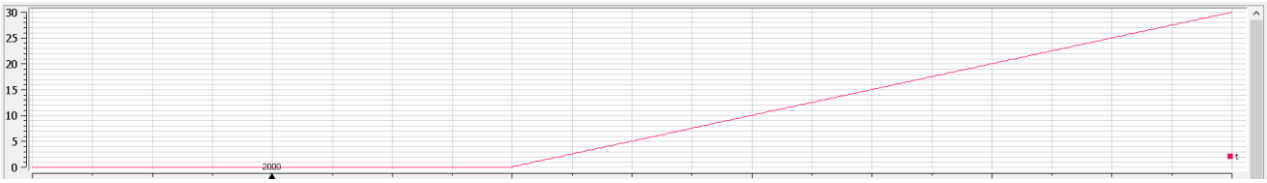


Figure 49 (Time) graph

4.1.5 Logging data to MATLAB using this script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  SCRIPT : Import DC motor log file to MATLAB workspace                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Read log csv file
file_path = './log/log.csv';
motor_log = readtable(file_path);

%% Store each column in a variable
t = motor_log.t;
speed = motor_log.SPEED;
pwm = motor_log.PWM;

%% export to mat file
save('motor_log','t','pwm','speed');v
```

Using these lines, we imported the dc motor log file to the MATLAB by writing the pass for the csv file that contains the measured data that we took from the motor

Getting the parameters

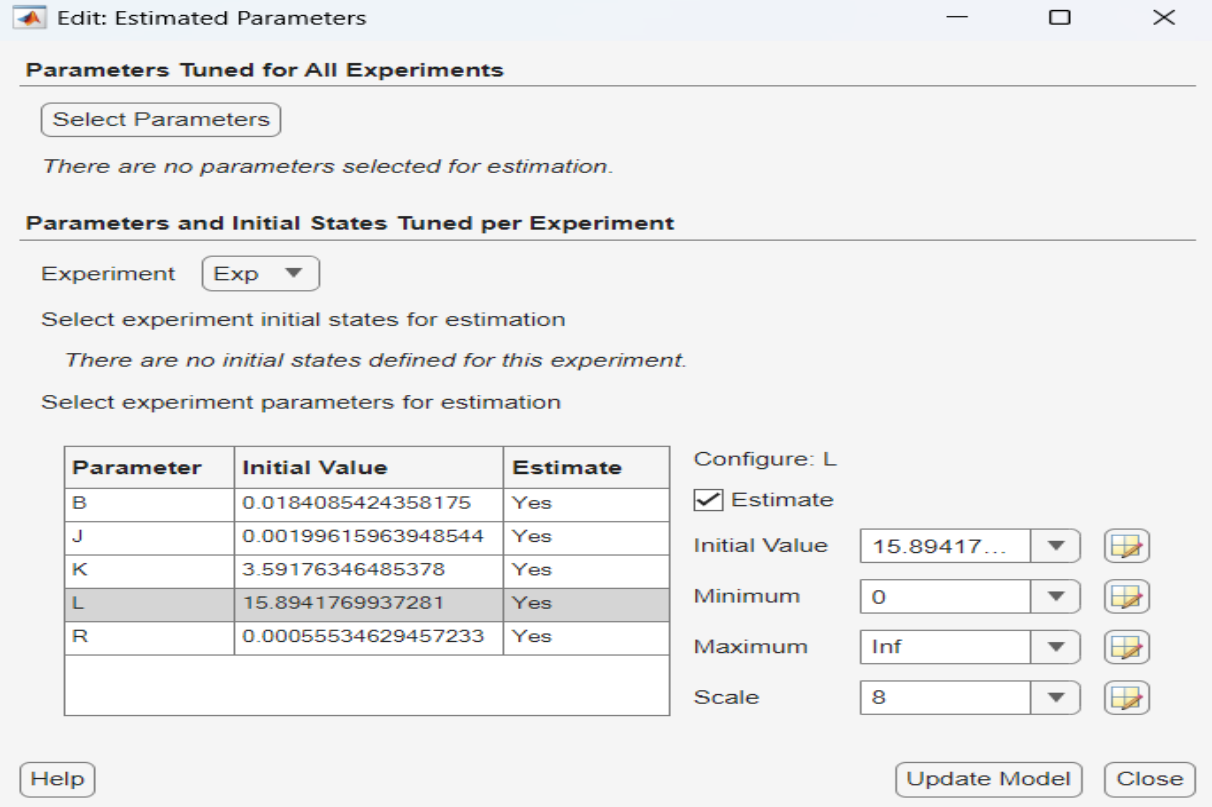
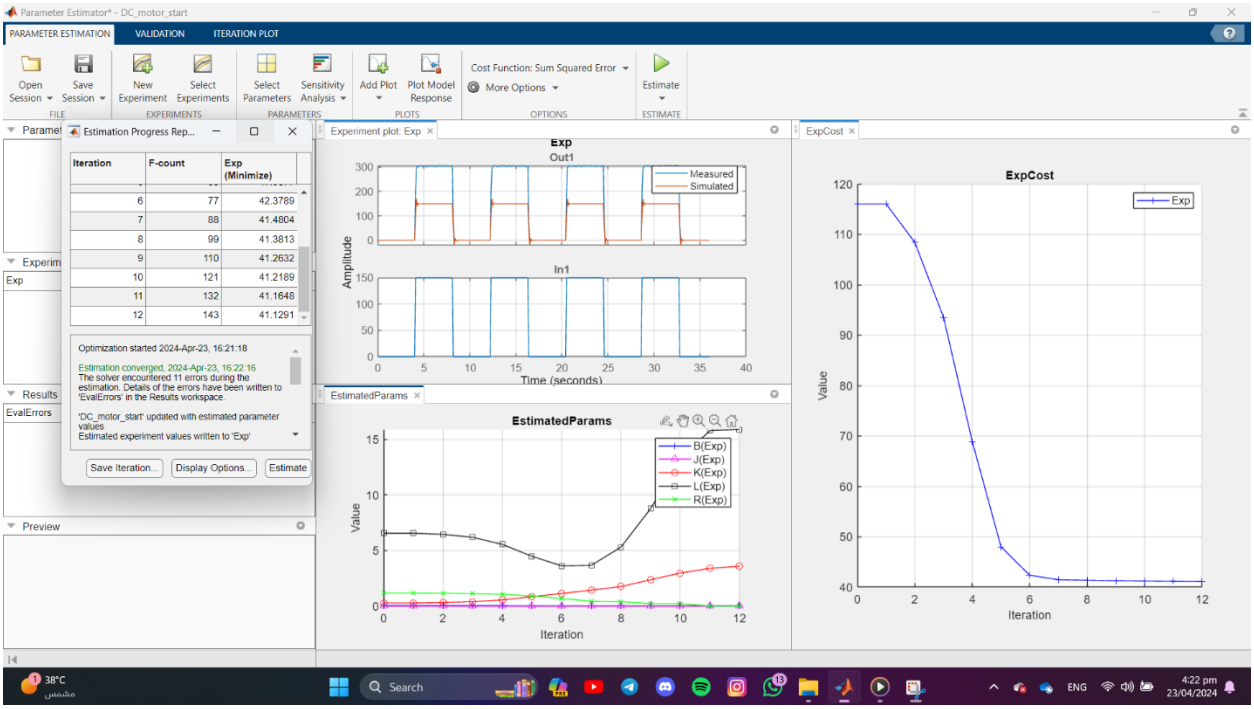


Figure 50 Estimated Parameters

Getting the parameters throw the process of tuning where the MATLAB starts to tune using iteration to the assumed data to reach as close as possible to the measured data as shown in the figure below.



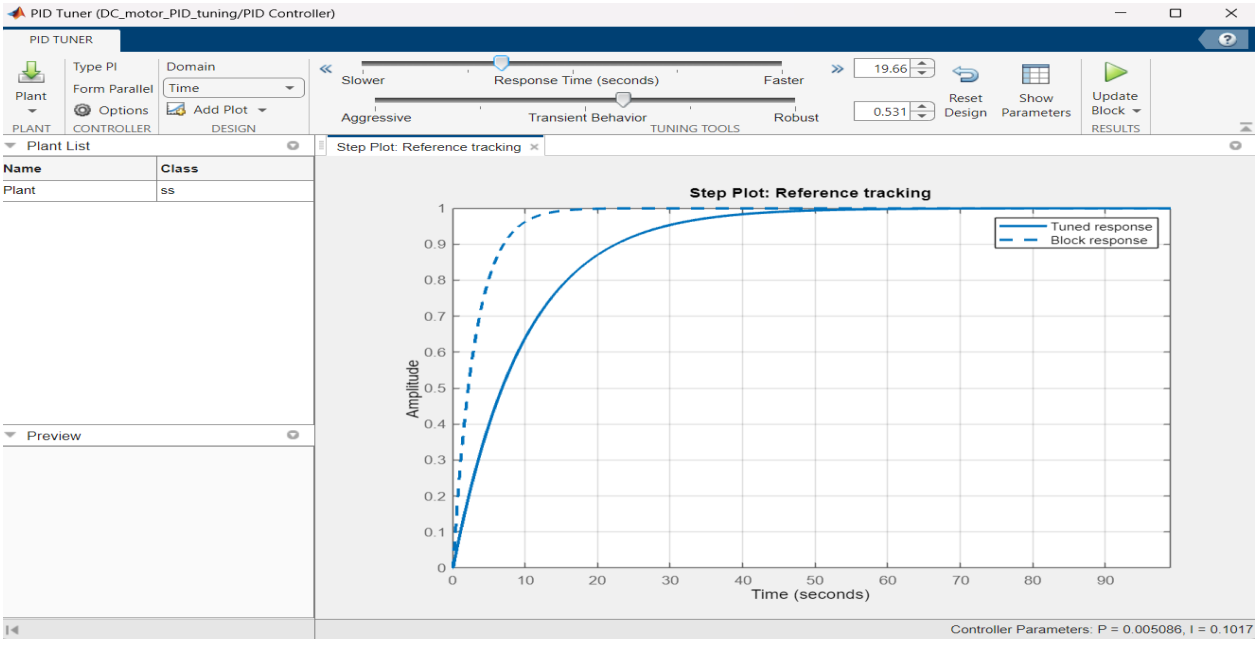


Figure 51 Step Plot: Reference tracking

Finally, we update the block to get the parameters of Kp, Ki, Kd

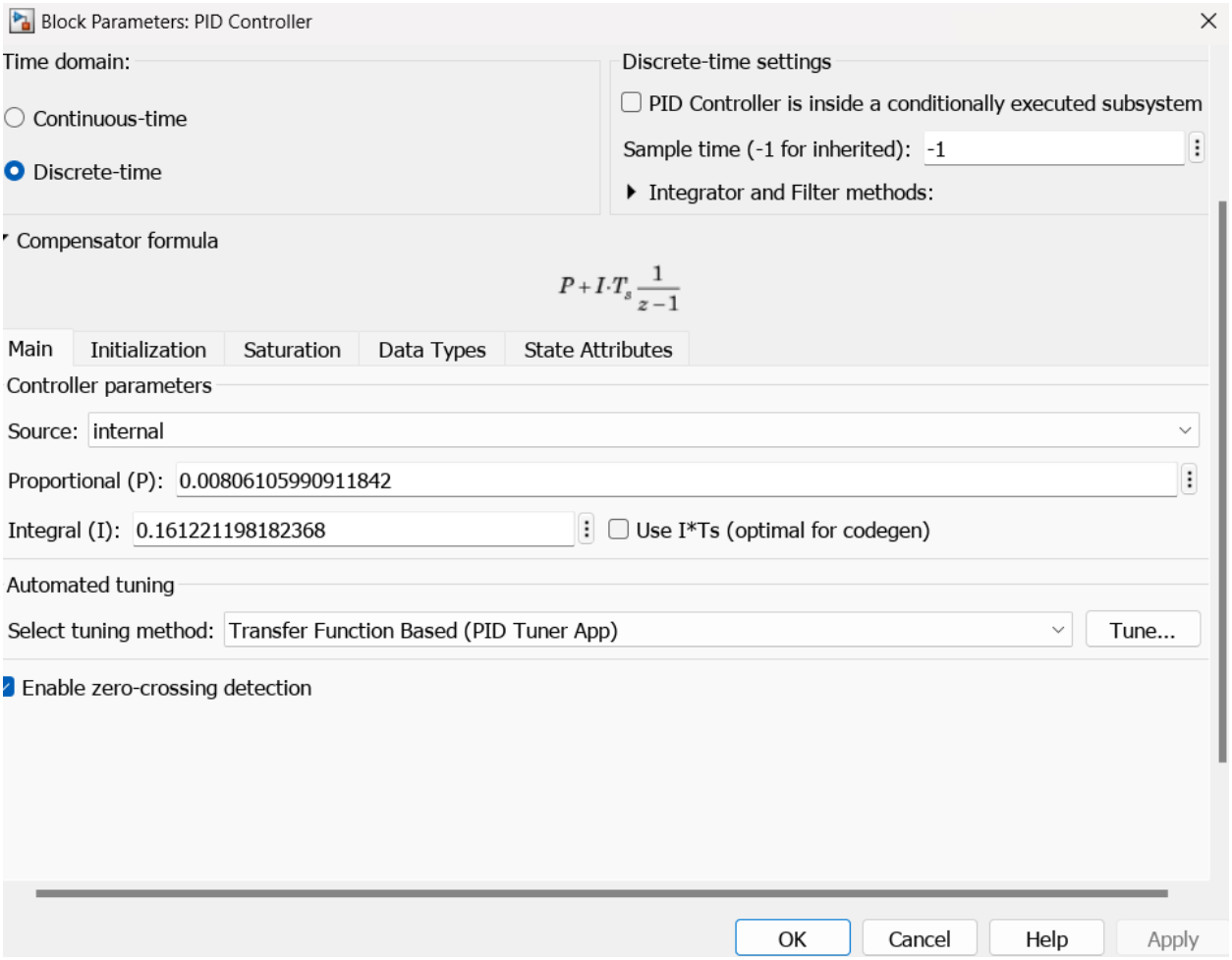


Figure 52 PID controller

4.2 Software block diagram of self-driving cars:

In this section, we will discuss a basic software block diagram of a self-driving car that was in DARPA Challenge:

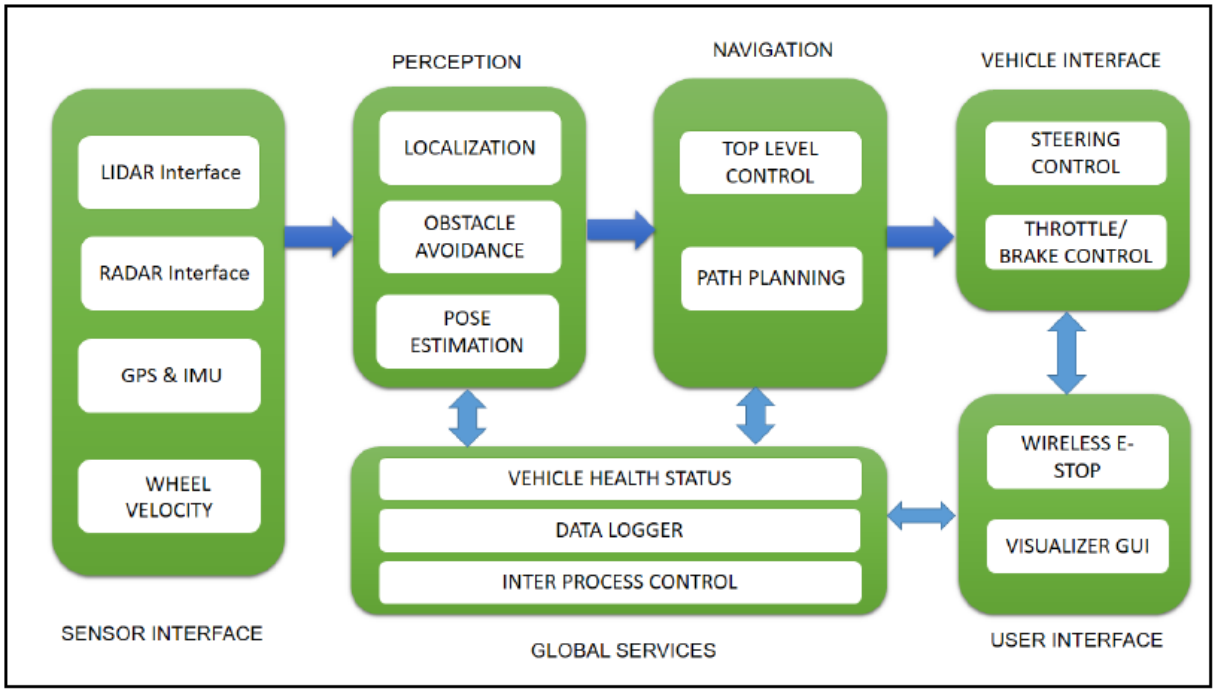


Figure 53 Software block diagram

(**IPC**) or shared memory. ROS messaging middleware is a perfect fit in this scenario. In DARPA Challenge, they implemented a publish/subscribe mechanism to do these tasks. One of the IPC library developments by MIT for 2006 DARPA challenge was **Lightweight Communications and Marshalling** (LCM).

