E. N. S. T
المدرسة الوطنية العليا للتكنولوجيا
Ecole Nationale Supérieure de Technologie
The National Higher School of Technology

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry Of Higher Education and Scientific Research
المدرسة الوطنية العليا للتكنولوجيا
Higher National School of Technology
Department Of Industrial Engineering & Maintenance

Department of Industrial Engineering & Maintenance

Final Year Project to Obtain the Diploma of Engineering

-Field-

ELECTROMECHANICS

-Specialty-

MECHATRONICS

-Subject-

Conception and Realization of Automated Guided Vehicle (AGV)

Realized by:

BENZEKRI Yasser

LAKAF Bakir

Members of the jury:

CHABANE Ali (MAB)	Advisor
GHAZI Nawal (MAA)	Co-advisor
AMRANE Ahmed (MCB)	President
ALEM Said (MCA)	Examiner

Algiers, 25 June 2023

Academic year 2022-2023

Acknowledgement

*First and foremost, we extend our deepest gratitude to **Allah**, the Almighty. Without His blessings, guidance, and the strength He graciously bestowed upon us, this work would not have come to fruition. During the challenges and difficulties faced since the inception of this journey, we are forever thankful to Him, and all great praise goes to Allah.*

*Heartfelt thanks to **Our Parents** who have played a significant role in shaping who we are today. Their unwavering faith in our abilities, constant encouragement, and boundless love have been the pillars of strength that have guided us through life's ups and downs.*

*We would like to express our sincere appreciation to our advisors **Mr. CHABANE ALI** and **Ms. GHAZI Nawal** for their invaluable advice, continuous support, and constructive feedback throughout the development of this thesis. Their dedication and expertise have been an inspiration and motivation for our work.*

We are grateful to member of jury for their thorough review and insightful comments that significantly improved the quality of this thesis.

*We are great thankful to **BENZEKRI SALAH**, **BENZEKRI NACER** and **LAKAF ABDELOUAHAB**, whose generous financial support greatly facilitated our research. Their contributions to this project will never be forgotten and we ask Allah to bless them in their wealth and deeds.*

*Last but not least, we are thankful for the companionship, understanding, and support of our friends, especially **SEKOUTI Mounir**, **BENZEKRI Mohamed**, **LAKAF Aboubaker** and **TICHOUCHAI Omar** Their friendship has been a source of happiness and comfort during this challenging journey.*

To all those mentioned above and those who indirectly contributed to this research journey, we extend our deepest gratitude.

BAKIR & YASSER

Dedication

Firstly, I dedicate this work to my beloved **parents SAID** and **ZOHRA** the pillars of my strength. You have shown me the value of perseverance and the joy of learning. Your sacrifices, love, and unending support have made me who I am today. I also dedicate this to my brothers **SALAH**, **Dr. YOUCEF**, **BRAHIM** and **ANES** and **my sisters FAFA and AMEL** They helped me with their morale, encouragement and support all the time. I'm grateful I have brothers like them This achievement is not just mine, but yours too. I dedicate this to you with all my love and gratitude.

A special dedicate to my dear grandmother, **MAMA FATIMA** may God bless his soul, who loved science and the people of science and always encouraged us morally and prayed to us for the highest grades and mattresses. And also, my maternal grandparents for their invitations and their faith in me for the last year of my studies.

to my nieces and nephews, the sons and daughters of my brothers and sisters (**AYMEN, SALSABIL, SANAA**), this work is dedicated to you. May it inspire you to dream, learn, and strive for your goals. I hope that my journey will become a beacon, guiding you through your own paths in life. Remember, no dream is too big, and no effort is ever wasted.

I dedicate this work specifically to **HASSAN**, **MOHAMMED**, **ABDULLAH** and **YASSER** for the support and encouragement throughout my career towards graduation.

to my family **UNCLES** and **AUNTS** and **COUSINS**, this work is for you. Your encouragement has been invaluable. Your love, support, this achievement is not just mine, but yours too. I dedicate this to you with all my love and gratitude.

To my invaluable partner **BAKIR** in this academic journey, your camaraderie and collaboration have been instrumental in reaching this milestone. I dedicate this work to you as a testament of our shared efforts and commitment towards our graduation.
To **MY FRIENDS** for all the good times and laughter we shared, thanks to you my years in university are unforgettable memories.

Yasser

Dedication

In memory of my very dear grandfather, **BABA SLIMAN**, may God bless his soul. He was and will forever remain in my heart. His dream of seeing me reach this point has guided me throughout this study. I dedicate this work to his beloved memory

To my dear **parents**, your strong support and belief in me have been my guide. Your hard work and patience have taught me to be strong and keep going. Your wise words and help have always been there when I needed them. I am very thankful for everything you have done for me and the lessons you've taught me.

To my brother **SAID**, thank you for your companionship, for always standing by my side and encouraging me to strive for more. To my sisters **KAOUTHER** and **DJAMILA**, you have taught me kindness and patience through your actions.

To my nephew **IBRAHIM**, you represent the future generation to whom I hope this work may be beneficial. With all my love, dear **Khaloo**.

To my entire **family**, near and far, a simple yet heartfelt thanks. Your love and support have bolstered me in ways words can't express.

A special thanks to my dear friends **MOUSSEDEK** and **ABDELAZIZ**. Your friendship has been a source of joy and strength.

To **YASSER**, my academic partner, your moral support, patience, and understanding have made this project a reality

Bakir

Abstract

In the era of Industry 4.0, where digital and physical worlds converge, robotics has emerged as a significant player. This project focuses on Automated Guided Vehicles (AGVs), autonomous vehicles central to warehouse operations, manufacturing facilities, or logistic centers. Our mission in this work was to augment the navigation and adaptability of AGVs, which is critical for efficient operations in dynamically changing environments. To this end, we proposed the integration of advanced technologies, namely, the Robot Operating System (ROS), Simultaneous Localization and Mapping (SLAM), and the A* algorithm. The project was executed in two main phases: a study and simulation phase and a realization phase. In the first phase, we sought to implement automatic navigation and map creation functionalities. The second phase was designed to translate our theoretical knowledge from the simulation phase into a functional AGV prototype.
Key-words : Automated Guided Vehicles (AGV), Robot Operating System (ROS) , Simultaneous Localization and Mapping (SLAM) algorithms, A* (A-star) algorithms.

Résumé :

À l'ère de l'Industrie 4.0, où les mondes numérique et physique convergent, la robotique est devenue un acteur important. Ce projet est axé sur les véhicules guidés automatisés (AGV), les véhicules autonomes au cœur des opérations d'entreposage, des installations de fabrication ou des centres logistiques. Notre mission dans ce travail était d'augmenter la navigation et l'adaptabilité des AGV, ce qui est essentiel pour des opérations efficaces dans des environnements en évolution dynamique. À cette fin, nous avons proposé l'intégration de technologies de pointe, à savoir le Robot Operating System (ROS), Simultaneous Localization and Mapping (SLAM), et l'algorithme A*. Le projet a été exécuté en deux phases principales : une phase d'étude et de simulation et une phase de réalisation. Dans la première phase, nous avons cherché à mettre en œuvre des fonctionnalités de navigation automatique et de création de map. La deuxième phase a été conçue pour traduire nos connaissances théoriques de la phase de simulation en un prototype d'AGV fonctionnel.
Mots-clés : Véhicules guidés automatisés (AGV), système d'exploitation robotisé (ROS), algorithmes de localisation et de cartographie simultanées (SLAM), algorithmes A* (A-star).

ملخص :

في عصر الصناعة 4.0، حيث تلتقي العوالم الرقمية والمادية، ظهرت الروبوتات كلاعب مهم. تركز هذه الأطروحة على مجال محدد داخل الروبوتات وهو مجال المركبات الآلية الموجهة(AGVs) ، تستخدم المركبات ذاتية القيادة المركزية للعمليات في المستودعات أو مرافق التصنيع أو المراكز اللوجستية. كانت مهمتنا في هذا العمل هي زيادة الملاحة والقدرة على التكيف فيAGVs ، وهو أمر بالغ الأهمية للعمليات الفعالة في البيئات المتغيرة ديناميكيًا. ولهذه الغاية، اقترحنا دمج التقنيات المتقدمة وهي نظام تشغيل الروبوت(ROS) ، والتوطين ورسم الخرائط في وقت واحد(SLAM) ، وخوارزمية* A تم تنفيذ المشروع على مرحلتين رئيسيتين: مرحلة المحاكاة ومرحلة التنفيذ. في مرحلة المحاكاة، هدفنا إلى تنفيذ قدرات الملاحة الذاتية وبناء الخرائط. من ناحية أخرى، تم تصميم مرحلة التصنيع لترجمة رؤيتنا النظرية من مرحلة المحاكاة إلى نموذج أولي عامل لـAGV .
كلمات مفتاحية : المركبات الموجهة الآلية (AGV)، نظام تشغيل الروبوت (ROS)، خوارزميات التوطين ورسم الخرائط المتزامنة (SLAM)، خوارزميات **(**A-star) * A .