Sensor interface modules: As the name of the module indicates, all the communication between the sensors and the vehicle is done in this block. The block enables us to provide the various kinds of sensor data to all other blocks. The main sensors include LIDAR, camera, radar, GPS, IMU, and wheel encoders.

Perception modules: These modules perform processing on perception data from sensors such as LIDAR, camera, and radar and segment the data to find moving and static objects. They also help localize the self-driving car relative to the digital map of the environment.

Navigation modules: This module determines the behavior of the autonomous car. It has motion planners and finite state machines for different behaviors in the robot.

Vehicle interface: After the path planning, the control commands, such as steering, throttle, and brake control, are sent to the vehicle through a **drive-bywire** (**DBW**) interface. DBW basically works through the CAN bus. Only some vehicles support the DBW interface. Examples are the Lincoln MKZ, VW Passat Wagon, and some models from Nissan.

User interface: The user interface section provides controls to the user. It can be a touch screen to view maps and set the destination. Also, there is an emergency stop button for the user.

Global services: This set of modules helps log the data and has time stamping and message-passing support to keep the software running reliably.

4.3 why ROS

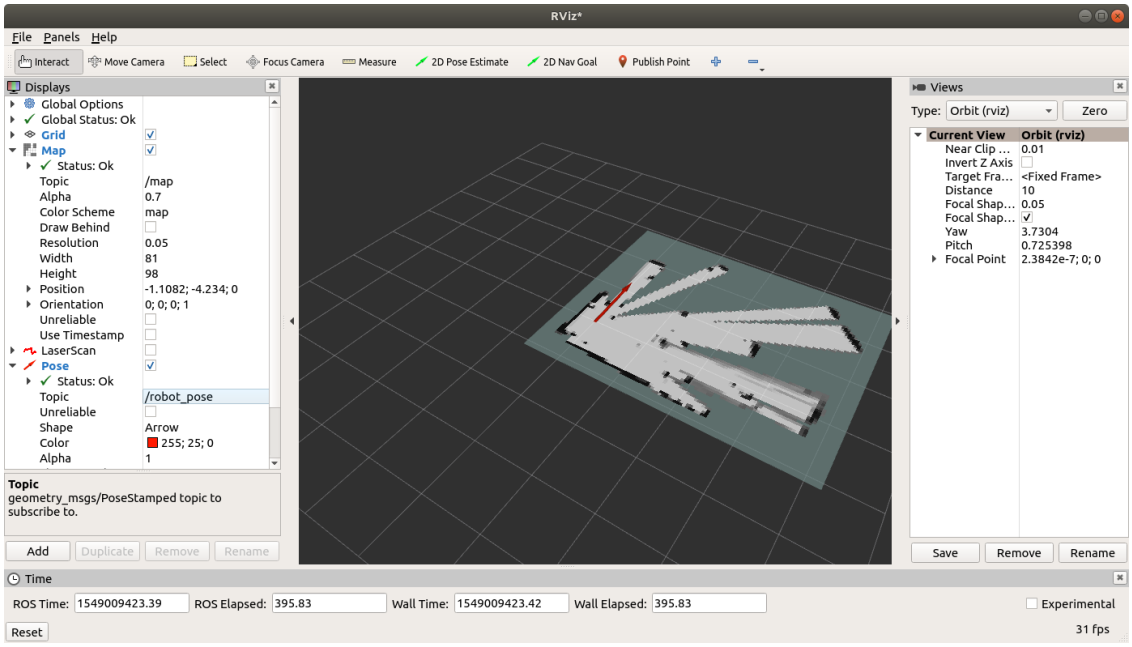
Modern robotic systems leverage the ecosystem and modularity of the Robot Operating System (ROS) to enhance the development environment and enable realistic simulations. ROS provides a flexible and robust framework that allows developers to create complex robotic systems with ease. This framework provides a wide range of tools and libraries that can be used to develop various robot applications.

ROS has a large and active community consisting of researchers, developers, and enthusiasts. It benefits from a wide range of packages and libraries created by the community, making it simpler to use existing solutions for different robotics tasks. Additionally, ROS supports simulation software like Gazebo, which expands its features.

ROS 2 Foxy Fitzroy, commonly known as ROS Foxy, is a significant release in the ROS 2 series, offering a robust, long-term support (LTS) version suitable for developing and deploying complex robotic applications. As an LTS release, ROS Foxy is designed to provide stability and reliability with extended maintenance and support, making it ideal for long-term projects. It introduces several key features such as the use of DDS (Data Distribution Service) for enhanced communication performance and flexibility, cross-platform compatibility supporting Linux, Windows, and macOS, and improved security mechanisms with Secure ROS (SROS 2). Additionally, ROS Foxy enhances real-time capabilities, crucial for applications demanding precise timing and synchronization. The managed node lifecycle feature allows for better control over node states, and the enhanced API and tooling improve usability, making development more efficient. These advancements position ROS Foxy as a powerful tool for modern robotics, catering to diverse use cases from industrial automation to autonomous vehicles.

4.3.1 Rviz:

Rviz, abbreviation for ROS visualization, is a powerful 3D visualization tool for ROS. It allows the user to view the simulated robot model, log sensor information from the robot's sensors, and replay the logged sensor information. By visualizing what the robot is seeing, thinking, and doing, the user can debug a robot application from sensor inputs to planned (or unplanned) actions. Rviz displays 3D sensor data from stereo cameras, lasers, Kinects, and other 3D devices in the form of point clouds or depth images. 2D sensor data from webcams, RGB cameras, and 2D laser rangefinders can be viewed in rviz as image data. If an actual robot is communicating with a workstation that is running rviz, rviz will display the robot's current configuration on the virtual robot model. ROS topics will be displayed as live representations based on the sensor data published by any cameras, infrared sensors, and laser scanners that are part of the robot's system. This can be useful to develop and debug robot systems and controllers. Rviz provides a configurable Graphical User Interface (GUI) to allow the user to display only information that is pertinent to the present task. In this chapter, we will use rviz to visualize our progress in creating a two-wheeled robot model. Rviz will use the URDF file that we create for our robot and display the visual representation. We will begin by checking whether rviz has been downloaded and installed on your system.



4.3.2 Gazebo simulator:

Gazebo is a 3D simulator that provides robots, sensors, environment models for 3D simulation required for robot development, and offers realistic simulation with its physics engine. Gazebo is one of the most popular simulators for open source in recent years, and has been selected as the official simulator of the DARPA Robotics Challenge7 in the US. It is a very popular simulator in the field of robotics because of its high performance even though it is open source. Moreover, Gazebo is developed and distributed by Open Robotics which is in charge of ROS and its community, so it is very compatible with ROS.

■■ Dynamics Simulation: In the early days of development, only ODE (Open Dynamics Engine) was supported. However, since version 3.0, various physical engines such as Bullet, Simbody, and DART are used to meet the needs of various users.

■■ 3D Graphics: Gazebo uses OGRE (Open-source Graphics Rendering Engines), which is often used in games, not only the robot model but also the light, shadow and texture can be realistically drawn on the screen.

■■ Sensors and Noise Simulation: Laser range finder (LRF), 2D/3D camera, depth camera, contact sensor, force-torque sensor and much more are supported and noise can be applied to the sensor data similar to the actual environment.

■■ **Plug-ins**: APIs are provided to enable users to create robots, sensors, environment control as a plug-in.

■■ Robot Model: PR2, Pioneer2 DX, iRobot Create, and TurtleBot are already supported in the form of SDF, a Gazebo model file, and users can add their own robots with an SDF file.

■■ TCP/IP Data Transmission: The simulation can be run on a remote server and Google's Protobufs, a socket-based message passing is used.

■■ Cloud Simulation: Gazebo provides cloud simulation CloudSim environment for use in cloud environments such as Amazon, Softlayer, and OpenStack.

■■ Command Line Tool: Both GUI and CUI tools are supported to verify and control the simulation status.

The latest version of Gazebo is 8.0, and just five years ago, it was 1.9. The current version is the adopted as a default application of the ROS Kinetic Kame used in this book. If ROS is installed as instructed in '3.1 ROS installation', Gazebo can be used without additional installation. Now let's run Gazebo. If there are no problems, you can see that Gazebo is running as shown in Figure 10-15. For now, Gazebo can be seen as an independent simulator as it is not related to ROS.

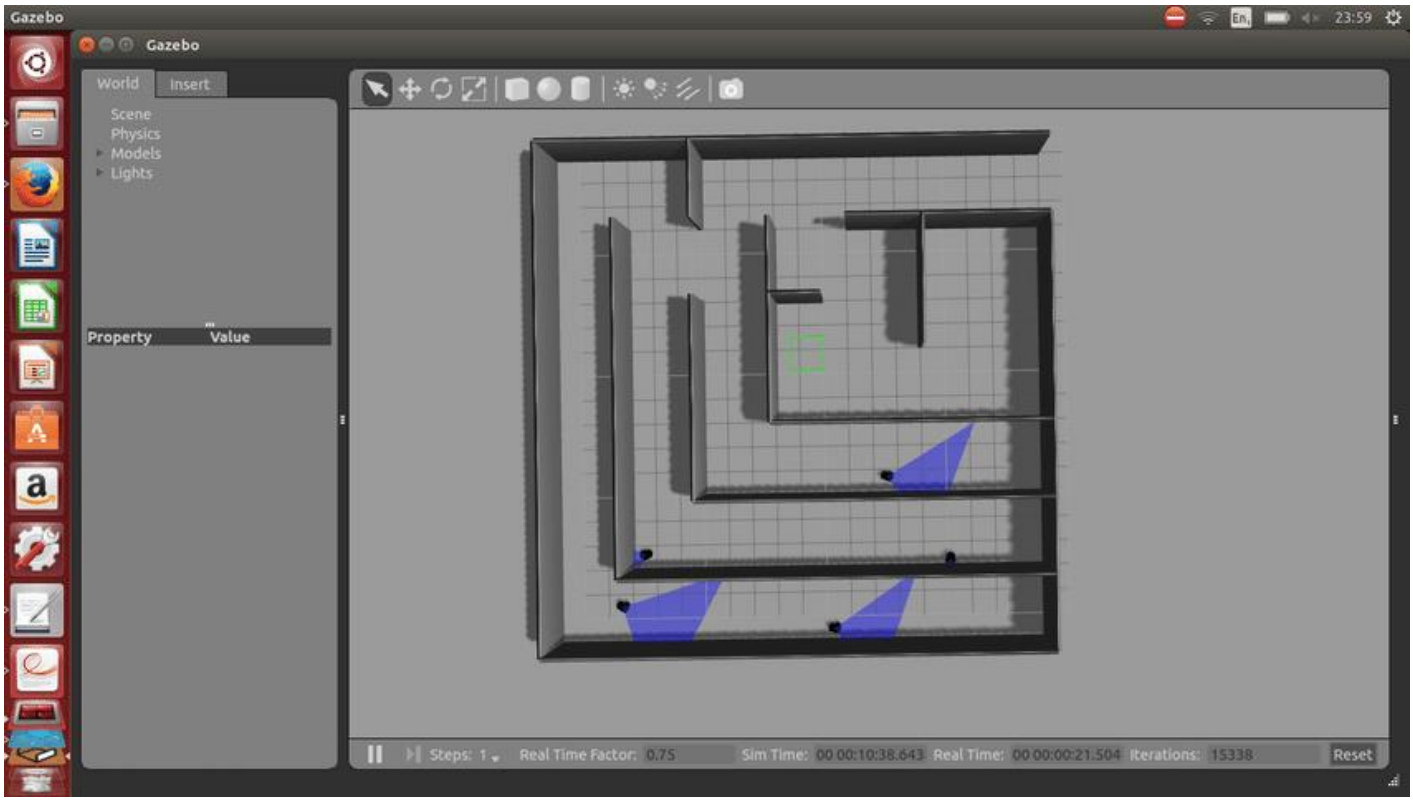


Figure 54 Gazebo

4.3.3 Debugging Tools in ROS2

ROS 2 provides several debugging tools and utilities to assist in the development and debugging of robotic applications. These tools help diagnose issues, monitor system behavior, and analyze the performance of ROS 2 components. Here are some commonly used debugging tools in ROS 2:

1. rqt_console: rqt_console is a graphical tool that displays log messages generated by ROS nodes. It allows users to monitor and filter log messages based on severity levels, node names, or specific topics. rqt_console helps identify errors, warnings, and other informative messages generated during the execution of ROS nodes.

2. rqt_graph: rqt_graph provides a visual representation of the ROS computation graph. It displays the nodes, topics, and connections between them. This tool is useful for inspecting the structure and communication patterns within a ROS system, helping to identify missing connections or unexpected interactions.

3. RViz: As mentioned earlier, RViz is a powerful visualization tool in ROS 2. It can be used for debugging purposes by visualizing sensor data, robot states, and other relevant information. RViz allows users to inspect the behavior of robotic systems, validate sensor readings, and debug complex robot behaviors in a 3D environment.

4. ROS2 topic echo: The `ros2 topic echo` command-line tool enables users to subscribe to a specific topic and display the messages being published on that topic. It is useful for observing the content and frequency of messages being exchanged between nodes, helping to identify data inconsistencies or unexpected behavior.

5. ROS2 node info: The `ros2 node info` command-line tool provides information about a specific ROS node. It displays details such as the list of topics, services, and actions that the node advertises or subscribes to. This tool helps to verify the configuration and connections of a node in the ROS system.

6. ROS2 bag: The `ros2 bag` command-line tool allows users to record and play back ROS topics, capturing data streams for offline analysis. It enables

users to record sensor data, logs, and other messages during runtime and later replay them for debugging or analysis purposes.

7. ROS Debugging Tools: Apart from ROS 2-specific tools, general debugging techniques and tools can also be used in ROS 2 development. These include using standard debugging tools like GDB (GNU Debugger) or integrated development environments (IDEs) that provide debugging capabilities for C++ or Python code.

8. These debugging tools and utilities help ROS 2 developers diagnose issues, analyze system behavior, and ensure the proper functioning of robotic applications. They facilitate the identification of errors, monitoring of ROS components, and verification of data exchanged between nodes, leading to more effective debugging and improved system performance.

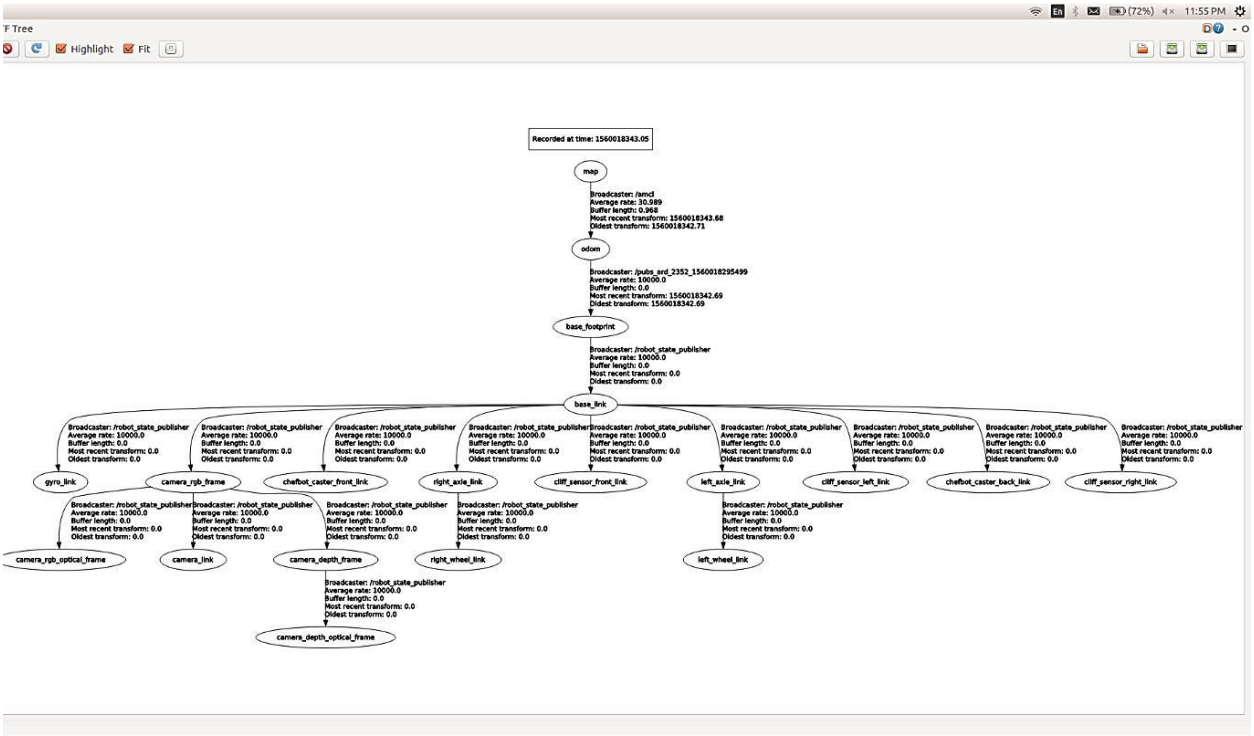


Figure 55 ROS rqt tool

4.4 Algorithms

- Set up and initialize ROS nodes for perception, planning, control, and simulation.
- Capture data from the camera and LiDAR sensors.
- Combine information from the camera and LiDAR for a more comprehensive perception of the environment.
- Determine the current position of the vehicle using sensor fusion and localization algorithms.
- Use computer vision and LiDAR processing algorithms to detect and classify objects in the environment.
- Make decisions based on the local path, detected objects, and other contextual information.
- Generate control commands for the vehicle's actuators based on the decisions made.
- Publish relevant data (sensor fusion, localization, object detection, paths, decisions, control commands) to ROS topics for communication between different nodes.
- Update the Gazebo simulation environment based on the control commands.

4.5 pseudo code

```
# Initialization
Initialize ROS nodes for perception, planning, control, and simulation
while True:
# Perception
camera_data = capture_camera_data()
lidar_data = capture_lidar_data()
# Sensor Fusion
fused_data = sensor_fusion(camera_data, lidar_data)
# Localization
current_position = localize(fused_data)
# Object Detection
detected_objects = object_detection(fused_data)
# Path Planning
global_path = plan_global_path(current_position)
local_path = plan_local_path(global_path, detected_objects)
# Decision Making
decision = make_decision(local_path, detected_objects)
# Control
control_command = generate_control_command(decision)
# ROS Communication
publish_data_to_ros_topics(fused_data, current_position, detected_objects, global_path,
local_path, decision, control_command)
# Simulation in Gazebo
```

4.6 URDF robot

The unified robotics description format (URDF) is an extensible markup language (XML) file type that includes the physical description of a robot. It is essentially a 3-D model with information around joints, motors, mass, etc. The URDF files are then run through the Robot Operating System (ROS). The data from the URDF file informs the human operator what the robot looks like and is capable of before they begin operating the robot.

We made our URDF to be used in the simulation and RViZ, added to the robot base is the camera and lidar.

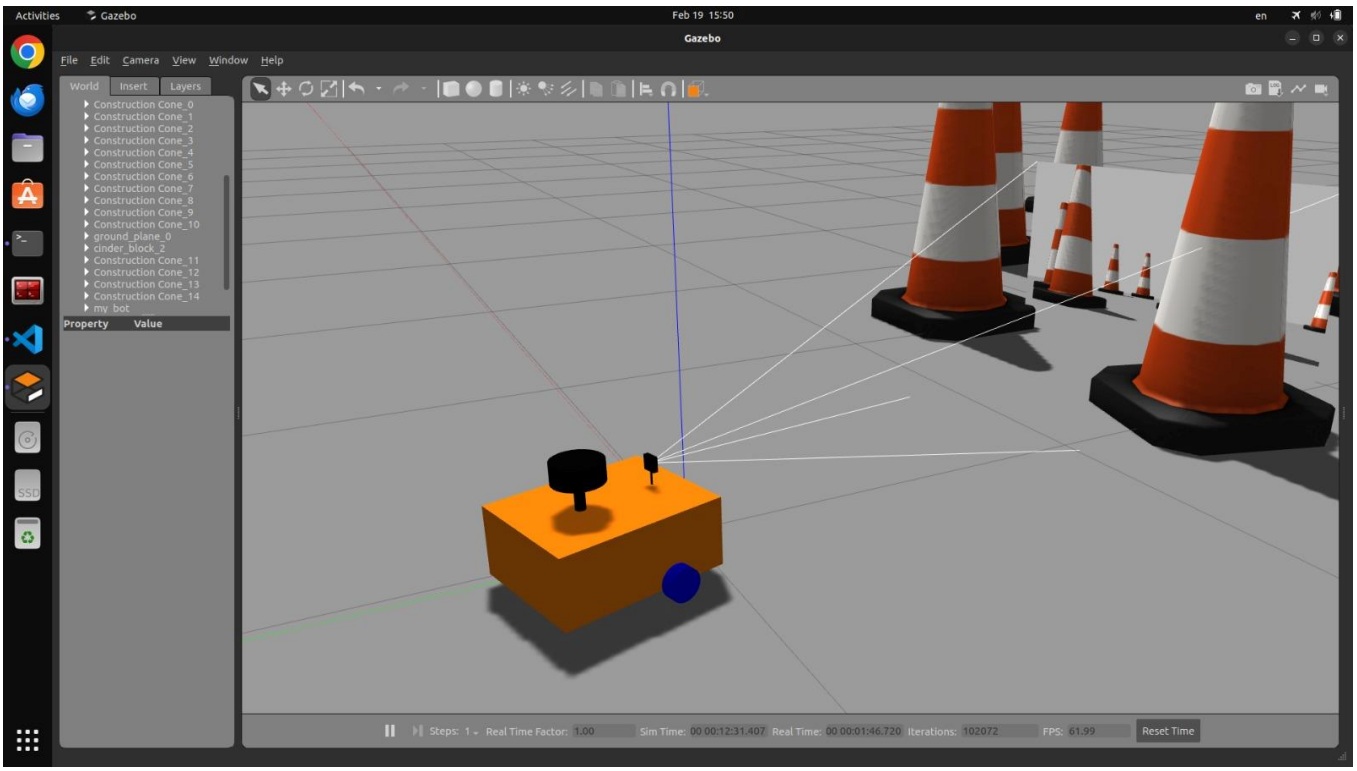


Figure 36. URDF with camera views

4.7 Simulation

4.7.1 Camera:

The camera Module got added to the URDF and described in Gazebo. It shows the environment while it is being navigated, and this can be added o RViz.

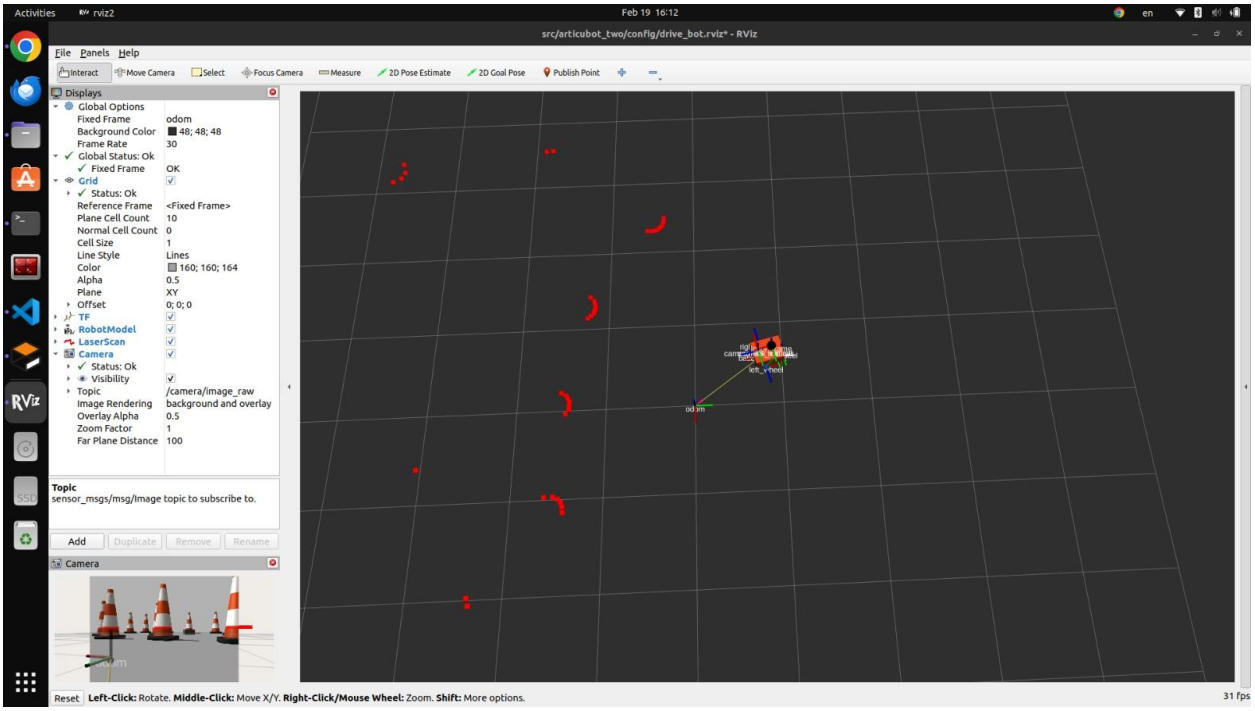


Figure 37. camera view in RVIZ

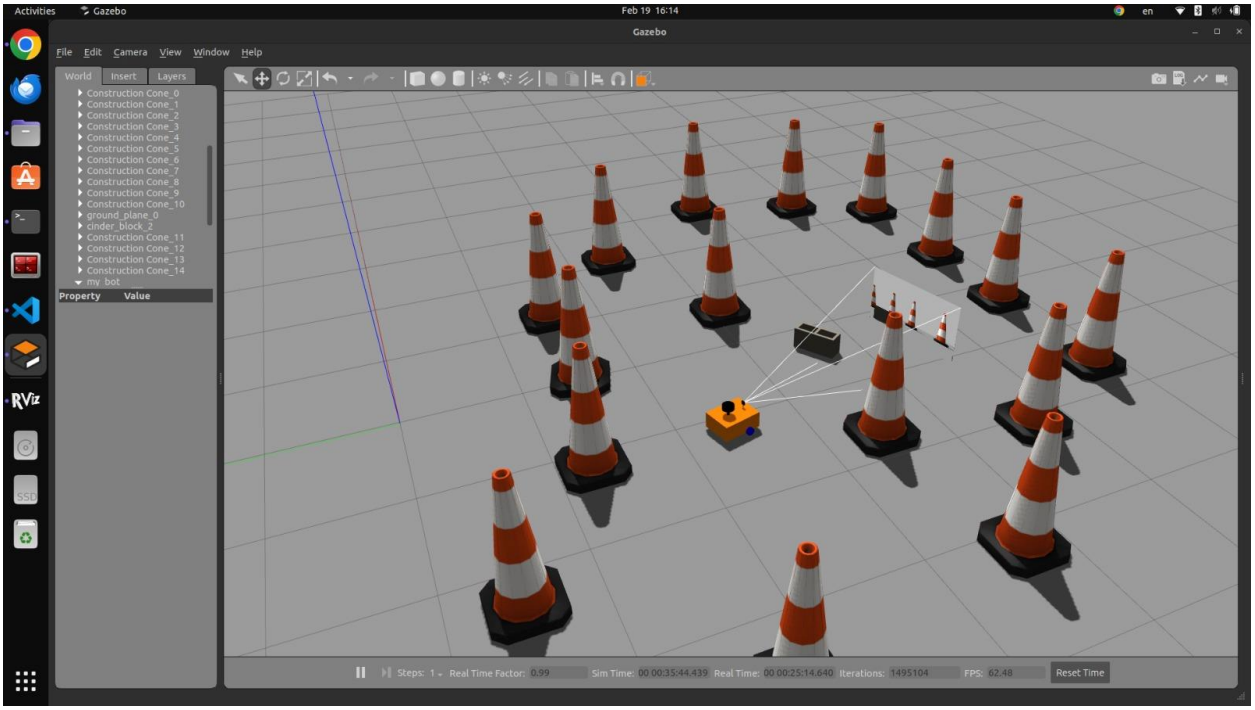


Figure 38. camera view in GAZEBO

4.7.2 Lidar:

RPLIDAR A1 is a low cost 360-degree 2D laser scanner (LIDAR) solution. The system can perform 360-degree scan within 12-meter range (6-meter range of A1M8-R4 and the bellowing models). The produced 2D point cloud data can be used in mapping, localization, and object/environment modeling.

RPLIDAR A1's scanning frequency reached 5.5 Hz when sampling 360 points each round. And it can be configured up to 10 Hz maximum. RPLIDAR A1 is basically a laser triangulation measurement system.

RPLIDAR A1 contains a range scanner system and a motor system. After power on each sub-system, RPLIDAR A1starts rotating and scanning clockwise. Users can get range scan data through the communication interface (Serial port/USB). RPLIDAR A1 comes with a speed detection and adaptive system. The system will adjust frequency of laser scanner automatically according to motor speed. And the host system m can get RPLIDAR A1's real speed through communication interface.