Table of Content

General Introduction ..... - 1 -
Chapter I. AGV and Robotic ..... - 4 -
I.1 Introduction ..... - 5 -
I.2 Robotics ..... - 5 -
I.2.1 Definition of Robotics ..... - 5 -
I.2.2 Robotics and Mechatronics ..... - 6 -
I.2.3 Types of Robots ..... - 7 -
I.2.4 Applications of Robotics ..... - 8 -
I.3 Automated Guided Vehicles (AGVs) ..... - 9 -
I.3.1 Types of AGVs in industry ..... - 9 -
I.3.2 Key Components of an AGV System ..... - 11 -
I.4 Embedded Systems ..... - 11 -
I.4.1 Definition of Embedded Systems ..... - 11 -
I.4.2 Components of an Embedded System ..... - 11 -
I.4.3 Applications of Embedded Systems ..... - 12 -
I.5 Intersection of Embedded Systems and Robotics ..... - 13 -
I.5.1 How Embedded Systems Power Robotics ..... - 13 -
I.5.2 The Role of Embedded Systems in AGVs ..... - 14 -
I.6 Description of AGV project ..... - 14 -
I.7 Conclusion ..... - 15 -
Chapter II. AGV Control ..... - 16 -
II.1 Introduction ..... - 17 -
II.2 Achieving Autonomous Navigation ..... - 17 -
II.3 SLAM Algorithm of localization ..... - 18 -
II.3.1 Types of SLAM ..... - 18 -
II.3.2 Process of slam ..... - 19 -
II.4 A-star Algorithm of path-planning ..... - 19 -
II.4.1 Process of A-star ..... - 20 -
II.5 Robot operating system (ROS) ..... - 21 -
II.5.1 ROS's Key Features ..... - 21 -
II.5.2 Core Components of ROS ..... - 21 -
II.6 Implementation of algorithms (SLAM, A*) ..... - 22 -
II.7 Conclusion ..... - 24 -
Chapter III. Study of AGV ..... - 25 -
III.1 Introduction ..... - 26 -
III.2 Functional Analysis ..... - 26 -
III.2.1 External Functional Analysis ..... - 26 -
III.2.2 Internal Functional Analysis: ..... - 30 -

III.3 Choice of motor ..... - 32 -
III.4 Grafcet ..... - 33 -
III.5 Conception ..... - 34 -
III.4.1 Presentation of components ..... - 34 -
III.4.2 Assembling component ..... - 35 -
III.5 Strength of materials ..... - 37 -
III.6 Conclusion ..... - 39 -
Chapter IV. Simulation AGV in ROS ..... - 41 -
IV.1 Introduction ..... - 42 -
IV.2 The Significance of Simulation in Robotics ..... - 42 -
IV.2.1 ROS Master ..... - 43 -
IV.2.2. Node ..... - 43 -
IV.2.3 Publisher ..... - 44 -
IV.2.4 Subscriber ..... - 44 -
IV.2.5 Topic ..... - 44 -
IV.2.6 Services ..... - 44 -
IV.2.7 Gazebo ..... - 44 -
IV.2.8 RViz ..... - 45 -
IV.3 Robot Representation ..... - 47 -
IV.4 Implementing ROS 2 Controllers ..... - 49 -
IV.5 Virtual Environment Gazebo ..... - 50 -
IV.6 Map Creation and Localization SLAM ..... - 51 -
IV.7 Achieving Autonomous Navigation ..... - 52 -
IV.8 Conclusion ..... - 54 -
Chapter V. Realization of AGV ..... - 55 -
V.1 Introduction ..... - 56 -
V.2 AGV Assembly Steps ..... - 56 -
V.2.1 Mechanical subsystem ..... - 56 -
V.2.2 Electrical Subsystem ..... - 57 -
V.2.3 Electronic subsystem ..... - 58 -
V.3 Motor Test: ..... - 59 -
V.3.1 Equilibration and speed motors ..... - 59 -
V.3.2 Motors problems: ..... - 60 -
V.4 Manuel control ..... - 62 -
V.4.1 Joystick control ..... - 62 -
V.4.2 Application control ..... - 62 -
V.5 Visualization ..... - 63 -
V.5.1 Lidar visualization ..... - 63 -
V.5.2 Camera visualization ..... - 64 -
V.6 Real Robot Test ..... - 65 -

V.6.1 Manuel control: - 65 -
V.6.2 Camera visualization in the robot: - 66 -
V.7 Conclusion - 67 -
GENERAL CONCLUSION - 68 -
References - 70 -
Annex - 72 -

List of Figures

Chapter I

Figure I.1 Example of Robot - 5 -
Figure I.2 Revolutionizing Industries: The Mighty KUKA Robots Taking the Lead - 8 -
Figure I.3 Unit load AGVs - 9 -
Figure I.4 Tow AGVs - 9 -
Figure I.5 Pallet Truck AGVs - 10 -
Figure I.6 Fork lift AGVs - 10 -
Figure I.7 Load AGVs - 10 -
Figure I.8 Assembly Line AGVs - 10 -
Figure I.9 ST NUCLEO-L476RG Cortex-M4 mbed STM32 Microcontroller Development Board for Arduino - 12 -
Figure I.10 Inputs and outputs of embedded system (4) - 12 -
Figure I.11 Structure of embedded system (4) - 12 -

Chapter II

Figure II.1 process of SLAM [19] - 19 -
Figure II.2 Process of A-star [20] - 20 -
Figure II.3 process of build AGV - 23 -
Figure II.4 process of control of AGV - 23 -
Figure II.5 Implemantation of Algorithms - 24 -
Figure II.6 SLAM Mapping with A* path planning - 24 -

Chapter III

Figure III. 1 Horned Beast Diagram - 27 -
Figure III. 2: Octopus diagram - 28 -
Figure III.3: FAST Diagram - 30 -
Figure III.4: Lithium Battery - 31 -

Figure III.5: BLDC Motor ..... - 31 -
Figure III.6: Odrive v 3.6 ..... - 31 -
Figure III.7: LiDAR Sensor ..... - 31 -
Figure III.8: Raspberry carte ..... - 31 -
Figure III.9: Arduino carte ..... - 31 -
Figure III.10 Grafcet ..... - 34 -
Figure III.11 : Robot Structure ..... - 35 -
Figure III.12 : Robot Caster wheels ..... - 35 -
Figure III.13 : Bolts ..... - 35 -
Figure III.14 : Motor Support ..... - 35 -
Figure III.15 : Rasbperry Pi-4 ..... - 35 -
Figure III.16 : Arduino Mega ..... - 35 -
Figure III.17 O-drive ..... - 35 -
Figure III.18 : LiDAR Sensor ..... - 35 -
Figure III.19 : Baterry 36V ..... - 35 -
Figure III.20 : BLDC motor ..... - 35 -
Figure III.21: AGV 2D conception ..... - 36 -
Figure III.22 Robot Bottom and Top ..... - 36 -
Figure III.23 Front Side of Robot ..... - 36 -
Figure III.24 Front and back of Robot ..... - 37 -
Figure III.25 Robot with LiDAR Sensor ..... - 37 -
Figure III.26: The high platform with load applies ..... - 38 -
Figure III.27: The result of simulation ..... - 38 -
Figure III.28: The low platform with load applies ..... - 39 -
Figure III.29: The result of simulation ..... - 39 -
Figure III.30: Low platform after modification ..... - 39 -

Chapter VI

Figure IV.1 A typical ROS Model: System components. ..... - 43 -
Figure IV.2 An Example of a Spawner Node. ..... - 43 -
Figure IV.3 Visual Representation of Node, Service, and Topic Architecture ..... - 44 -
Figure IV.4 Simulation of YB425 Robot with Camera and Li-DAR in Gazebo Environment - 45 -

Figure IV.5 Visualization of YB425 Robot with Camera View and Li-DAR Obstacle Detection in Rviz Simulation Environment.....- 45 -
Figure IV.6 Visualization of YB425 Robot in Gazebo and Rviz Environments.....- 46 -
Figure IV.7 URDF XML Representation of the YB425 Robot.....- 47 -
Figure IV.8 Representation of the Left Wheel as a Rigid Link in the Robot URDF Code.....- 47 -
Figure IV.9 Left Wheel Joint Connecting the Wheel (Link) with the Robot Base Link (Chassis) in the Robot URDF Code.....- 47 -
Figure IV. 10 Camera Sensor Configuration in the Robot URDF Code.....- 48 -
Figure IV.11 Digital Representation of Full Structure of URDF code in Gazebo Simulation: YB425 Robot with LiDAR at top and Front Camera.....- 48 -
Figure IV.12 Figure IV.12 Adding Controller for Left and Right Wheels to Enable Joystick Commands - Velocity and Position Control.....- 49 -
Figure IV.13 Code for Node Loading the Joystick Controller Command .....- 49 -
Figure IV.14 Robot Navigation and Obstacle Sensing in the Virtual Environment .....- 50 -
Figure IV.15 Virtual Environment of the Project - Gazebo Simulation of Real-Life Room Design. .....- 50 -
Figure IV.16 Map Creation of the Virtual Environment using SLAM Toolbox (Cartography)- 51 -
Figure IV.17 Full Map of the Environment for Autonomous Navigation.....- 51 -
Figure IV. 18 Robot and Virtual Environment Loading (Right) and Map Localization (Left).- 52 -
Figure IV.19 YB425 Robot Reaching the Goal via Shortest Path (A* Algorithm) .....- 53 -
Figure IV. 20 YB425 Robot Path Planning with Obstacle Avoidance.....- 53 -
Figure IV. 21 YB425 Robot Path Planning with Multiple Goals and Dynamic Path Generation .....….- 54 -

Chapter V

Figure V.1 AGV structure .....- 56 -
Figure V.2 AGV wheels .....- 57 -
Figure V.3: Electric circuit of motor .....- 58 -
Figure V.4:Odrive v3.6 Figure V.5: Raspberry Pi Figure V.6: LiDAR Sensor .....- 58 -
Figure V.7: Electronic circuit .....- 59 -
Figure V.8: Different test of motors .....- 60 -

Figure V.9: Error of noise between motor.....- 61 -
Figure V.10: Solve problem of noise error.....- 61 -
Figure V.11: Joystick Manuel control diagram.....- 62 -
Figure V.12: App mobile interface.....- 63 -
Figure V.13: Visualization of LiDAR in RViz.....- 64 -
Figure V.14: visualization camera in RViz.....- 65 -
Figure V. 15 controlling real robot with joystick.....- 65 -
Figure V.16 the video streamed from camera of robot via mobile phone.....- 66 -
Figure V.17 multiple device access to camera.....- 66 -

List of Tables

Chapter I

Table I. 1 Types of AGVs.....- 9 -

Chapter III

Table III.1 table question of Horned beast diagram.....- 26 -
Table III.2 Table functional specification.....- 28 -
Table III. 3 technical specification table.....- 31 -
Table III.4 Table Part component of AGV.....- 35 -

List of Abbreviations

AGV	Automated guided vehicle
LiDAR	Light Detection and Ranging
A*	A-Star
SLAM	Simultaneous Localization and Mapping
ROS	Robot operating system
URDF	Unified Robot Description Format
RViz	Robot visualization
AI	Artificial Intelligence
HRI	Human-Robot Interaction

ML	Machine learning
PWM	Pulse Width Modulation
RTOS	real-time operating systems

General Introduction

We live in the era of Industry 4.0, an age in which the boundaries between the physical and digital world blur, leading to a new level of interconnectedness and automation. Within this context, robotics has played a pivotal role, transforming manufacturing and logistics and revolutionizing operations across various industries. One notable development that has significantly influenced this field is the advent of Automated Guided Vehicles (AGVs).

AGVs are autonomous vehicles primarily used for material handling and transportation within a warehouse, manufacturing facility, or logistic center. They follow a predetermined path, guided by a combination of software and sensor-based systems. Utilizing technologies such as laser scanners and encoders, AGVs can maneuver around an environment, avoid obstacles, and achieve their tasks with minimal human intervention.

Despite their current effectiveness, there is significant room for improvement and optimization within AGV systems. Persistent challenges in navigation, mapping, and path planning limit the full potential of these vehicles. A crucial problem lies in the robustness of AGVs to efficiently adapt and navigate within dynamic environments, where the layout or the positions of objects can change frequently.

In response to this issue, we propose a work on AGV development based on the integration of ROS (Robot Operating System), SLAM (Simultaneous Localization and Mapping), and the A* algorithm. By combining these technologies, we hope to enable AGVs to navigate more efficiently and adapt to dynamic changes within their operational environments. These algorithms were chosen as part of the Master's thesis (MFE).

ROS provides a flexible framework for writing robot software. In combination with SLAM, a method used by robots and autonomous vehicles to build up a map within an unknown environment while simultaneously keeping track of their location, it can significantly improve AGV autonomy and efficiency. Conversely, the A* algorithm, a widely used pathfinding and graph traversal algorithm, can enhance AGV's ability to plan the most efficient route to its destination.

This thesis aims to explore these technologies in-depth, understand their potential and limitations, and integrate them into the design and realization of an AGV. We aim to push the boundaries of what is currently possible and develop an AGV system that is more intelligent, adaptable, and efficient, embodying the principles of Industry 4.0.

This thesis is structured as follows: The first chapter provides a comprehensive introduction to Embedded Systems and robotics, focusing specifically on Automated Guided Vehicles (AGVs), their various types, and their primary components. The second chapter delves into AGV control systems and navigation algorithms. Chapter three begins the project study,

outlining a functional analysis and Resistance of material (RDM) for the 3D design of the AGV. The fourth chapter details the simulation of the YB-425 AGV within the Robot Operating System (ROS) platform, including its visualization through the RViz simulator. Finally, the fifth chapter describes the physical realization of the AGV, explaining system modeling, mechanical and electronic components, and the computational aspects of the project. The outcomes of this study should offer valuable insights into the development and application of AGVs, contributing towards the broader efforts to realize the full potential of Industry 4.0.

Chapter I. AGV and Robotic

I.1 Introduction

Integrating robotic and embedded systems has transformed industries and reshaped our interaction with technology. Mechatronic engineering serves as the interface between these fields, enabling the development of intelligent machines. The integration of Autonomous Guided Vehicles (AGVs) within mechatronics exemplifies the synergy between mechanical design, electronics, control systems, and embedded software. AGVs are specialized robotic vehicles designed for efficient material handling and transportation within controlled environments. Understanding this relationship is crucial for harnessing the potential of robotic systems and advancing industrial automation. The purpose of this chapter is to describe the basic concepts of robotics and AGVs.

I.2 Robotics

Robotics is an interdisciplinary field that merges concepts from mechanical engineering, electrical engineering, computer science, and others to design, manufacture, and apply robots. In their broadest sense, robots are machines capable of carrying out a series of actions autonomously or semi-autonomously. The use of robots spans diverse sectors, including manufacturing, healthcare, agriculture, defense, and service industries, among others.

The concept of robots and automatons dates back to ancient times, with descriptions of automated mechanical devices appearing in Greek mythology and Chinese history. However, the modern robotics concept took shape in the 20th century. The term "robot" was first introduced by Czech writer Karel Čapek in his 1920 play "R.U.R." (Rossum's Universal Robots). A prolific science fiction author, Isaac Asimov later popularized the term "robotics" and established the influential "Three Laws of Robotics." [1].

I.2.1 Definition of Robotics

Robotics is a multidisciplinary branch of engineering and science that involves the design, construction, operation, and use of robots and computer systems for their control, sensory feedback, and information processing. The ultimate goal of robotics is to create machines that can substitute for humans, replicate human actions, or perform tasks in hazardous environments that are unsafe for humans. The discipline of robotics combines fields like artificial intelligence, computer science, electronics, mechatronics, nanotechnology, and bioengineering.



Figure I.1 *Example of Robot*

I.2.2 Robotics and Mechatronics

Mechatronics and robotics are two intertwined fields that play crucial roles in modern technology and industry. Both fields overlap significantly and complement each other in many ways.

Mechatronics is a multidisciplinary field that combines mechanical engineering, electronics, computer science, and control systems engineering. The primary goal of mechatronics is to design and create systems or products that feature controlled movements, usually achieved by integrating sensors, actuators, and controllers. Critical examples of mechatronic systems include industrial machinery, automotive systems, and consumer appliances.

Robotics, on the other hand, is a specific application of mechatronics. It focuses on robot design, construction, operation, and application. Robots are mechatronic systems designed to interact with the world autonomously or semi-autonomously. They include components like mechanical structures (linkages, wheels, or arms), actuators (for producing motion), sensors (for perceiving the environment), and controllers (for decision-making and control).

Hence, the **relationship between mechatronics and robotics** is such that mechatronics provides the foundational knowledge and tools for creating robotic systems. Robotics applies mechatronics principles to build machines that can perform tasks and interact with their environment in a way that mimics or replaces human actions. This could range from simple repetitive tasks, such as in assembly line robots, to complex tasks requiring decision-making and adaptability, such as in autonomous vehicles or humanoid robots.

This relationship also extends into education and research. Many academic programs in mechatronics include robotics courses, and robotics research often involves mechatronics advancements. Therefore, the two fields are mutually supportive, pushing the boundaries of what machines can achieve.

Mechanical Engineering in Robotics: Any robot's core is its physical structure or body. The robot's design must be mechanically sound and suitable for the tasks it's intended to perform. Mechanical engineering principles are utilized in the design and analysis of robots, considering factors such as materials, movement dynamics, and structural integrity. A subfield of mechanical engineering relevant to robotics is kinematics, which focuses on the motion of bodies in a robotic system without considering the forces that cause the movement [2].

Electrical Engineering in Robotics: Electrical engineering plays an integral role in robotics, dealing with the design of the electrical systems that power the robots. This includes power management systems, sensor technology, motor controllers, and other electronic components. Electrical engineers work on designing circuits, wiring, and all the electrical connections within the robot. Moreover, they work on the signals used to control and get feedback from the robot's various sensors and actuators [2].

Computer Science in Robotics: A robot's brain is its control system, which is made possible through computer science. This involves writing software to control the robot and implementing algorithms that allow it to navigate its environment, process sensor data, and perform its tasks. Sub-disciplines of computer science, like artificial intelligence (AI) and machine learning (ML), are becoming increasingly critical in robotics. These technologies enable robots to learn from experience, recognize patterns, and make decisions, bringing us closer to robots that can autonomously navigate complex, real-world environments [2].

Control Theory in Robotics: Control theory in robotics deals with the behavior of dynamic systems, which includes robots. It's about designing control systems for a robot that can make it perform specific tasks, respond to changes in the environment and correct errors in its performance. For example, a robotic arm needs to move in specific ways to pick up an object, and a self-driving car needs to adjust its speed and steering based on traffic conditions [2].

Human-Robot Interaction: As robots are increasingly used in daily life and work environments, studying human-robot interaction (HRI) has become increasingly important. This involves understanding how humans and robots can communicate effectively, how robots can be designed to be safe and comfortable to interact with, and how robots can understand and respond to human behaviors and emotions [2].

Cybersecurity in Robotics: With the rise of connected robotic systems, cybersecurity has become a crucial concern in the robotics domain. Protecting robotic systems from cyber threats is critical, not only to prevent operational disruptions but also to protect sensitive data that robots may handle and to ensure the safety of humans who interact with these systems [2].

I.2.3 Types of Robots

Robots come in a variety of forms tailored to their specific functions [2]:

- **Industrial Robots** are programmable machines capable of moving and performing tasks in three dimensions. They are mainly used in manufacturing processes, including assembly lines, painting, welding, packaging, and product inspection.

- **Service Robots:** Service robots operate semi- or fully autonomously to perform services useful to the well-being of humans. They can be used in fields like medicine, agriculture, retail, and domestic chores. Examples include robotic vacuum cleaners, hospital delivery robots, and telepresence robots.

- **Mobile Robots:** These are autonomous machines that can move around in their environment, including self-driving cars, unmanned aerial vehicles (drones), underwater, and Automated Guided Vehicles (AGVs) that fall into this category.

- **Educational Robots**: These robots are used as teaching aids and tools in various levels of education, from elementary school to university research labs [2].

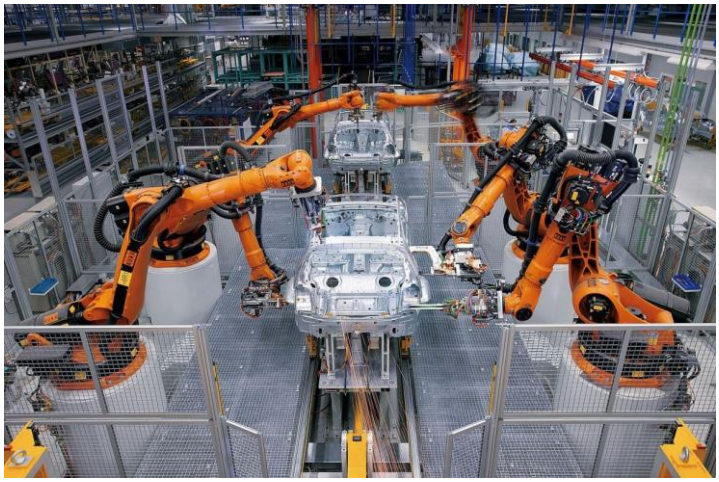


Figure I.2 *Revolutionizing Industries: The Mighty KUKA Robots Taking the Lead*

I.2.4 Applications of Robotics

Robotics has found widespread applications across diverse fields:

- **Manufacturing:** Robots are used for tasks such as assembly, packing, and quality assurance in various industries.
- **Healthcare:** Robots assist in surgery, patient care, rehabilitation, and even in drug dispensing.
- **Space Exploration:** Robots and rovers are used to explore distant planets and moons, collect data, and perform experiments.
- **Military and Defense:** Robots are employed for surveillance, reconnaissance, and bomb disposal.
- **Domestic Tasks:** Robots are now used for everyday tasks like vacuuming, mowing lawns, and even for companionship.