RPLIDAR is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by SLAMTEC. The system measures distance data in more than 2000 times per second and with high resolution distance output (<1% of the distance). RPLIDAR emits modulated infrared laser signal, and the laser signal is then reflected by the object to be detected.

The returning signal is sampled by vision acquisition system in RPLIDAR A1 and the DSP embedded in RPLIDAR A1 starts processing the sample data and output distance value and angle value between object and RPLIDAR A1 through communication interface. The high-speed ranging scanner system is mounted on a spinning rotator with a built-in angular encoding system.

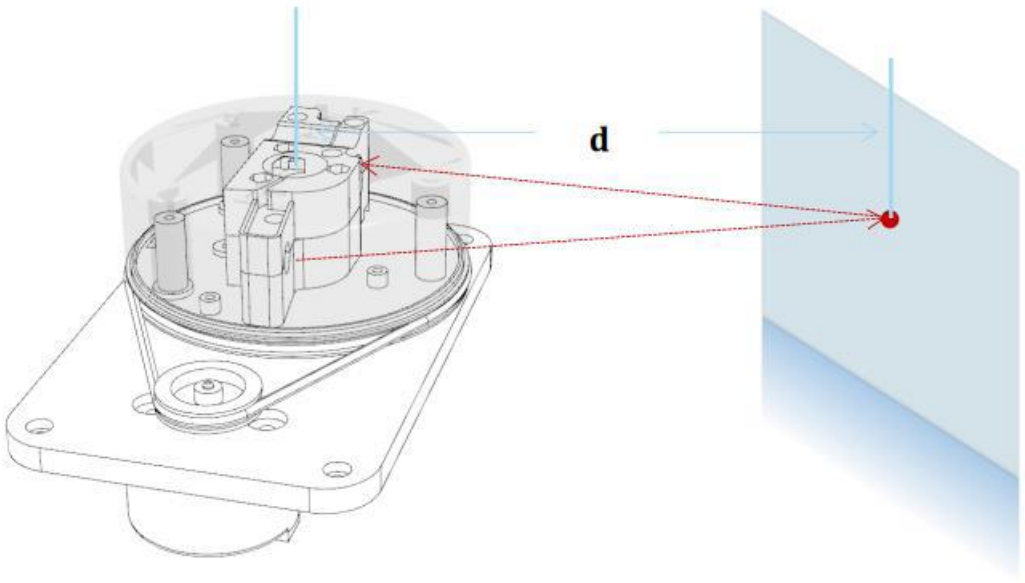


Figure 56 Lidar mechanism

4.7.3 Lidar in simulation:

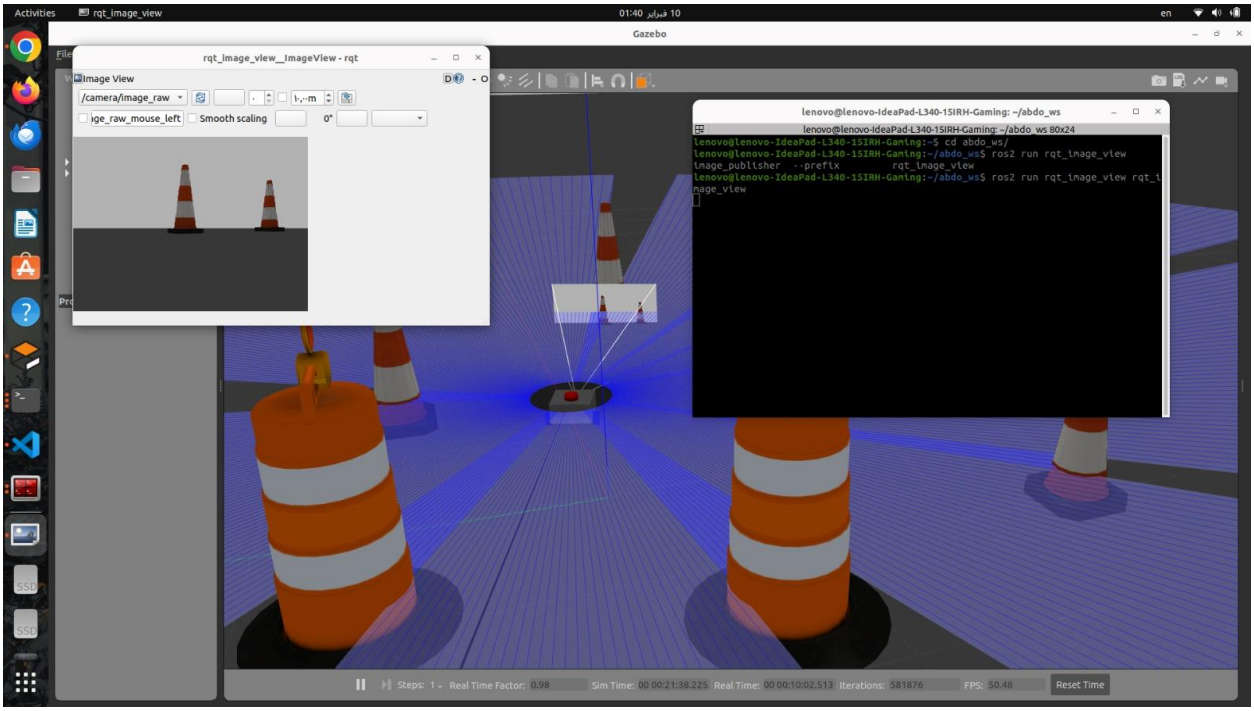


Figure 39. lidar view in GAZEBO

Links:

The URDF shows different link origins, and an overview on the transformations between the different links.

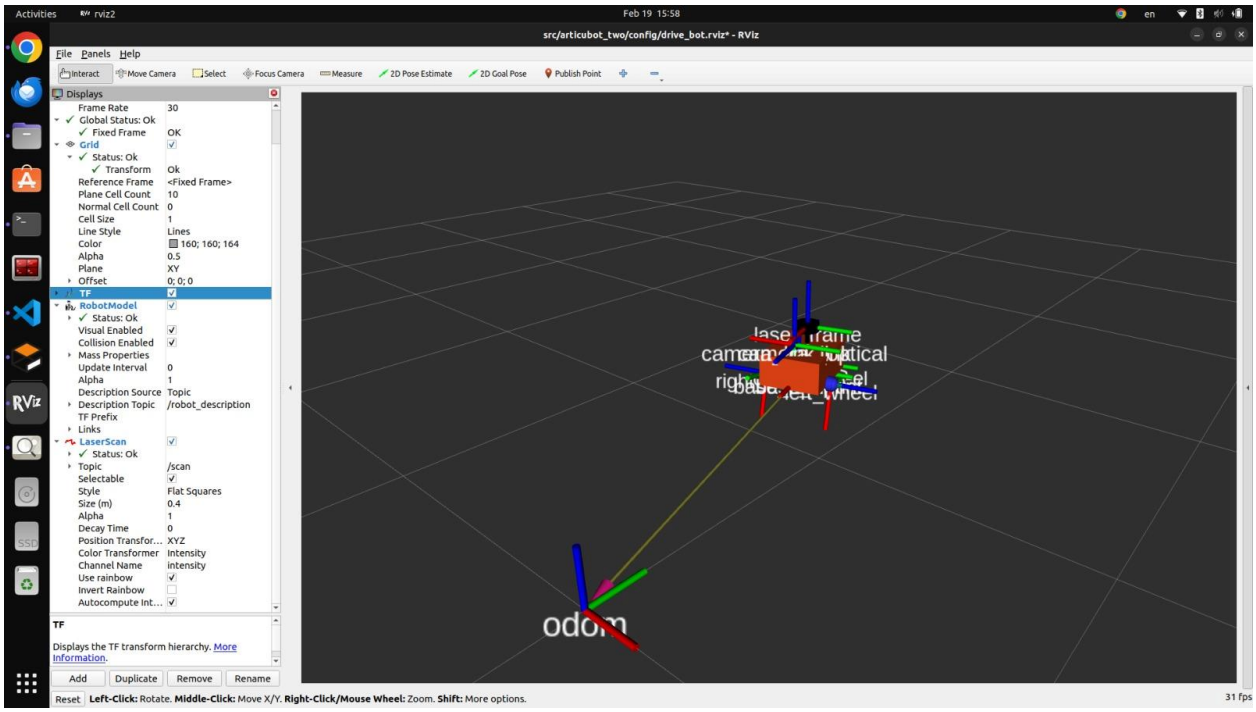


Figure 40. URDF links

4.8 Rqt graph:

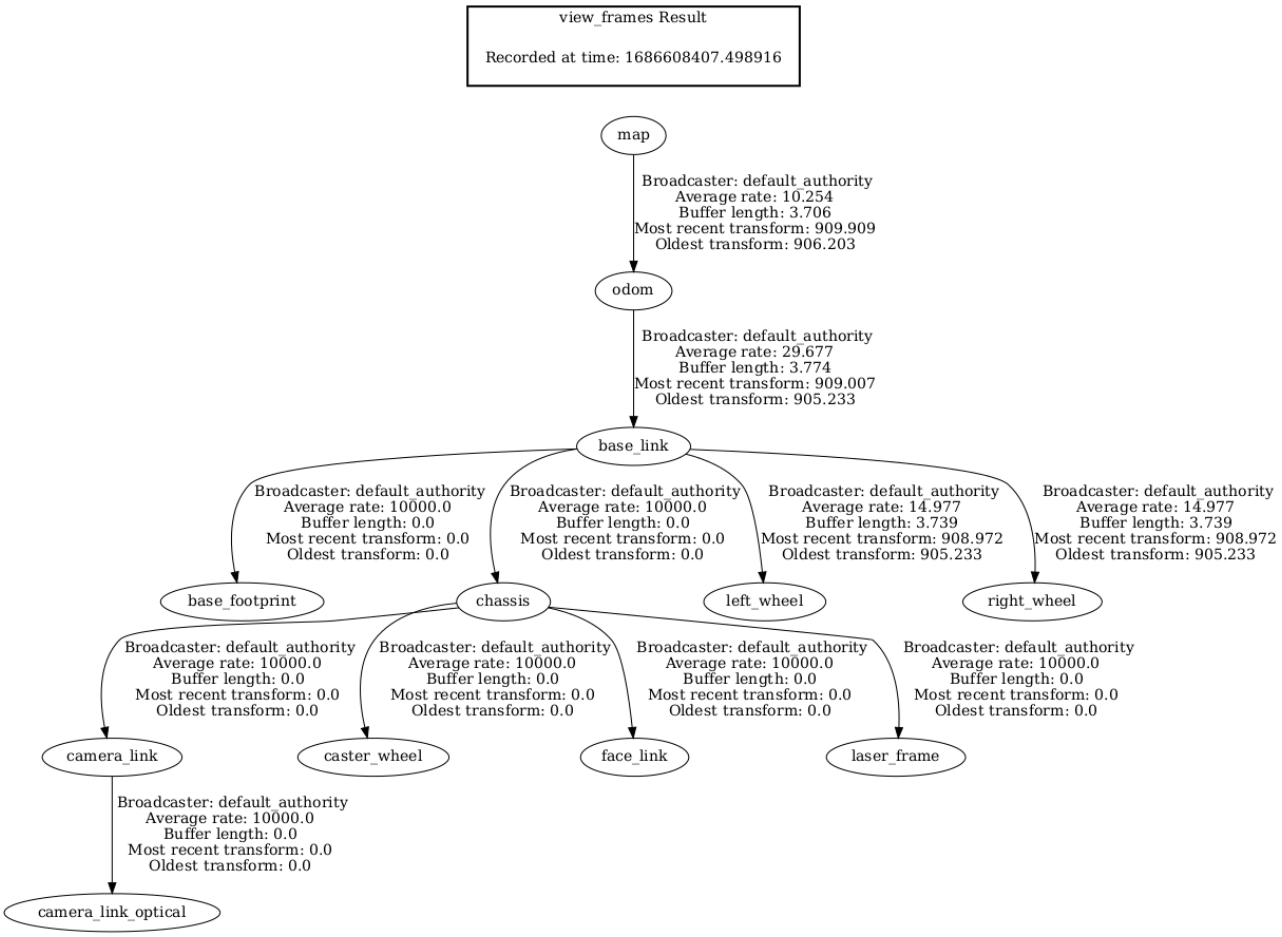


Figure 57 RQT graph

4.9 SLAM and Navigation:

Simultaneous Localization and Mapping (SLAM) is a pivotal technology in robotics and autonomous systems, enabling a robot or vehicle to map an unknown environment while keeping track of its own location within that environment. This dual capability is crucial for navigation tasks where pre-existing maps are not available, or the environment is dynamic and constantly changing. SLAM algorithms process data from various sensors, such as laser scanners, cameras, ultrasonic sensors, and inertial measurement units (IMUs), to build a comprehensive and accurate representation of the surroundings.

The core challenge of SLAM lies in addressing uncertainties and inconsistencies in sensor data due to noise, environmental changes, and movement. Advanced SLAM techniques, such as Graph SLAM and Particle Filter SLAM, employ sophisticated mathematical models and optimization methods to correct these errors and improve the accuracy of the generated maps. Loop closure detection is a critical component, allowing the system to recognize previously visited locations and adjust the map and the robot's position accordingly, thus preventing drift over time.

SLAM has a wide array of applications beyond traditional robotics. In autonomous vehicles, SLAM is integral for real-time navigation and obstacle avoidance. Drones use SLAM for tasks like surveying, inspection, and delivery, where GPS may be unreliable or unavailable. In indoor environments, SLAM enables service robots to perform tasks like cleaning, security patrolling, and warehouse automation with high precision.

Moreover, SLAM technology is being increasingly utilized in augmented reality (AR) and virtual reality (VR) to provide immersive and interactive experiences by mapping and understanding real-world environments. The continual advancements in SLAM are driving the evolution of intelligent systems capable of operating autonomously in complex and unstructured environments, marking significant progress in fields such as logistics, healthcare, exploration, and beyond.

SLAM Toolbox is a robust set of tools and algorithms designed to provide Simultaneous Localization and Mapping (SLAM) capabilities for robots using the Robot Operating System (ROS). It is highly regarded for its flexibility, reliability, and performance, especially in indoor environments. Here are some key features and aspects of SLAM Toolbox:

Key Features of SLAM Toolbox

1. **2D SLAM**:
    - Primarily focused on 2D SLAM, making it suitable for robots operating in planar environments such as indoor spaces.
2. **Multiple Modes**:
    - **Synchronous Mode**: Designed for robots with powerful onboard computation resources, processing data in real-time.
    - **Asynchronous Mode**: Useful for less powerful robots or scenarios where data can be processed after being collected, allowing for batch processing.
3. **Map Serialization**:
    - Supports saving and loading maps, enabling robots to resume operations with pre-existing maps and reducing the need for re-mapping the same areas.
4. **Loop Closure**:
    - Integrates loop closure detection and correction, ensuring that the map remains consistent and accurate over time by recognizing and adjusting for previously visited locations.