The field of robotics is constantly evolving, pushing the boundaries of what machines can achieve. As AI and machine learning continue to advance, robots are expected to become even more sophisticated, expanding into new domains and becoming an even more integral part of our everyday lives. The future of robotics holds exciting possibilities in areas like space exploration, disaster response, and human augmentation.

I.3 Automated Guided Vehicles (AGVs)

Automated Guided Vehicles (AGVs) are mobile robots widely used in industrial applications for material handling and transportation within a warehouse, distribution center, or manufacturing facility. They're designed to navigate autonomously, primarily using various guidance systems like laser, wire, vision, and magnetic tape.

AGVs have seen significant advancements over the past few decades, driven by technological advancements such as robotics, artificial intelligence, sensor technology, and battery technology.

I.3.1 Types of AGVs in industry

Automated Guided Vehicles (AGVs) are revolutionizing various industries' material handling and transportation tasks. Different types of AGVs have been developed to cater to specific material handling requirements. Each type of AGV possesses unique capabilities tailored to address challenges in diverse operational environments. The table below showcases some commonly used AGV types and their respective applications.

Table I.1 *Types of AGVs*

Type	Description	Example
Unit Load AGVs	These are used to transport large loads, such as pallets, rolls, or large containers.	 ***Figure I.3*** *Unit load AGVs*
Tow AGVs	Also known as tuggers, these are used to pull a chain of trailers or carts.	 ***Figure I.4*** *Tow AGVs*

Pallet Truck AGVs	These are designed to pick up and drop off pallets at designated locations.	 ***Figure I.5*** *Pallet Truck AGVs*
Fork lift AGVs	They can lift and move loads to different heights and are often used in warehouses and distribution centers. Light	 ***Figure I.6*** *Fork lift AGVs*
Load AGVs	Designed for light loads, these are often used in office environments or for transporting small parts.	 ***Figure I.7*** *Load AGVs*
Assembly Line AGVs	These are used directly in the production process, carrying parts from one workstation to another.	 ***Figure I.8*** *Assembly Line AGVs*

I.3.2 Key Components of an AGV System

An AGV system is comprised of several key components that work together to enable autonomous navigation:

- **AGV Vehicle:** The physical AGV is designed to transport materials from one location to another. The size, shape, and functionalities of the AGV can vary significantly depending on its intended use.
- **Guidance System:** The AGV requires a system to guide its movement along a path. This may include methods such as wire-guided, laser-guided, vision-guided, magnetic tape, inertial navigation, or natural navigation.
- **Sensors:** Sensors are crucial for AGVs to interact with their environment and detect obstacles. These may include LiDAR, ultrasonic sensors, cameras, or infrared sensors.
- **Communication System:** AGVs often need to communicate with a central control system or other AGVs. This may be done via Wi-Fi, Radio Frequency, or newer technologies like 5G.
- **Power Supply:** Most AGVs are battery-powered, most commonly lead-acid or lithium-ion batteries. The choice of battery can significantly affect the AGV's run time and efficiency.
- **Control System:** This is the AGV's brain. It's often an embedded system that receives input from the guidance system and sensors, processes the information, and then controls the AGV's movements accordingly.

I.4 Embedded Systems

Embedded systems are dedicated computer systems designed to perform specific tasks. Unlike general-purpose computers that can carry out a wide range of tasks based on the installed software, embedded systems are designed with a particular function in mind, and their operations are limited to that function [3].

I.4.1 Definition of Embedded Systems

An embedded system can be defined as a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific task. The term "embedded" signifies that these systems are an integral part of a larger system. They help to provide the system's functionality in whole or in part.

I.4.2 Components of an Embedded System

Microcontroller/Microprocessor: This serves as the brain of the system. It is a compact integrated circuit designed to govern the operations of the embedded system.



Figure I.9 *ST NUCLEO-L476RG Cortex-M4 mbed STM32 Microcontroller Development Board for Arduino*

Input/output Interfaces: The system can interact with the external environment. Input interfaces include sensors, buttons, or switches, whereas output interfaces involve devices like actuators, LEDs, or display screens.

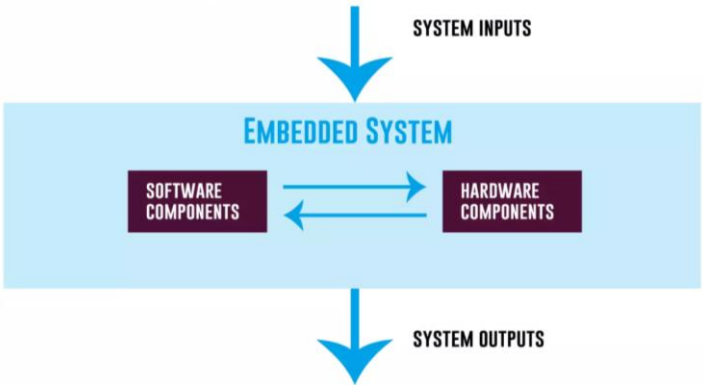


Figure I.10 *Inputs and outputs of embedded system*

Memory: This component stores the software or firmware that runs on the microcontroller or microprocessor. There are usually two types of memory in an embedded system: volatile memory (like RAM) and non-volatile memory (like Flash memory or ROM).

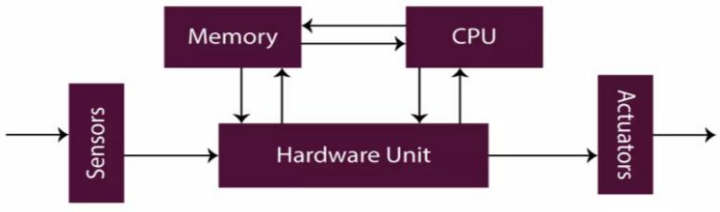


Figure I.11 *Structure of embedded system*

I.4.3 Applications of Embedded Systems

Embedded systems are omnipresent in modern life. You can find them in transportation systems (like vehicle engine control systems), medical devices (like heart monitors and pacemakers), household appliances (like microwaves and washing machines), and communication systems (like routers and phones), to name just a few examples. They are also found in industrial automation, where they form the backbone of control systems, including those for Automated Guided Vehicles (AGVs).

In conclusion, embedded systems play an indispensable role in many areas, from improving industrial processes' efficiency and safety to enhancing everyday life's convenience and comfort. A firm understanding of these systems is crucial when exploring complex automated systems like AGVs, as they form a fundamental component of their control mechanisms [3].

I.5 Intersection of Embedded Systems and Robotics

The overlap between embedded systems and robotics is quite substantial, as embedded systems serve as the foundational technology enabling robotics. They control the behavior of robots, giving them the ability to interact with their environment and perform tasks.

I.5.1 How Embedded Systems Power Robotics

Embedded systems play a key role in powering robotics. Here are some of the ways they do so:

- **Control and Computation:** Embedded systems in robots usually include microcontrollers or microprocessors, which serve as the 'brain' of the robot, microcontrollers, and microprocessors, like the ARM Cortex series or Intel Atom, each having particular benefits depending on the application. Programming languages like C/C++ and Python and frameworks such as ROS (Robot Operating System) are frequently used for creating robot control software. Advanced computational tasks include algorithms for path planning, like A* or Dijkstra's, and control strategies, such as PID (Proportional Integral Derivative) control for maintaining desired states.

- **Sensory Perception:** Robots need to interact with their environment, requiring various sensors. Embedded systems process the sensor data and translate it into meaningful information that the robot can use to make decisions or respond to its environment. Multiple sensors are used in robotics, including LiDAR for creating detailed 3D maps, depth cameras like Microsoft's Kinect, and inertial measurement units (IMUs) for determining orientation and movement. Data from these sensors are processed using techniques from fields like computer vision and machine learning. For example, convolutional neural networks (CNNs) might be used to process image data and recognize objects.

- **Communication**: Many robots need to communicate with other devices, either to receive instructions or to send data. Embedded systems handle these communication tasks using different protocols and technologies. The robots use protocols such as RS232, I2C, and SPI for intra-device communication. For wireless communication, Wi-Fi and Bluetooth are common choices. Technologies like MQTT and DDS (Data Distribution Service) are also widely used in robot communication to support real-time, reliable, high-performance messaging.

- **Real-time Processing:** Most robots operate in real-time environments and must respond promptly to changes. Embedded systems are designed to perform real-time processing,

ensuring timely and appropriate robot responses. Real-time processing in robotics is often accomplished through real-time operating systems (RTOS) like RTLinux or QNX. These systems can handle tasks with stringent timing requirements, ensuring the robot can respond promptly to environmental changes. Also, techniques like multithreading and priority scheduling can help manage multiple concurrent tasks and ensure critical tasks are completed on time [5,6].

I.5.2 The Role of Embedded Systems in AGVs

Automated Guided Vehicles (AGVs) exemplify the intersection of embedded systems and robotics. AGVs are mobile robots used for transportation tasks in various settings, such as warehouses, hospitals, and manufacturing facilities.

The embedded systems within AGVs control several critical functions:

- **Navigation:** AGVs use various types of sensors (like LiDAR, cameras, and ultrasonic sensors) to perceive their environment. The embedded system processes this sensory data to navigate the AGV along its path while avoiding obstacles.
- **Control:** The AGV's movements (starting, stopping, turning, speed control) are governed by the embedded system. This system ensures the AGV moves in the correct direction at the right speed, according to its programmed instructions.
- **Communication:** AGVs often need to communicate with a central control system or other AGVs. The embedded system handles this communication, allowing the AGV to receive instructions and send status updates or alerts in case of problems.

In conclusion, embedded systems play an indispensable role in robotics, from simple single-task robots to complex systems like AGVs. By providing computation, control, sensory processing, and communication, they empower robots to perform their tasks autonomously and effectively.