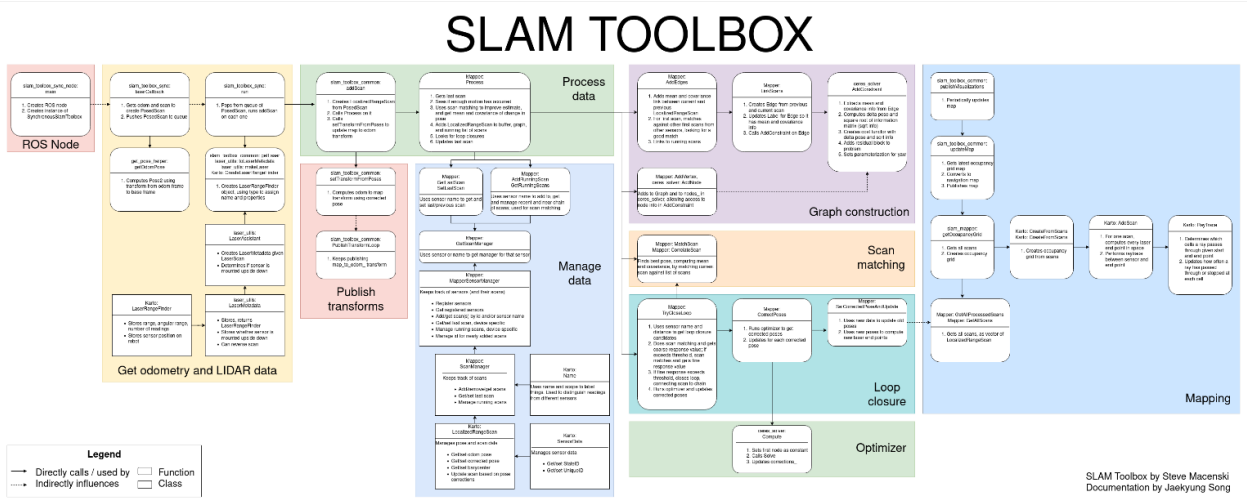


Figure 58 SLAM TOOLBOX

SLAM flowchart:

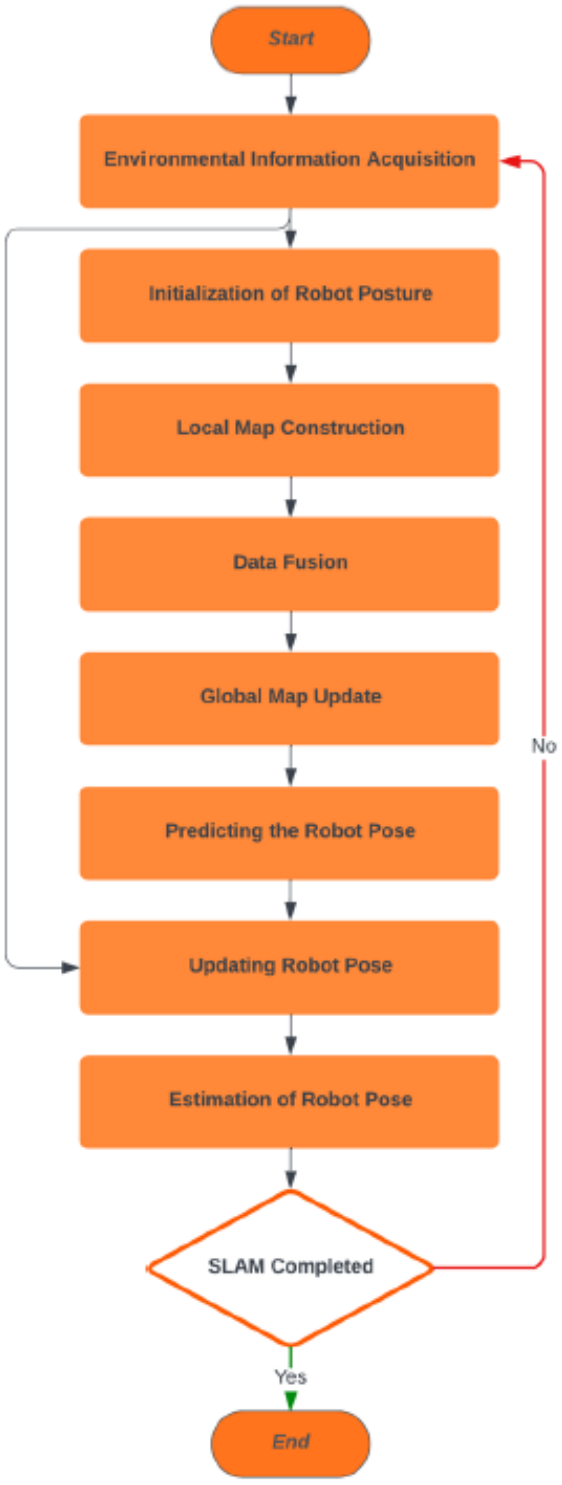


Figure 59 SLAM flowchart

SLAM:

SLAM (simultaneous localization and mapping) is a method used for autonomous vehicles that lets you build a map and localize your vehicle in that map at the same time. SLAM algorithms allow the vehicle to map out unknown environments. Engineers use the map information to carry out tasks such as path planning and obstacle avoidance.

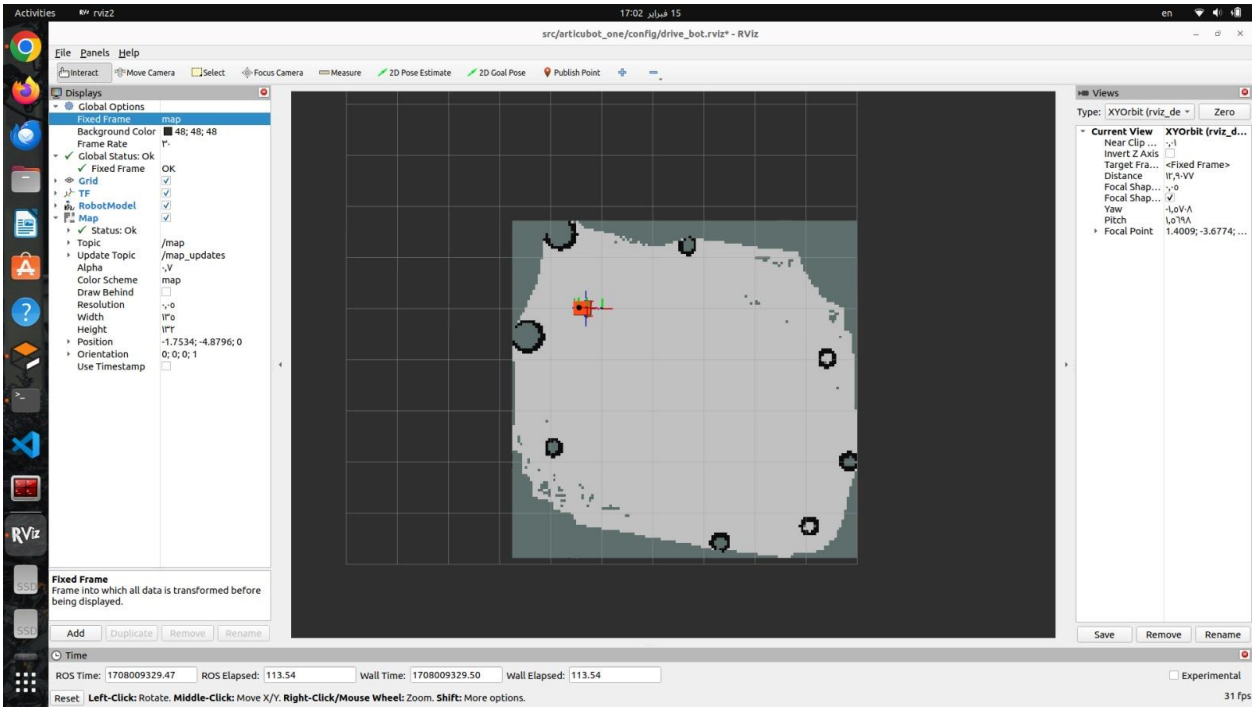


Figure 41. SLAM on RVIZ

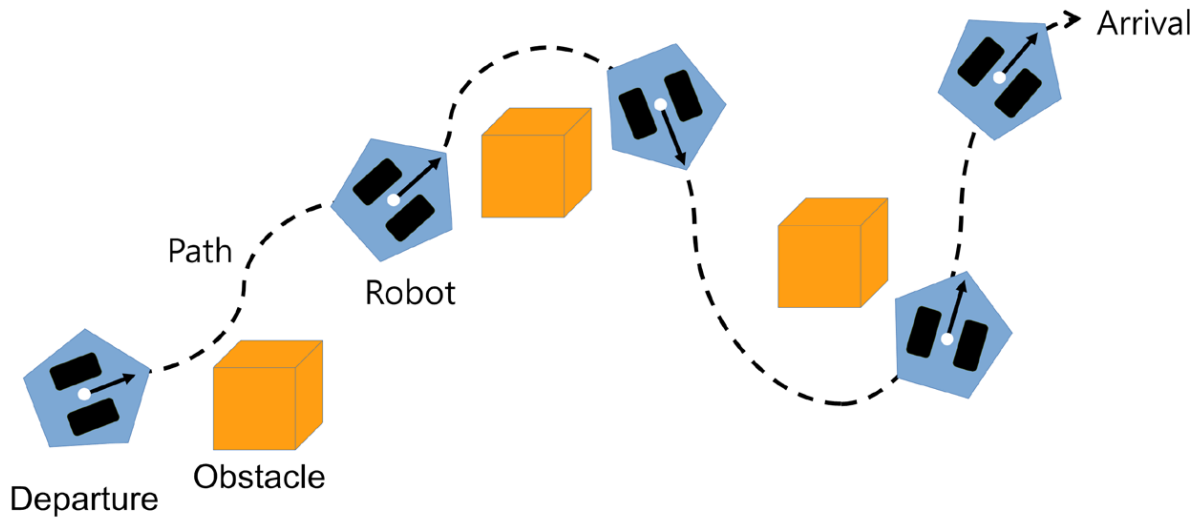
4.10 Navigation

It might be easier to understand navigation as the GPS navigation in a daily life. If you set a destination on the navigation device, the navigation allows you to check the distance and travel time from current location to the destination, and you can set specific preferences and information such as places to stop by and preferred roads on the way.

The navigation system has a relatively short history. In 1981, a Japanese car maker Honda first proposed an analog system based on a three-axis gyroscope and a film map called 'Electro Gryrocator1'. Afterwards Etak Navigator2, an electronic navigation system operated with electronic compass and sensors attached to wheels, was introduced by the U.S. automotive supply company Etak. However, mounting the sensor and electronic compass to the car was a heavy burden for the automobile prices and had reliability problems of the navigation system. Since the 1970s, the United States has been developing satellite positioning systems for military purposes, and in the 2000s, 24 GPS3 (Global Positioning System) satellites became available for general purpose, and triangulation-based navigation systems using these satellites began to spread.

Navigation of Mobile Robot

Let's return to robots now. The foundation and highlight of a mobile robot are without a doubt, navigation. Navigation in robotics is inseparable and essential. Navigation is the movement of the robot to a defined destination, which is not as easy as it sounds. But it is important to know where



the robot itself is and to have a map of the given environment. It is also important to find the optimized route among the various routing options, and to avoid obstacles such as walls and furniture. There's not an easy mission.

What do we need to implement navigation in robots? It may vary depending on the navigation algorithm, and the followings may be required as basic features.

1. Map
2. Pose of Robot
3. Sensing
4. Path Calculation and Driving

4.10.1 Map

The first essential feature for navigation is the map. The navigation system is equipped with a very accurate map from the time of purchase, and the modified map can be downloaded periodically so that it can be guided to the destination based on the map. But will a map of the room be available where the service robot will be placed? Like a navigation system, a robot needs a map, so we need to create a map and give it to the robot, or the robot should be able to create a map by itself.

SLAM4 (Simultaneous Localization And Mapping) is developed to let the robot create a map with or without a help of a human being. This is a method of creating a map while the robot explores the unknown space and detects its surroundings and estimates its current location as well as creating a map.

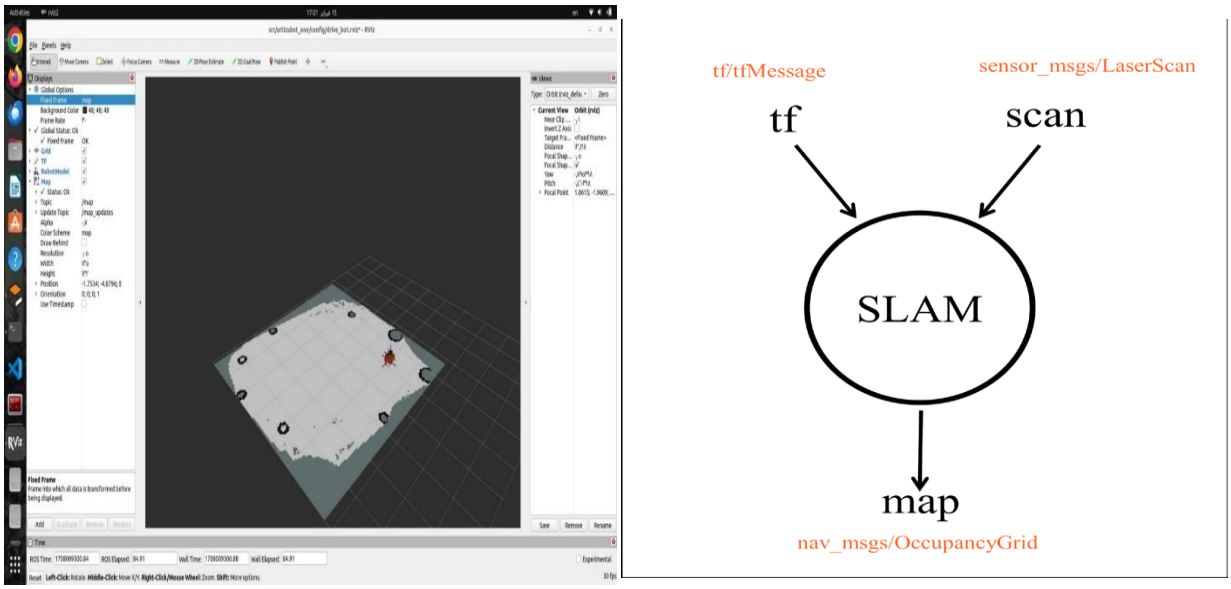


Figure 60 Map

4.10.2 Pose of Robot

Secondly, the robot must be able to measure and estimate its pose (position + orientation). In case of an automobile, the GPS is used to estimate its pose. However, the GPS cannot be used indoors, and even if it can be used, a GPS with large errors cannot be used for robots. Nowadays, high-precision system such as DGPS5 is used, but this is also useless indoors as well as being too expensive for general purpose. In order to overcome this problem, various methods such as marker recognition and indoor location estimation have been introduced. However, in terms of cost and accuracy, it is still insufficient for general use. Currently, the most widely used indoor pose estimation method for service robots is dead reckoning6 7, which is a relative pose estimation, but it has been used for a long time and is composed of low-cost sensors and can obtain a certain level of accurate pose estimation result. The amount of movement of the robot is measured with the rotation of the wheel. However, there is an error between the calculated distance with wheel rotation and the actual travel distance. Therefore, the inertial information

4.10.3 Sensing

Thirdly, figuring out whether there are obstacles such as walls and objects requires sensors. Various types of sensors such as distance sensors and vision sensors are used. The distance sensor uses laser-based distance sensors (LDS, LRF, LiDAR), ultrasonic sensors and infrared. distance sensors. The vision sensor includes stereo cameras, minicameras, omnidirectional. cameras, and recently, RealSense, Kinect, Xtion, which are widely used as Depth camera, are. used to identify obstacles.

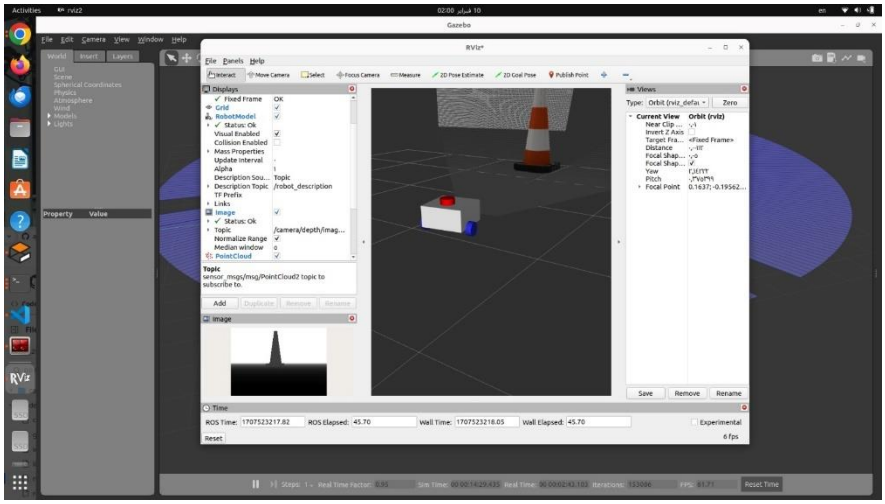


Figure 61 sensing

4.10.4 Path Calculation and Driving

The last essential feature for navigation is to calculate and travel the optimal route to the destination. This is called path search and planning, and there are many algorithms that perform this such as A* algorithm8, potential field9, particle filter10, and RRT (Rapidly-exploring Random Tree) 11.

In this section, we have briefly summarized SLAM and the components of navigation, but it is still difficult and vast to understand. The robot pose measurement and estimation was explained in the previous section. The obstacle measurement such as wall and objects, is described in the Chapter 8 Robots, Sensors and Motors. Now, let's look at the SLAM to create a map, and navigation with the generated map.

After the map being published to visualize how the path planning works we can see the details in the global costmap. The global costmap in Nav2 is responsible for representing the occupancy of the environment and the associated costs for path planning at a global scale.

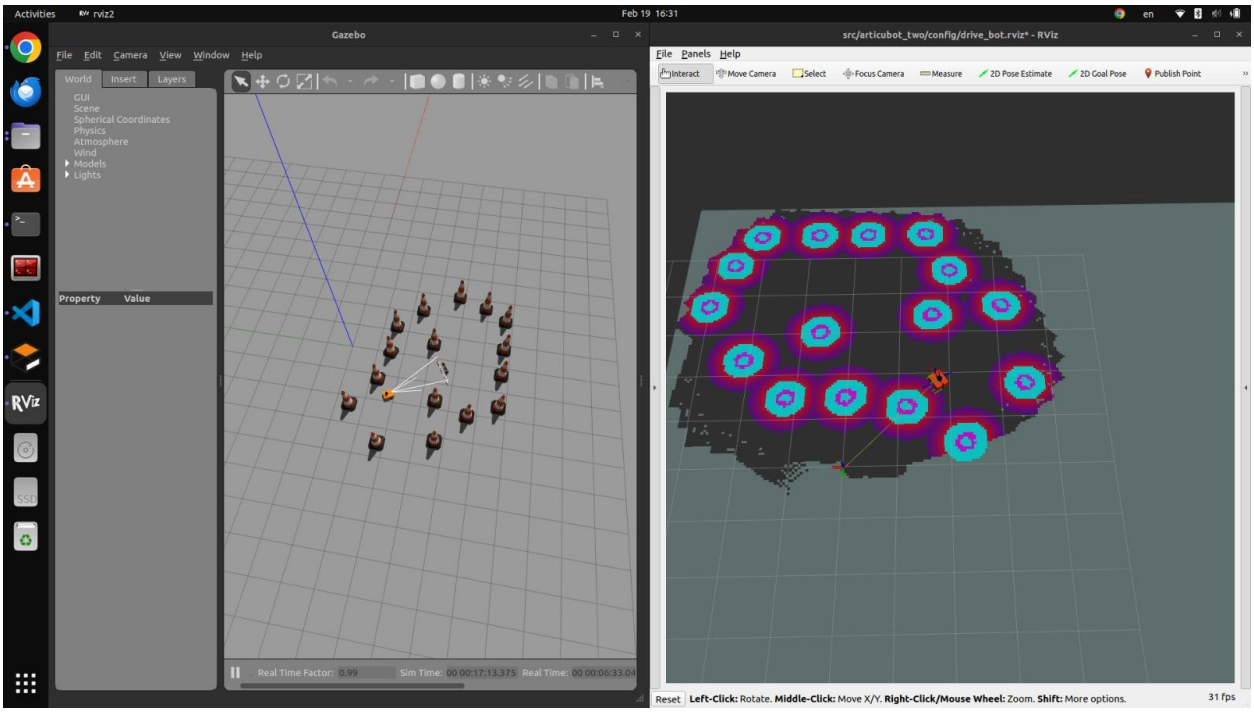


Figure 42. Global costmap after the map being created

4.11 Navigation Theory:

Costmap:

The pose of the robot is estimated based on the odometry obtained from the encoder and inertial sensor (IMU sensor). And the distance between the robot and the obstacle is obtained by the distance sensor mounted on the robot. The pose of robot and sensor, obstacle information, and the occupancy grid map obtained as a result from SLAM are used to load the static map and utilize the occupied area, free area, and unknown area for the navigation.

In navigation, costmap calculates obstacle area, possible collision area, and a robot movable area based on the aforementioned four factors. Depending on the type of navigation, costmap can be divided into two. One is the ‘global_costmap’, which sets up a path plan for navigating in the global area of the fixed map. The other is ‘local_costmap’ which is used for path planning and obstacle avoidance in the limited area around the robot. Although their purposes are different, both costmaps are represented in the same way.

The costmap is expressed as a value between ‘0’ and ‘255’. The meaning of the value is shown in Figure 11-19, and to briefly summarize, the value is used to identify whether the robot is movable or colliding with an obstacle.

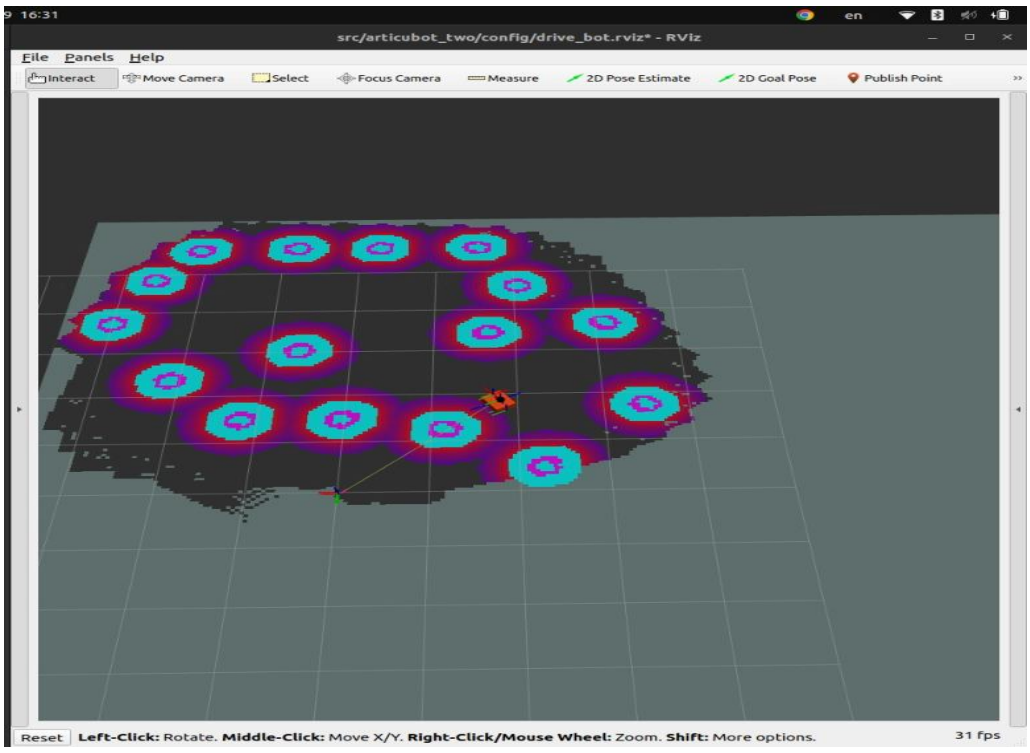


Figure 62 costmap

In the path planning process, you use the 2D goal Pose in Rviz to choose the desired position and orientation. The robot then starts moving to the desired pose taking in consideration its global cost map and considering the local planner in the case that an obstacle not in the map initially showed up.

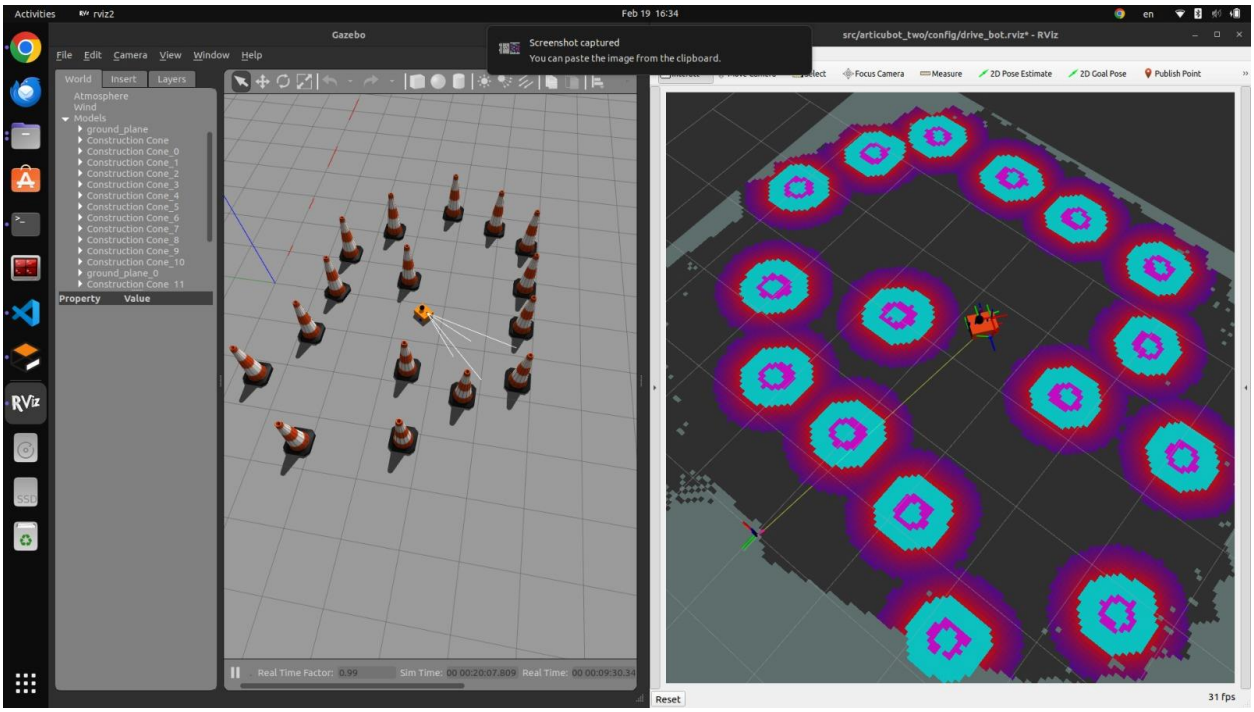


Figure 43. Robot moving to the desired pose

4.12 Conclusion

- In this chapter, we have successfully utilized ROS Foxy to create and simulate a URDF robot equipped with a LiDAR sensor for mapping and navigation purposes. We began by defining the robot's structure using the Unified Robot Description Format (URDF), ensuring that each component was accurately represented and configured. The integration of the LiDAR sensor was a critical step, enabling our robot to perceive its environment and gather essential spatial data.

- Through the implementation of SLAM (Simultaneous Localization and Mapping) algorithms, we transformed the raw LiDAR data into a coherent map of the robot's surroundings. This map served as the foundation for the navigation system, allowing the robot to plan and execute paths within its environment autonomously.

- The successful execution of these tasks demonstrates the powerful capabilities of ROS Foxy in robotics development, particularly in the realms of perception, mapping, and autonomous navigation. By leveraging these tools, we have created a robust platform that can be further expanded and adapted to various applications, showcasing the flexibility and extensibility of ROS in modern robotics projects.

- Overall, this chapter highlights the importance of integrating various ROS packages and tools to achieve complex robotic behaviors and lays the groundwork for future advancements and explorations in robotic autonomy.

5. Chapter V: hardware and testing:

5.1 Hardware result:

5.1.1 Mechanical design:

After we designed our CAD model, and checked the stress analysis, we moved on to manufacturing.

Figure 63 CAD model

Then we assembled the body.

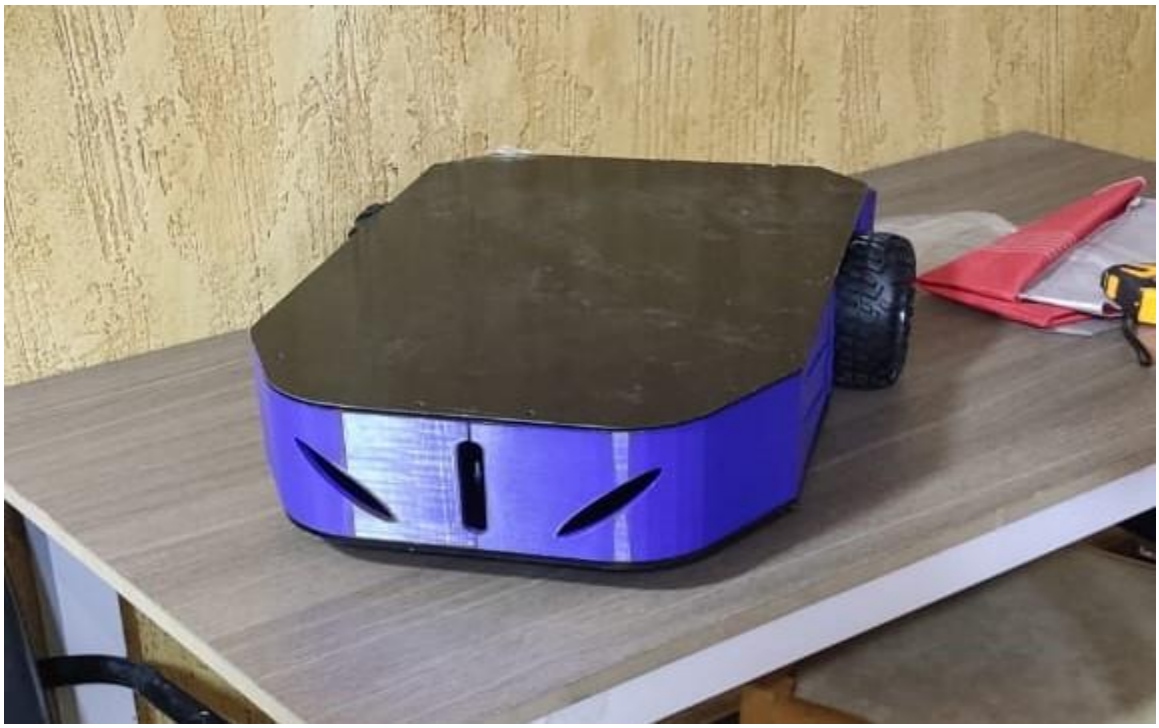


Figure 64 Implemented project_1

Figure 65 Implemented project 2

Electrical circuit:

After we designed our PCB circuit, we moved on to manufacturing.