I.6 Description of AGV project

Our project is centered on developing an autonomous mobile robot capable of efficiently navigating from point A to point B within an unknown environment, with the primary goal of transporting products. To actualize this vision, we intend to leverage the power of the Robot Operating System (ROS) platform, a versatile tool that facilitates the programming and implementation of complex robotics algorithms.

We will integrate navigation and path-planning algorithms within ROS for the core functionality. These algorithms will empower our robot to understand its environment, locate itself accurately, and compute the most efficient routes, thereby ensuring optimized navigation.

In order to detect and circumvent obstacles in its path, our robot will be equipped with a LIDAR sensor, an advanced technology capable of creating detailed 3D maps of the surrounding environment. This will significantly enhance the robot's perception capabilities, allowing it to navigate through even the most intricate landscapes safely.

The entire command system will be compactly embedded within a Raspberry Pi card. The Raspberry Pi, given its potent processing capabilities and compact form factor, serves as an ideal

choice for controlling our autonomous mobile robot. It will host our ROS environment and manage the computational tasks, ranging from sensor data processing to executing the navigation commands.

I.7 Conclusion

In conclusion, Chapter 1 provided a comprehensive understanding of embedded systems and robotics, exploring their components, functions, and broad applications. We have also examined the critical role of embedded systems within robotics, as exemplified by the case of Automated Guided Vehicles (AGVs). This foundation will now set the stage for Chapter 2, where we delve deeper into AGV control, exploring pivotal algorithms like SLAM and A* and their implementation within the Robot Operating System (ROS) for autonomous navigation and operational efficiency.

Chapter II. AGV Control

II.1 Introduction

This chapter delves into the operational dynamics of Automated Guided Vehicles (AGVs), which have revolutionized today's industrial landscape. By autonomously handling materials, AGVs greatly enhance productivity, efficiency, and safety across various complex environments, such as warehouses, factories, and distribution centers. A significant part of this chapter is dedicated to understanding the key algorithms that govern AGV operations [7]: SLAM for localization and A* for path planning. These algorithms, in combination with various sensors like cameras, LIDAR, and GPS, enable AGVs to accurately ascertain their position and devise efficient routes [8,9]. In addition, we will explore the importance of the Robot Operating System (ROS) framework in integrating these elements for optimal AGV performance.

II.2 Achieving Autonomous Navigation

For an AGV to achieve autonomous navigation, it must be capable of the following:

- **Localization:** This is the problem of estimating the AGV's position and orientation (pose) within the environment. GPS can be used for outdoor localization, but it's not always reliable or precise enough, and it doesn't work indoors. Therefore, other methods such as Simultaneous Localization and Mapping (SLAM), which uses sensor data to concurrently map the environment and localize within that map, are often employed. Moreover, sensor fusion techniques, which combine data from multiple sensors (e.g., LiDAR, cameras, IMUs), are used to improve the reliability and accuracy of localization.

- **Obstacle Detection and Avoidance:** This involves the use of sensors such as lidar, radar, ultrasonic sensors, or cameras to detect obstacles in the path of the AGV. The raw sensor data is usually processed using techniques like image processing (for cameras), point cloud processing (for LiDAR), or signal processing (for radar and ultrasonic sensors) to identify potential obstacles. Once obstacles are detected, the AGV should decide how to safely navigate around them. This could involve stopping, changing direction, or adjusting speed. The specific strategy depends on the dynamics of the AGV, the nature of the obstacle, and the overall mission goals.

- **Path Planning:** This is essentially the problem of finding an optimal path from a starting point to a goal location. There are different algorithms for this, such as Dijkstra's algorithm, A* algorithm, Rapidly-exploring Random Trees (RRT), etc. The choice of algorithm depends on factors such as the complexity of the environment, the dynamics of the AGV, and the available computational resources.

  Furthermore, path planning involves consideration of constraints such as avoiding obstacles, adhering to traffic rules in the environment, and minimizing some cost (e.g., time, energy, distance). In many cases, the environment is not completely known in advance or is dynamic, which calls for online (or real-time) path planning.

- **Control and Navigation:** Once a path is planned and the AGV is localized within the environment, the AGV needs to follow the planned path. This is achieved through control algorithms that take the current pose and desired pose as input and output control commands for the AGV's actuators (e.g., motors). The controllers need to ensure that the AGV maintains the correct speed and orientation while following the path. Feedback from sensors is used to correct deviations from the path due to factors like slippage, inaccurate control commands, or dynamic obstacles.

- **Error Handling:** Despite careful planning, control, and navigation, unexpected situations or errors can occur. These could include sensor failures, actuator failures, unexpected obstacles, or drastic changes in the environment. The AGV should be designed with a robust error handling system that can detect such situations, report them, and take appropriate action. This could involve stopping safely, switching to backup systems, re-planning the path, or even asking for human intervention. The successful integration and coordination of these components and capabilities result in an AGV capable of autonomous navigation, enabling it to perform its tasks reliably and efficiently with minimal human intervention.

II.3 SLAM Algorithm of localization

It is a technique used in autonomous vehicles that enables the creation of a map and the localization of the vehicle within that map concurrently [10]. By using this technology, the vehicle is capable of mapping unfamiliar surroundings.

II.3.1 Types of SLAM

Numerous SLAM systems can be found that rely on different sensors as their main input, with some using cameras (Visual SLAM) and others using laser range scanners such as LiDAR (Lidar-based SLAM) [11].

- **Visual SLAM**

V-SLAM, or Vision SLAM, is a type of SLAM system that leverages cameras as the primary input sensors to capture visual information of unknown environments and objects [12]. The basic idea behind V-SLAM is to use the motion of the camera between consecutive frames to estimate the robot's position and orientation, and to use the visual features observed in these frames to build a map of the environment.

- **LiDAR-based SLAM**

LiDAR-based SLAM methods utilize laser range scanners to produce precise 3D or 2D scans of the environment, which are represented as point clouds. These point clouds can be used to accurately localize the robot and construct a dense map of the environment [11]. LiDAR sensors typically emit thousands of laser beams per second, allowing them to generate a 2D or 3D map of the environment in real-time.

II.3.2 Process of slam

SLAM can be defined as having two major parts: one, building a map of unknown indoor or outdoor environments, and at the same time, tracking the position or movement of the sensors and camera. The SLAM algorithm consists of five essential steps as shown in Figure II.1.

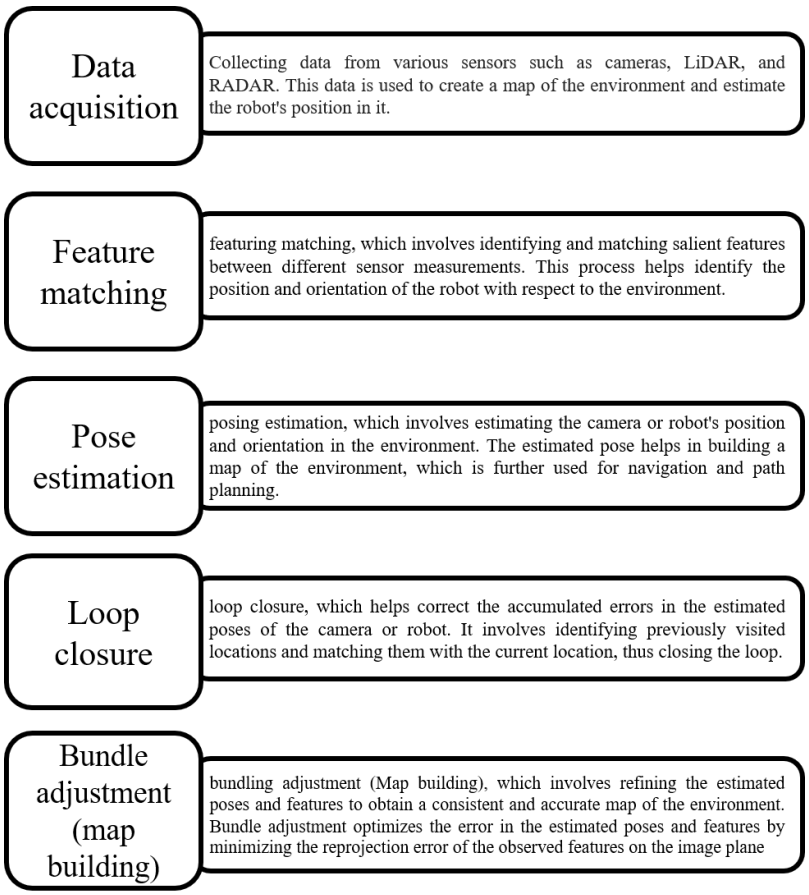


Figure II.1: *Process of SLAM [13]*

With the completion of the localization algorithm, we now transit to the path-planning algorithm, which utilizes the precise position information obtained through localization to intelligently determine the most efficient paths for AGVs, thus ensuring smooth and optimized navigation within the environment.

II.4 A-star Algorithm of path-planning

The A* algorithm also referred to as the A-star algorithm, is a popular path-planning algorithm that is widely used in various fields such as robotics and video games where an agent needs to navigate through a map or graph. It can be applied to both metric and topological configuration space. The A* algorithm is a combination of heuristic search and shortest path search, and it is classified as a best-first algorithm. This is because it evaluates each cell in the configuration space

by calculating the sum of the heuristic distance to the goal state and the length of the path from the initial state to the evaluated cell f(n)=g(n)+h(n). f(n) refers to the Estimated total path to reach the goal and its value is denoted as f(v) and is used to determine the next cell to be evaluated in the sequence. where n is the next node on the path, g(n) is the cost of the path from the start node to "n", and h(n) is a heuristic function that estimates the cost of the cheapest path from "n" to the goal [14].

II.4.1 Process of A-star

The A* algorithm works by keeping track of two lists: the open list of nodes to explore and the closed list of already explored nodes. It starts by adding the starting node to the open list and estimating the distance to the target node. The algorithm then selects the node with the lowest cost from the open list and checks its neighboring nodes. It calculates the cost to reach each neighboring node and adds them to the open list if they haven't been explored yet. If a neighboring node is already on the open list, the algorithm updates its cost and parent node if the new cost is lower than the previous cost. If the neighboring node is on the closed list, the algorithm skips it. The algorithm keeps expanding nodes from the open list until it reaches the target node or runs out of nodes to explore. If the target node is reached, the algorithm traces back through the parent nodes to find the shortest path from the starting node to the target node [15].