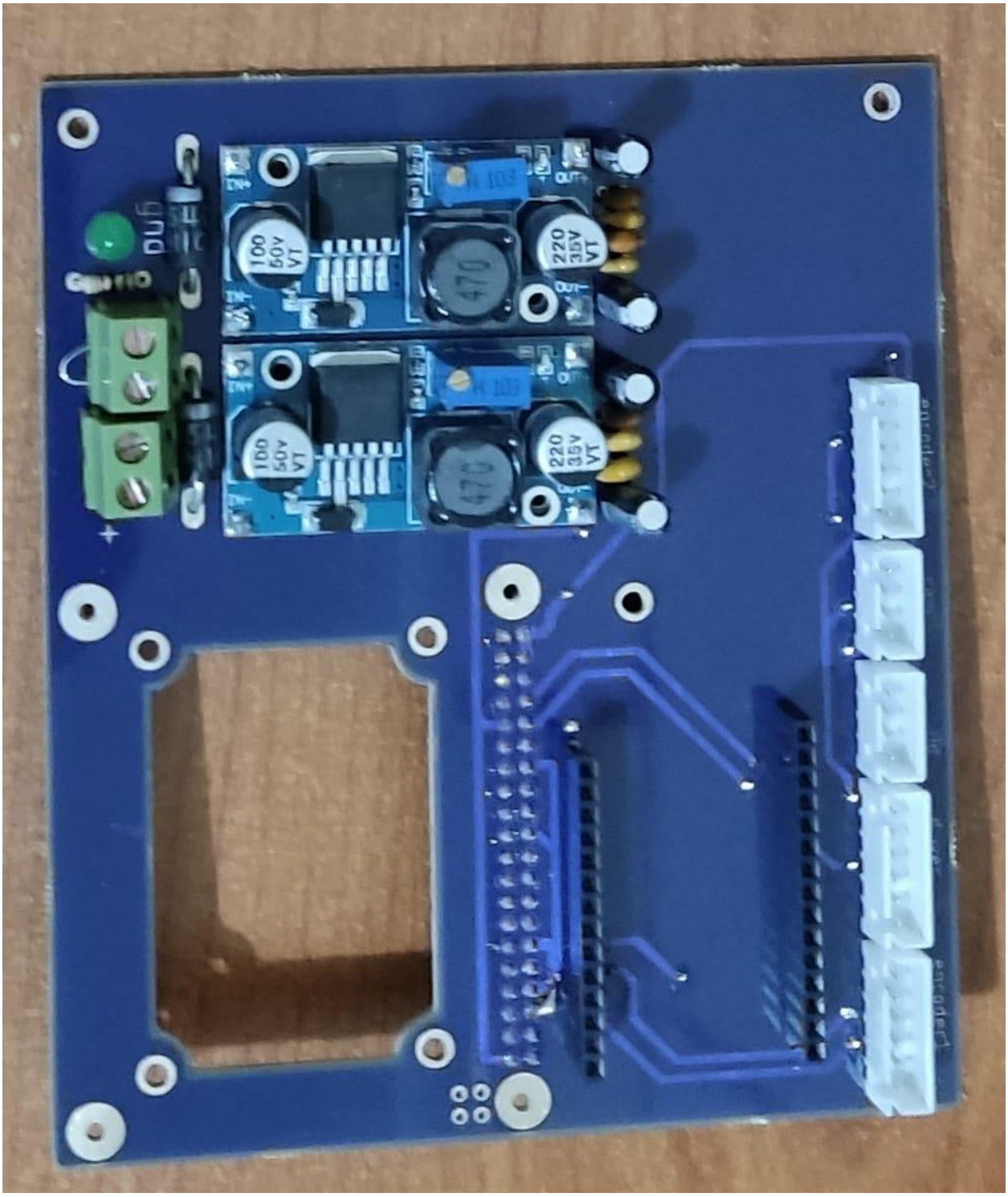


Figure 66 PCB circuit

After we finish from the battery calculation, we moved on to manufacturing.

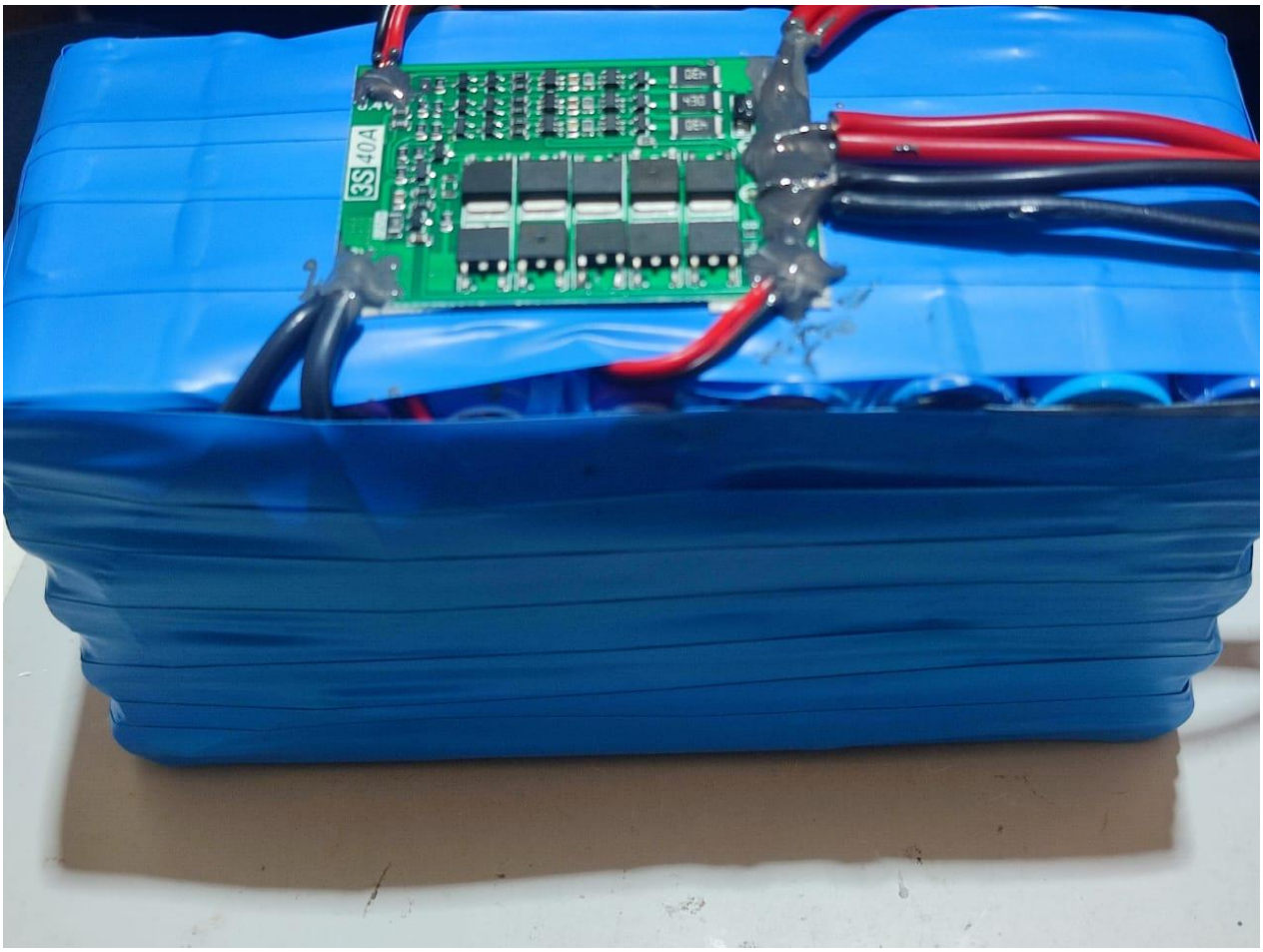


Figure 67 Battery

5.1.2 Electrical design:

After we designed our circuit on eagle program, we moved on to manufacturing.

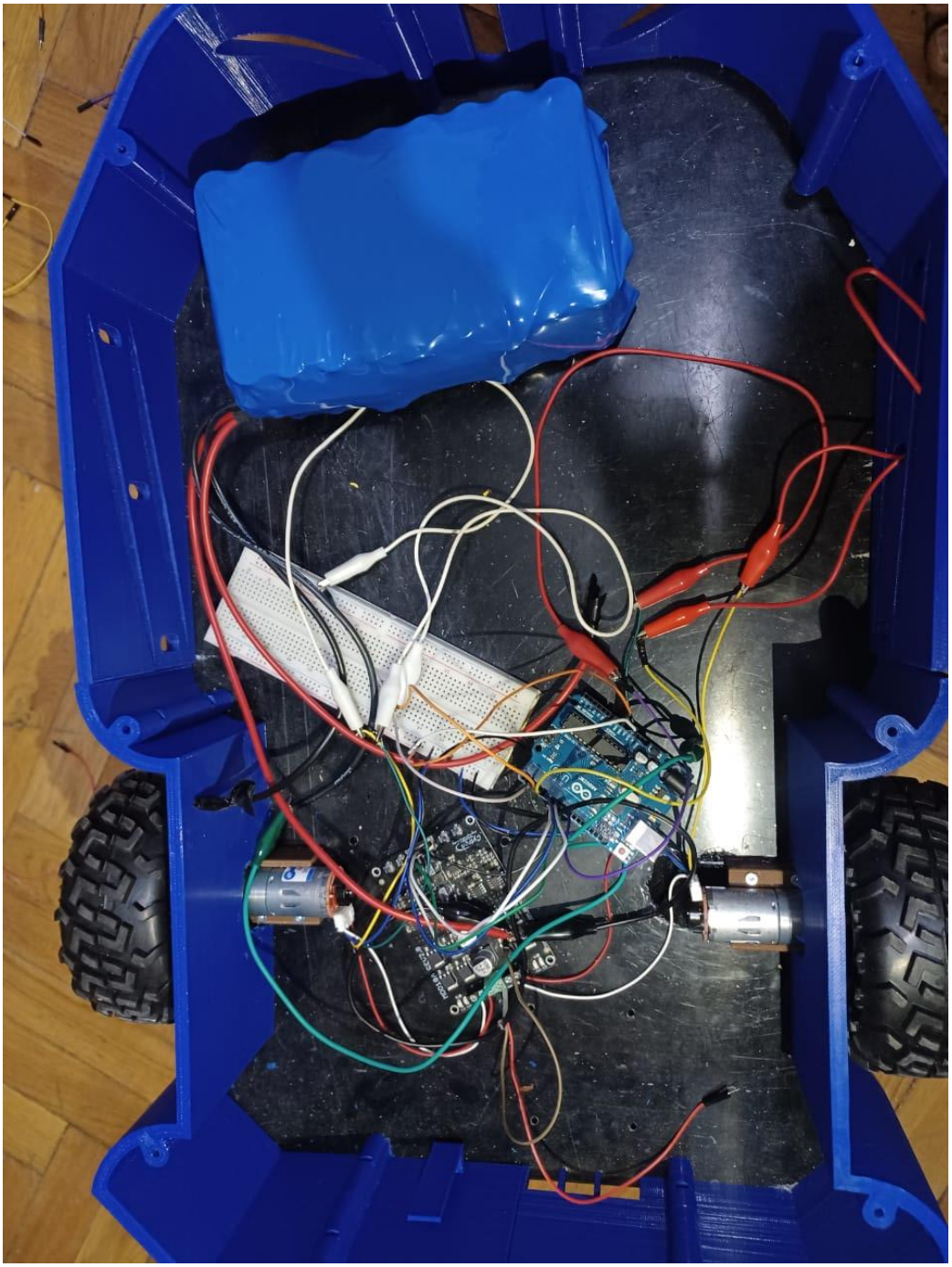


Figure 68 Implemented project 3

5.2 Testing and evaluation of the project:

the robot start moves on.

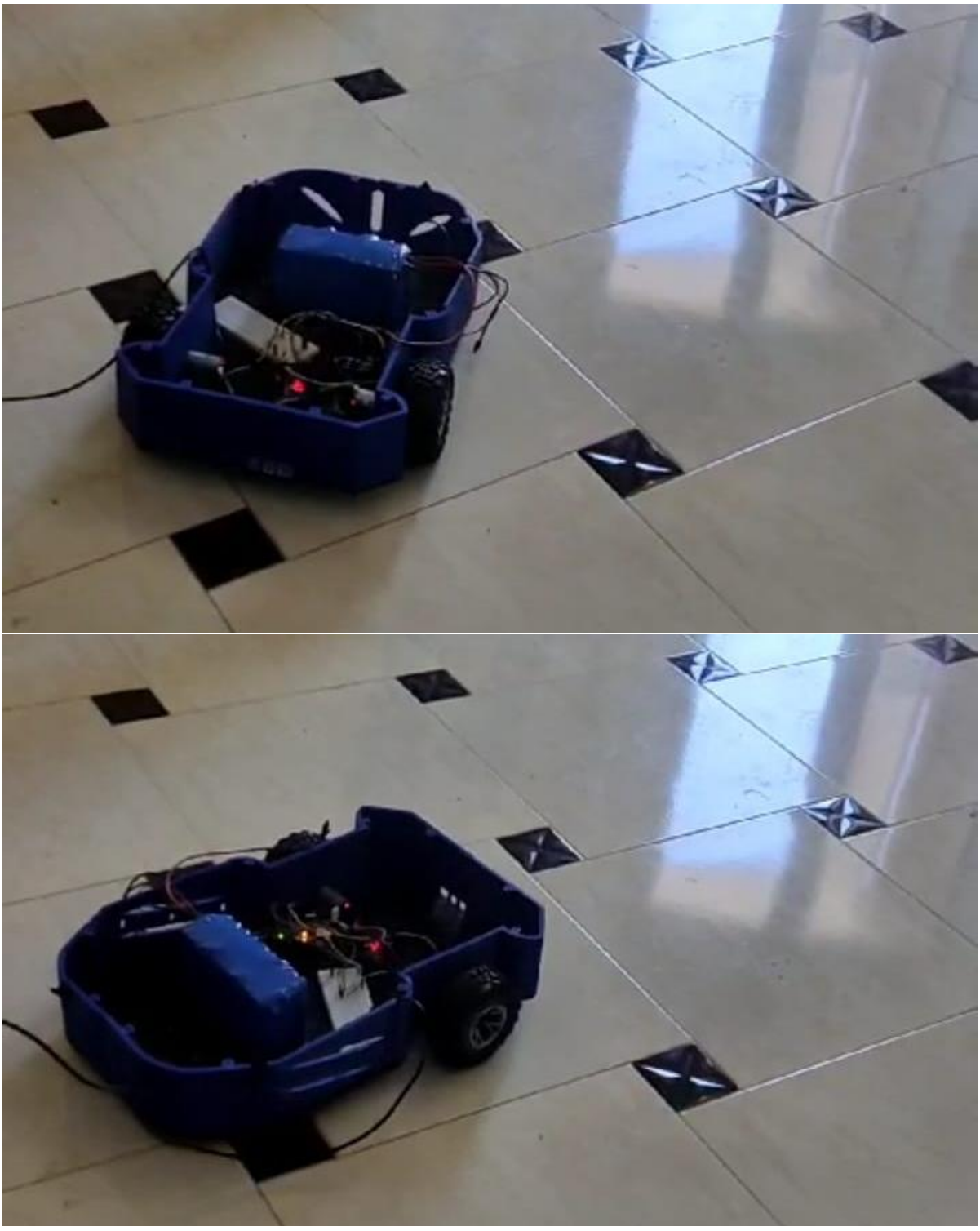


Figure 69 Evaluation of the project

5.3 Conclusion

- In this chapter, we thoroughly tested the integration and functionality of the URDF robot equipped with a LiDAR sensor, focusing on mapping and navigation within ROS Foxy. Through a series of structured tests, we validated the performance and reliability of the robot in various scenarios.