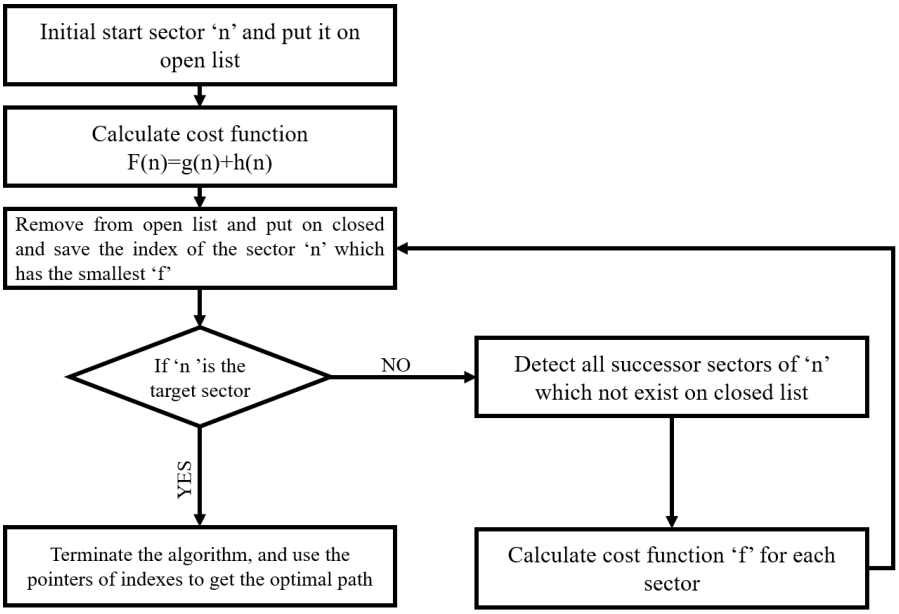


Figure II.2: *Process of A-star [15]*

For implement these algorithms in the ROBOT AGV we need a platform to realize this step to actualize these algorithms and bring them into the realm of practical application, a reliable and flexible platform is needed. This is where the Robot Operating System (ROS) steps in. ROS provides a robust, open-source framework that facilitates the integration of these algorithms into

a vast array of robotic hardware platforms, thus bridging the gap between theoretical algorithm development and real-world robotics applications.

II.5 Robot operating system (ROS)

The Robot Operating System (ROS) is not a traditional operating system as the name might suggest. Rather, it is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions designed to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [16].

II.5.1 ROS's Key Features

Modularity: ROS is designed with a distributed and modular approach. This means you can design a complex robot system as a set of independent nodes, each responsible for a small part of the system. This modular approach not only makes the design easier to understand, manage, and debug but also promotes reusability [16].

Multilingual Support: ROS supports multiple programming languages. in September 2021, the core libraries are primarily written in C++, but Python is also extensively used and well-supported. Additional experimental libraries exist for other languages like Lisp, Java, and Lua [16].

Powerful Tools: ROS comes with a suite of tools that assist in building and visualizing robotic systems. These include RViz for 3D visualization, Gazebo for robot simulation, RQT for GUI development, and more [16].

Message Passing: ROS uses a publisher/subscriber paradigm for nodes to communicate with each other. This is a very flexible way to exchange data between different parts of a robot system [16].

Broad Range of Libraries: ROS comes with a wide range of libraries that address many common tasks in robotics. These include perception, object identification, pose estimation, localization and mapping, path planning, and manipulation [17].

Active Community and Ecosystem: ROS has a very active community and ecosystem. This means you can often find existing packages that do what you need, or find help if you're having trouble [17].

Open Source: ROS is open-source software, which means that it is freely available for anyone to use, modify, and distribute. This encourages collaboration and transparency [16].

Robustness and Scalability: The newer version, ROS 2, addresses several limitations in the original ROS and provides support for real-time systems, more robust communication, and more tools for testing and debugging. This makes ROS suitable for industrial applications as well as research [17].

Hardware Abstraction: ROS provides a simplified interface to the various sensors and actuators of a robot, making it easier to write software without needing to understand the hardware details [18].

II.5.2 Core Components of ROS

ROS is built around the concept of nodes, topics, and services:

Nodes: These are the single-purpose, executable programs that are the building blocks of ROS. Each node can communicate with others in the ROS network.

Messages: This is the data format for communicating between nodes. Different types of messages are used depending on the data to be sent, such as integers, floats, strings, and more complex types.

Topics: A named bus over which nodes exchange messages. One node can publish a message to a topic, and another node can subscribe to this topic to receive the message.

Services: This is another way that nodes can communicate with each other. Unlike topics where nodes simply publish messages and other nodes receive them, a service allows for request/response communication.

Bags: Bags are a format for saving and playing back ROS message data. They're primarily used for storing data logs that are useful for post-run analysis and for developing and testing new algorithms.

Master: This provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another.

Ros out: This is ROS's built-in logging system. It's used for debugging and introspection.

Ros launch: A tool for easily launching multiple ROS nodes as well as setting parameters on the Parameter Server.

Parameter Server: A shared, multi-variate dictionary that is accessible via network APIs. This is used for storing data that is used by nodes to configure their operation.

The combination of these components gives ROS the ability to be flexible and robust, capable of controlling a wide range of robotic systems, while remaining adaptable and scalable. It's important to note that while these components might seem complex when first introduced, the modular structure of ROS means that you can start simply and expand your system as needed[16kno].

II.6 Implementation of algorithms (SLAM, A*)

The figure represents the approach proposed in this work. The implementation of algorithms for an AGV involves three crucial steps. Firstly, the process begins with the construction of the AGV itself, encompassing the design and assembly of its various components. This step also includes testing the AGV's static stress to ensure its structural integrity and operational reliability.

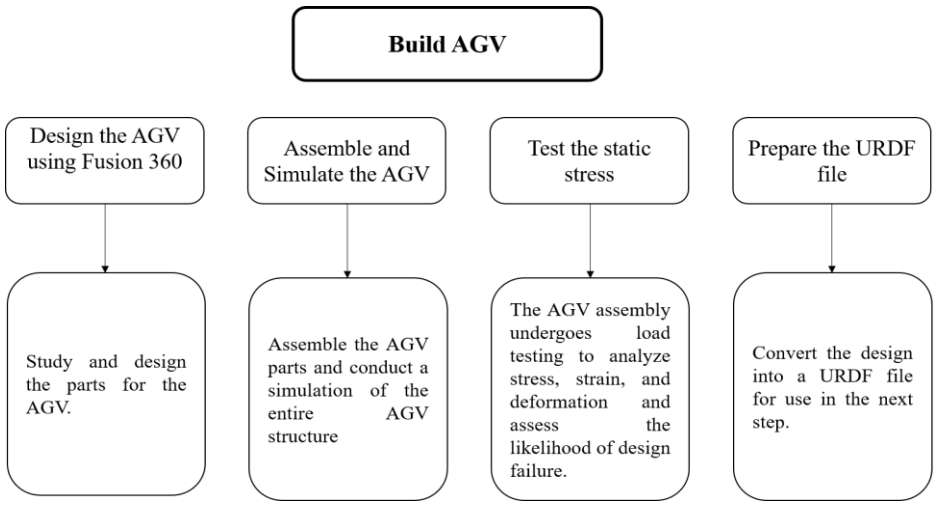


Figure II.3 *Process of Build AGV*

Once the AGV is built, the second step focuses on controlling its movements. This involves setting up the Robot Operating System (ROS) and ROS Gazebo simulation environment, as well as implementing manual control mechanisms for initial testing and fine-tuning. To achieve autonomous navigation,

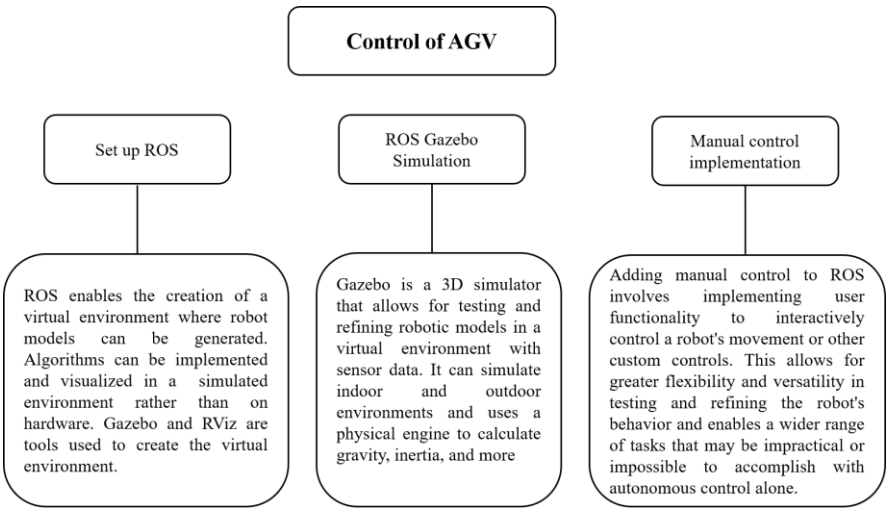


Figure II.4 *Process of Control of AGV*

The third step entails the implementation of algorithm for path planning and localization. These algorithms enable the AGV to make intelligent decisions regarding its movement, avoiding obstacles and efficiently reaching its destination. By executing these three steps building the AGV, establishing control mechanisms, and implementing path planning and localization algorithms an effective and autonomous AGV system can be realized as illustrated in figure.