- The testing phase began with verifying the URDF model's accuracy, ensuring that all physical dimensions and joint configurations matched the intended design. We then proceeded to evaluate the LiDAR sensor's data acquisition capabilities, confirming that it could accurately capture and reflect the environment's spatial characteristics.

- Subsequent tests involved implementing SLAM algorithms to create detailed maps of the robot's surroundings. We assessed the quality and accuracy of these maps under different environmental conditions, making necessary adjustments to optimize performance. The navigation tests were equally rigorous, involving the robot's ability to plan and execute paths within the mapped environment. We tested the navigation system's responsiveness and accuracy, ensuring the robot could avoid obstacles and reach its designated targets effectively.

- The successful completion of these tests confirms the robustness of our system. The URDF model, LiDAR sensor, SLAM mapping, and navigation algorithms have been integrated seamlessly, resulting in a reliable and functional autonomous robotic platform. This comprehensive testing process not only validates our current implementation but also provides a solid foundation for future enhancements and applications.

- In conclusion, this chapter demonstrates the critical role of systematic testing in developing a reliable and efficient robotic system. By rigorously validating each component and their interactions, we ensure that our robot can perform autonomously and accurately in real-world scenarios, showcasing the potential of ROS Foxy in advancing autonomous robotic technologies.

References

1. Goyal, S. (2020b). Survey on self driving vehicle. Ccpmohali. https://www.academia.edu/78729673/Survey_on_Self_Driving_Vehicle
2. Crenganis, M., Biris, C., & Girjob, C. (2021). Mechatronic Design of a Four-Wheel drive mobile robot and differential steering. MATEC Web of Conferences, 343, 08003. https://doi.org/10.1051/matecconf/202134308003
3. Autonomous Driving: Pros & Cons | SWARCO. (n.d.). SWARCO. https://www.swarco.com/mobility-future/autonomous-driving/autonomous-driving-pros-cons
4. Self-Driving Cars Survey (2022) – Policygenius. (n.d.). Retrieved December 26, 2023, from https://www.policygenius.com/auto-insurance/self-driving-cars-survey-2022/
5. What is an Autonomous Car? – How Self-Driving Cars Work | Synopsys. (n.d.). Retrieved December 26, 2023, from https://www.synopsys.com/automotive/what-is-autonomous-car.html
6. Development History and Future of Self Driving Autonomous Cars. (n.d.). Retrieved December 26, 2023, from https://techinspection.net/development-history-and-future-of-self-driving-autonomous-cars/
7. Aria, M. (2019). A Survey of Self-driving Urban Vehicles Development. IOP Conference Series: Materials Science and Engineering, 662, 042006. https://doi.org/10.1088/1757-899X/662/4/042006
8. The Advantages of Acrylic Plastic—Acme Plastics. (n.d.). Retrieved December 26, 2023, from https://www.acmeplastics.com/content/advantages-acrylic/
9. Properties & Applications of Acrylic | Carville Plastics. (n.d.). CarvillePlastics. Retrieved December 26, 2023, from https://www.carvilleplastics.com/latest_news/key-properties-acrylic/
10. Acrylic Plastic | Compare Plastics & View Acrylic Material Properties | Curbell Plastics. (n.d.). Retrieved December 26, 2023, from https://www.curbellplastics.com/materials/plastics/acrylic/

11. *A textbook of machine design*. (n.d.). Google Books. https://books.google.co.in/books?id=6FZ9UvDgBoMC&printsec=frontcover#v=onepage&q&f=false
12. *DC Gear Motor with Magnetic Linear Encoder 25GA370 (7.8KG-210RPM-12V) – Free Electronics*. (n.d.). https://free-electronic.com/product/dc-gear-motor-with-magnetic-linear-encoder-25ga370-7-8kg-250rpm-12v/
13. Wilfried Voss (2007) Guide_to_Servo_Motor_Sizing. https://cnctar.hobbycnc.hu/VarsanyiPeter/CNC%20vezerles%20-%20szervoval/Guide_to_Servo_Motor_Sizing.pdf
14. Czernia, D. (2024, January 18). *Mass moment of inertia calculator*. https://www.omnicalculator.com/physics/mass-moment-of-inertia
15. A. Dommenech, T. Domenech and J. Cerbiran, "Introduction to The Study of Rolling Friction", American Association of Physics Teachers, Am. J. Phys. 55(3), March 1987.
16. Rony Argueta, "The Electric Vehicle," Technical Report, Santa Barbara College of Engineering, University of California, Mar. 2010
17. Dr N J S Gorst, Dr S J Williamson, Eur Ing P F Pallett and Professor L A Clark, "Friction in Temporary Works", Technical Report 071, School of Engineering, University of Birmingham, Birmingham, 2003.
18. (Difference Between Self Driving Cars and Regular Cars | Difference Between, n.d.)
19. A1M8 Datasheet by Seeed Technology Co., Ltd | Digi-Key Electronics. (n.d.). Retrieved December 26, 2023, from https://www.digikey.dk/htmldatasheets/production/3265529/0/0/1/a1m8.html
20. All you need to know about Slamtec RPLIDAR, MAPPER and Slamware—Latest Open Tech From Seeed. (n.d.). Retrieved December 26, 2023, from https://www.seeedstudio.com/blog/2019/08/05/all-you-need-to-know-about-slamtec-rplidar-mapper-and-slamware/

21. kexugit. (2015, August 10). Kinect—3D Sight with Kinect. https://learn.microsoft.com/en-us/archive/msdn-magazine/2012/november/kinect-3d-sight-with-kinect
22. Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors—PMC. (n.d.). Retrieved December 26, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4701245/
23. 10Amp 5V-30V DC Motor Driver (2 Channels). (n.d.). Retrieved December 26, 2023, from https://www.cytron.io/p-10amp-5v-30v-dc-motor-driver-2-channels
24. Mass Moment of Inertia Calculator. (n.d.). Retrieved December 26, 2023, from https://www.omnicalculator.com/physics/mass-moment-of-inertia
25. ESP32 VS ESP8266- Detailed Comparison Guide in 2023. (n.d.). Retrieved December 26, 2023, from https://www.etechnophiles.com/esp32-vs-esp8266/
26. Lithium-ion battery—Wikipedia. (n.d.). Retrieved December 26, 2023, from https://en.wikipedia.org/wiki/Lithium-ion_battery
27. *A comparative study of electrochemical battery for electric vehicles applications*. (2019, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9071955
28. Abdelhalim. (n.d.). *Battery Technology Comparison IEEE 2020*. Scribd. https://www.scribd.com/document/547699183/Battery-Technology-Comparison-IEEE-2020
29. Crenganis, M., Biris, C., & Girjob, C. (2021). Mechatronic Design of a Four-Wheel drive mobile robot and differential steering. MATEC Web of Conferences, 343, 08003. https://doi.org/10.1051/matecconf/202134308003
30. How to calculate battery capacity for my robot system—Robotics Stack Exchange. (n.d.). Retrieved December 26, 2023, from

https://robotics.stackexchange.com/questions/19661/how-to-calculate-battery-capacity-for-my-robot-system.

31. High Quality Rechargeable 14.8v 4400mah 5200mah 6600mah Lithium Battery Pack For Emergency Power Supply—Buy 14.8v Li Ion Battery Pack 18650 Battery Pack 18650 Li-ion Batteries With Bms For Vacuum Cleaner Sweepwer,Oem 14.4v Battery Pack 2600mah 2200mah 2000mah 2500mah 3500mah,Customized Wholesale 3.7v 7.4v 12v 14.8v 2000mah 6000mah 12000mah 18650 26650 21700 Lithium Battery Pack14.8v Li Ion Batteries Product on Alibaba.com. (n.d.). Retrieved December 26, 2023, from https://www.alibaba.com/product-detail/High-Quality-Rechargeable-14-8V-4400mah_1600914965184.html?spm=a2700.7735675.0.0.291fPZ4pPZ4paF&s=p.
32. (625) Building a mobile robot—YouTube. (n.d.). Retrieved December 26, 2023, from https://www.youtube.com/playlist?list=PLunhqkrRNRhYAffV8JDiFOatQXuU-NnxT
33. ROS2 For Beginners (ROS Foxy, Humble—2023) | Udemy. (n.d.). Retrieved December 26, 2023, from https://www.udemy.com/course/ros2-for-beginners/?kw=ros&src=sac
34. ROS2 For Beginners (ROS Foxy, Humble—2023) | Udemy. (n.d.). Retrieved December 26, 2023, from https://www.udemy.com/course/ros2-for-beginners/?kw=ros&src=sac
35. ROS2 Nav2 [Navigation 2 Stack]—With SLAM and Navigation | Udemy. (n.d.). Retrieved December 26, 2023, from https://www.udemy.com/course/ros2-nav2-stack/

Appendices

Robot_core.urdf:

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" >

    <xacro:include filename="inertial_macros.xacro"/>


    <xacro:property name="chassis_length" value="0.335"/>
    <xacro:property name="chassis_width" value="0.265"/>
    <xacro:property name="chassis_height" value="0.138"/>
    <xacro:property name="chassis_mass" value="1.0"/>
    <xacro:property name="wheel_radius" value="0.033"/>
    <xacro:property name="wheel_thickness" value="0.026"/>
    <xacro:property name="wheel_mass" value="0.05"/>
    <xacro:property name="wheel_offset_x" value="0.226"/>
    <xacro:property name="wheel_offset_y" value="0.1485"/>
    <xacro:property name="wheel_offset_z" value="0.01"/>
    <xacro:property name="caster_wheel_radius" value="0.01"/>
    <xacro:property name="caster_wheel_mass" value="0.01"/>
    <xacro:property name="caster_wheel_offset_x" value="0.075"/>
    <xacro:property name="caster_wheel_offset_z" value="${wheel_offset_z - wheel_radius + caster_wheel_radius}"/>

```

```
<material name="white">
        <color rgba="1 1 1 1" />
    </material>

    <material name="orange">
        <color rgba="1 0.3 0.1 1"/>
    </material>

    <material name="blue">
        <color rgba="0.2 0.2 1 1"/>
    </material>

    <material name="black">
        <color rgba="0 0 0 1"/>
    </material>

    <material name="red">
        <color rgba="1 0 0 1"/>
    </material>
```

Figure 70 URDF file

```
<!-- BASE LINK -->

    <link name="base_link">

    </link>

    <!-- BASE_FOOTPRINT LINK -->

    <joint name="base_footprint_joint" type="fixed">
        <parent link="base_link"/>
        <child link="base_footprint"/>
        <origin xyz="0 0 0" rpy="0 0 0"/>
    </joint>

    <link name="base_footprint">
    </link>
```

Figure 71 Base Link

```
<!-- CHASSIS LINK -->

    <joint name="chassis_joint" type="fixed">
        <parent link="base_link"/>
        <child link="chassis"/>
        <origin xyz="${-wheel_offset_x} 0 ${-wheel_offset_z}"/>
    </joint>

    <link name="chassis">
        <visual>
            <origin xyz="${chassis_length/2} 0 ${chassis_height/2}"/>
            <geometry>
                <box size="${chassis_length} ${chassis_width} ${chassis_height}"/>
            </geometry>
            <material name="orange"/>
        </visual>
        <collision>
            <origin xyz="${chassis_length/2} 0 ${chassis_height/2}"/>
            <geometry>
                <box size="${chassis_length} ${chassis_width} ${chassis_height}"/>
            </geometry>
        </collision>
        <xacro:inertial_box mass="0.5" x="${chassis_length}" y="${chassis_width}" z="${chassis_height}">
            <origin xyz="${chassis_length/2} 0 ${chassis_height/2}" rpy="0 0 0"/>
        </xacro:inertial_box>
    </link>

    <gazebo reference="chassis">
        <material>Gazebo/Orange</material>
    </gazebo>
```


Figure 72 Chassis Link

```
<!-- RIGHT WHEEL LINK -->

  <joint name="right_wheel_joint" type="continuous">
      <parent link="base_link"/>
      <child link="right_wheel"/>
      <origin xyz="0 ${-wheel_offset_y} 0" rpy="${pi/2} 0 0" />
      <axis xyz="0 0 -1"/>
  </joint>

  <link name="right_wheel">
      <visual>
          <geometry>
              <cylinder radius="${wheel_radius}" length="${wheel_thickness}"/>
          </geometry>
          <material name="blue"/>
      </visual>
      <collision>
          <geometry>
              <sphere radius="${wheel_radius}"/>
          </geometry>
      </collision>
      <xacro:inertial_cylinder mass="${wheel_mass}" length="${wheel_thickness}" radius="${wheel_radius}">
          <origin xyz="0 0 0" rpy="0 0 0"/>
      </xacro:inertial_cylinder>
  </link>

  <gazebo reference="right_wheel">
      <material>Gazebo/Blue</material>
  </gazebo>
```

Figure 73 Right Wheel Link

```
<!-- LEFT WHEEL LINK -->

   <joint name="left_wheel_joint" type="continuous">
       <parent link="base_link"/>
       <child link="left_wheel"/>
       <origin xyz="0 ${wheel_offset_y} 0" rpy="-${pi/2} 0 0" />
       <axis xyz="0 0 1"/>
   </joint>

   <link name="left_wheel">
       <visual>
           <geometry>
               <cylinder radius="${wheel_radius}" length="${wheel_thickness}"/>
           </geometry>
           <material name="blue"/>
       </visual>
       <collision>
           <geometry>
               <sphere radius="${wheel_radius}"/>
           </geometry>
       </collision>
       <xacro:inertial_cylinder mass="${wheel_mass}" length="${wheel_thickness}" radius="${wheel_radius}">
           <origin xyz="0 0 0" rpy="0 0 0"/>
       </xacro:inertial_cylinder>
   </link>

   <gazebo reference="left_wheel">
       <material>Gazebo/Blue</material>
   </gazebo>
```

Figure 74 Left Wheel Link

```
<!-- CASTER WHEEL LINK -->

    <joint name="caster_wheel_joint" type="fixed">
        <parent link="chassis"/>
        <child link="caster_wheel"/>
        <origin xyz="${caster_wheel_offset_x} 0 ${caster_wheel_offset_z}"/>
    </joint>


    <link name="caster_wheel">
        <visual>
            <geometry>
                <sphere radius="${caster_wheel_radius}"/>
            </geometry>
            <material name="white"/>
        </visual>
        <collision>
            <geometry>
                <sphere radius="${caster_wheel_radius}"/>
            </geometry>
        </collision>
        <xacro:inertial_sphere mass="${caster_wheel_mass}" radius="${caster_wheel_radius}">
            <origin xyz="0 0 0" rpy="0 0 0"/>
        </xacro:inertial_sphere>
    </link>

    <gazebo reference="caster_wheel">
        <material>Gazebo/White</material>
        <mu1 value="0.001"/>
        <mu2 value="0.001"/>
    </gazebo>

</robot>
```

Figure 75 Caster Wheel Link