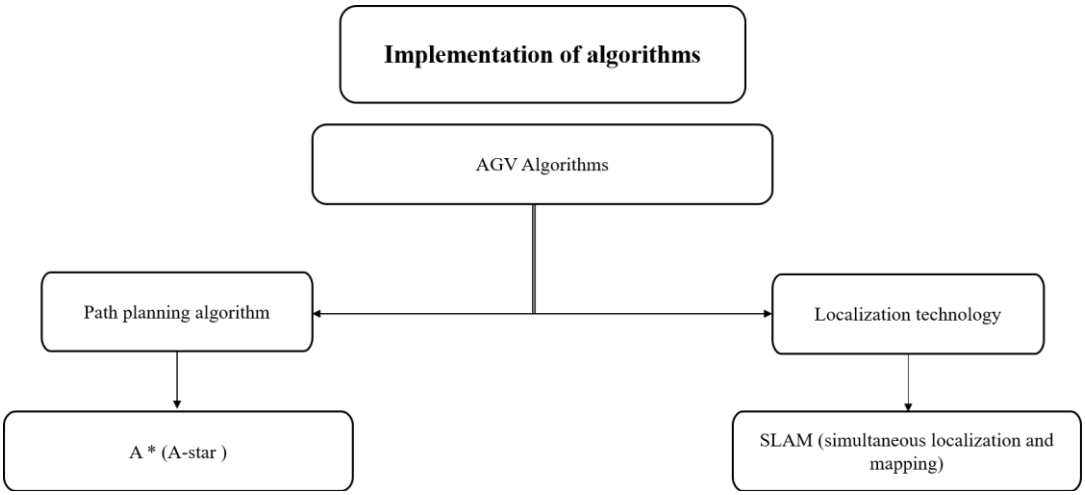


Figure II.5: *Implemantation of Algorithms*

The resulst of this chapiter is to building a virtuel map in RViz sumilator including data collecting from LiDAR sensor using SLAM algorithm and choose the chortest path with using A* algorithm the result as shows in the figure II.6

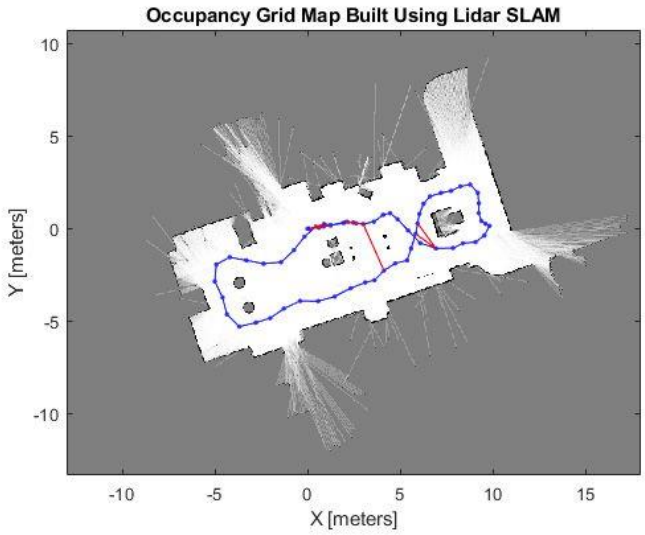


Figure II.6: *SLAM Mapping with A* path planning*

II.7 Conclusion

This chapter analyzed SLAM and A* algorithms crucial to AGV operation and the integral role of the ROS framework. The SLAM's localization, A*'s path planning, and ROS's standardization contribute significantly to AGV development. Building on this foundation, the subsequent chapter will focus on the functional analysis and conception of AGVs, translating these theoretical foundations into practical, real-world AGV application.

Chapter III. Study of AGV

III.1 Introduction

As we transition into Industry 4.0, it becomes essential to embrace cutting-edge technologies. One crucial area in this respect is robotics, with Automated Guided Vehicles (AGVs) becoming increasingly relevant due to their inherent flexibility, intelligence, and automation. We suggest harnessing the capabilities of the Robot Operating System (ROS) to integrate AGVs into Industry 4.0 seamlessly. ROS is a simplifying platform for implementing control algorithms and bridging the gap between actual design and its virtual counterpart, commonly called the Digital Twin. This approach aims to enable AGVs to optimize their handling paths, circuitry, time, and energy consumption more effectively. Hence, Industry 4.0 offers an unprecedented opportunity for digitization and optimization, significantly enhancing industrial efficiency. In this chapter, we delve into the function analysis, design, and strength testing of our AGV. We detail the technical specifications of its components, discuss the rationale behind design choices such as wheel diameter, and outline the AGV's general functionality using Grafcet. We also guide you through the conception and assembly of various AGV parts, before concluding with a comprehensive strength test to ensure its robustness. This chapter offers a concise yet thorough understanding of our AGV's operational efficiency and durability.

III.2 Functional Analysis

The functional analysis consists of searching for and characterizing the functions offered by a product placed within a system to meet needs. Depending on whether we focus on service functions or technical functions, we refer to external or internal functional analysis.

III.2.1 External Functional Analysis

The external functional analysis describes the user's point of view and only considers the product as a "black box" capable of providing services in its environment during its usage cycle.

- **Needs definition**

The first step in the design process is to analyze the situation to consider constraints, risks, and any other relevant elements. It ensures that the construction or process meets the client's needs.

The TableIII.1 answered the questions needed to introduce the Horned Beast Diagram.

Table III.1 *Table Question of Horned Beast Diagram*

Questions	Answers
Who does he serve?	Product
About what?	Handling space
For what purpose?	Move the product

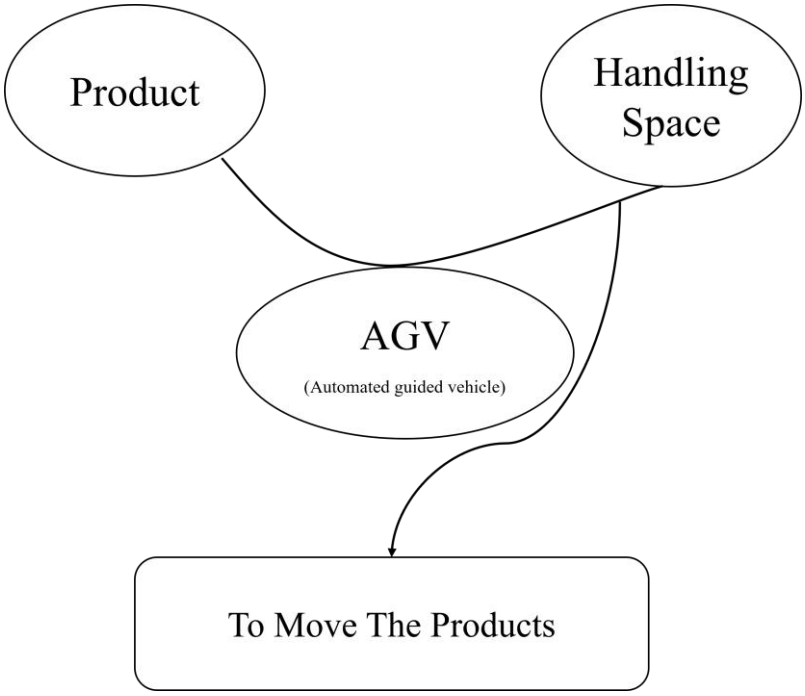


Figure III. 1 *Horned Beast Diagram*

The system serves the product by acting on the handling space in order to move the products.

Once the need is defined, we checked its stability over time. To do this, we must answer the following questions:

→ Why does this AGV system exist?
For automation of handling systems in the factory
→ Why does this need exist? For what purpose?
To develop the industry
→ What are the averages of hope with the system?
Exists and created a new technology of AGV (automated mobile robots)

- **Needs expression**

In this phase, we identify external environment elements that interact with the AGV and the Service Functions. For this, we use the 'Octopus diagram'.

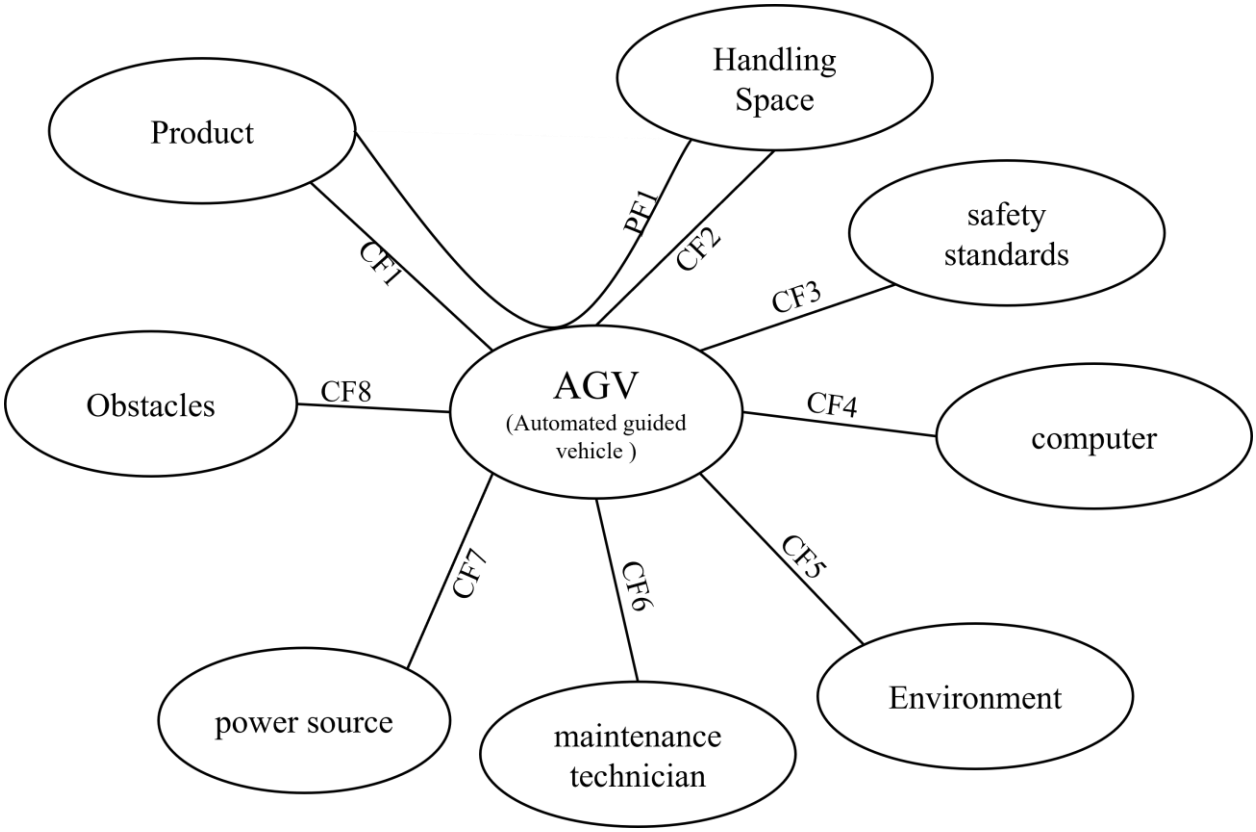


***Figure III. 2**: Octopus diagram*

PF1: Move product between workstations
CF1: Transports the product
CF2: Find the path in the Handling Space
CF3: Respect the safety standards
CF4: Pilot the AGV.
CF5: Protecting from the effects of the environment
CF6: Maintain by a specialist.
CF7: Power by the Energy
CF8: Avoid obstacles

- **Contract documents functional:**

We identify the requirements and analyze the interactions between our system and its external environment. Once we have a clear understanding of these relationships, we define the criteria and levels of requirements for each function. This information is documented in the functional specifications, which enable us to conduct internal functional analysis and determine the appropriate technological solutions for designing our system.

Table III.2: *Table functional specification*

Function	Function Name	Designation Criteria	Level	Flexibility
PF1	Move product between workstations	Implementation of the control program in order to optimize the trajectory and avoid obstacles	Raspberry	0
			ROS	0
			SLAM	0
			A-Star	0
		Speed	V=0.5m/s	± 0.1
CF1	Transports the Product	Dimension	L<60cm	1
			l<45cm	1
		Weight	99 Kg	1
CF2	Find the path in the handling space	Identical path (practical)	Shortest	1
CF3	Respect the safety standards	45001	/	/
CF4	Pilot The AGV.	Send a command	ROS SLAM A-star	/
CF5	Protecting from the environment effects	Humidity	<95%	± 5%
		Dust	IP54	/
		Impermeable	<20 %	±10%
		Temperature	<50˚	±5˚
CF6	Maintain by a specialist.	Applies a maintenance strategy	(Preventive /Corrective) Maintenance	1
CF7	Power by the Energy	Power	AC = 220v	±10 v
		DC	DC=36v	± 2 v
		AC	P=350 w	± 10 w
CF8	Avoid obstacles	Scan Lidar	Scan 5 m	± 1m

III.2.2 Internal Functional Analysis:

To fulfill the defined service functions, a product comprises various elements such as components and mechanical parts. These combined units perform technical roles that align with and satisfy service functions. We have access to a range of tools to facilitate this stage of the product's functional analysis. These will be elaborated upon in the following sections.

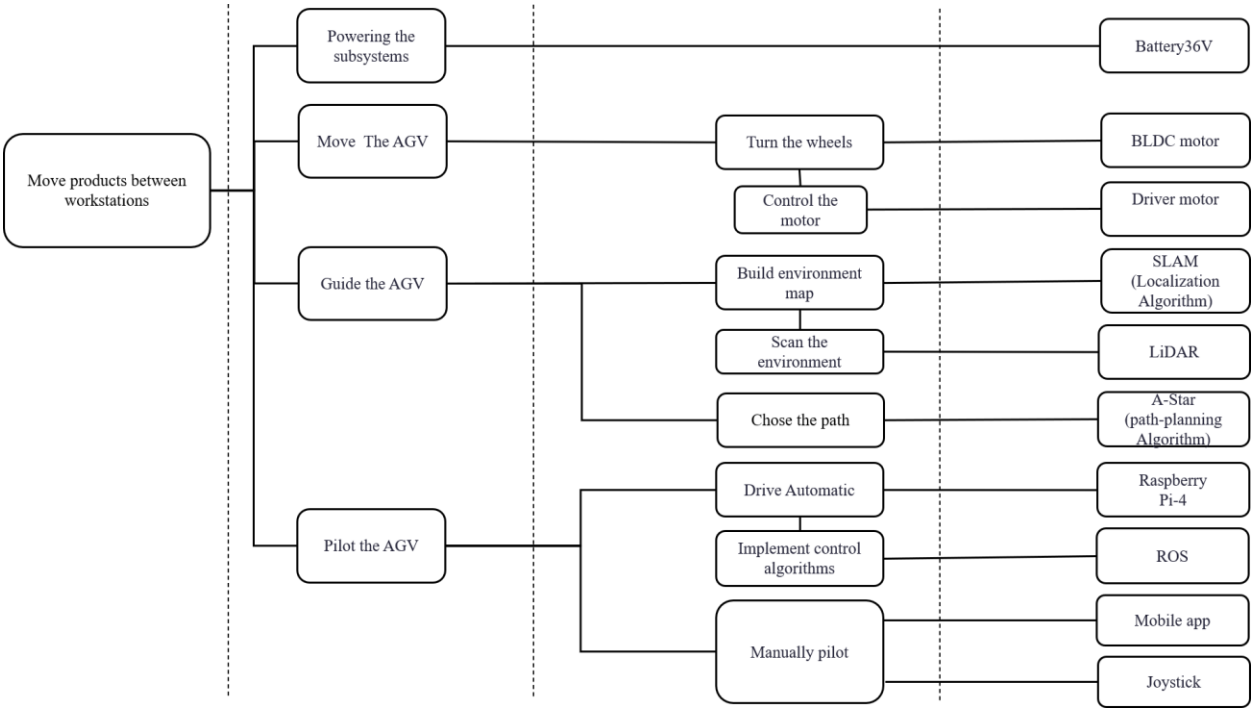


Figure III.3: *FAST Diagram*

Figure III.3 presents the FAST diagram of the Automated Guided Vehicle (AGV) system we are investigating. The system's primary function is to facilitate product movement between different workstations. This primary function is achieved through the execution of three critical technical functions: propulsion of the AGV, navigation, and control, which operate sequentially.

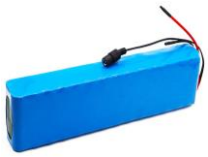
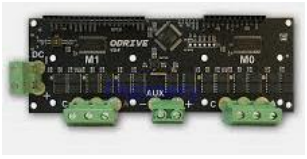
Each technical function involves specific foundational technical tasks. For instance, propelling the AGV necessitates the activation of wheels through motors, and a driver manages these motors. Furthermore, for successful AGV control, a manual learning process is essential to establish an environmental map.

Accomplishing each technical function requires applying a specific technological solution, which could take the form of a subsystem or a component. The culmination of these technological solutions makes up the AGV system under study.

III.2.3 technical specification

Table III.2 shows the various components of the system studied characteristics and their functions.

Table III. 3: *Technical specification table*

Component	Number	Property	Function	Weight	Picture
Battery	1	Batterie Lithium 36v, 10Ah (385x80x40mm)	Powering the subsystems	2 kg	 ***Figure III.4:*** *Lithium Battery*
BLDC motor	2	Revolute per minute: 600 RPM Power: 350 w Torque: 0.53	Move the AGV	2.5 kg	 ***Figure III.5:*** *BLDC Motor*
Driver motor	1	Voltage: 36v Current max: 120 A	Move the AGV	50g	 ***Figure III.6:*** *Odrive v 3.6*
LiDAR	1	Distance: 0.15-6m Degree: 0-360̊	Guide the AGV	100g	 ***Figure III.7:*** *LiDAR Sensor*
Raspberry Pi 4	1	8 GB RAM Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz wireless, Bluetooth 5.0 Gigabit Ethernet 2 USB 3.0 ports 2 USB 2.0 ports.	Pilot the AGV	50g	 ***Figure III.8:*** *Raspberry carte*
Arduino	1	5V Clock speed 16 MHz DC Current per I/O Pin 40 mA	Control motor	50g	 ***Figure III.9:*** *Arduino carte*

III.3 Choice of motor

To move our AGV with the products, we need to use DC (Direct Current) direct current motors. In this part, we make the necessary calculations to find the torque to be generated by the motors to reach a maximum speed of 0.4 m/s to choose the ideal motors.

The data:

– Gravity:

$$g = 9.81 \frac{kg.m}{s^2}$$

– Total mass (AGV + product):

$$M = 80kg$$

– Maximum speed:

$$V = 0.4 \frac{m}{s}$$

– Wheel diameter:

$$D = 15.24\ cm$$

These data are selected theoretically so the AGV can move adequately on the handling space.

• Calculation of the rotation speed:

To determine the speed of rotation, we have the following formula (III.1):

$$\omega = \frac{60V}{\pi D} \quad \text{..... (III.1)}$$

So:

$$\omega = \frac{60(0.4)}{\pi(0.1524)}$$

$$\omega = 51\ \mathbf{RPM}$$

• Calculation of motor power:

The following formula determines the power (III.2):

$$P = F.V \quad \text{..... (III.2)}$$

With:

$$F = M.g \quad \text{..... (III.3)}$$

F: the force applied to the AGV

As follow:

$$P = 80 \times 9.81 \times 0.4 = 314\ \boldsymbol{w}$$

Therefore, the power of each motor is (III.4):

$$\frac{P}{2} = \frac{392.4}{2} = 157\,\boldsymbol{w} \quad \text{..... (III.4)}$$

• **Calculation of motor torque:**

To determine the torque of the motor, we have the following formula (III.5):

$$C = \frac{P}{\omega} \quad \text{..... (III.5)}$$

As follow:

$$C = \frac{157}{50.92} = 3.07\,\boldsymbol{Nm}$$

In short, our motor must have the following characteristics:

- Power: 157 **w**
- Torque: 3.07 **Nm**
- Rotation speed: 51 **RPM**

III.4 Grafcet

The AGV is then placed in its parking, serving as well as a charging station, to ensure sufficient power for its tasks. After complete charging, the AGV is activated and transitions to an idle state, ready to receive orders.

Material pick-up or delivery orders are directly loaded into the AGV's control system. Upon receipt of an order, the AGV calculates the most efficient route from its current location to the station target, considering factors such as traffic, obstacle locations, and shortest distance.

Once the AGV arrives at the targeted station, an external system (like a robotic arm or human operator) loads/unload the designated material. The AGV then calculates the optimal path to the next target location, considering real-time environmental factors.

Upon reaching the next target location, the external system loads/unloads the material. Once the required operation is complete, depending on its battery status and the control system's directives, the AGV either awaits new orders or returns to the parking.

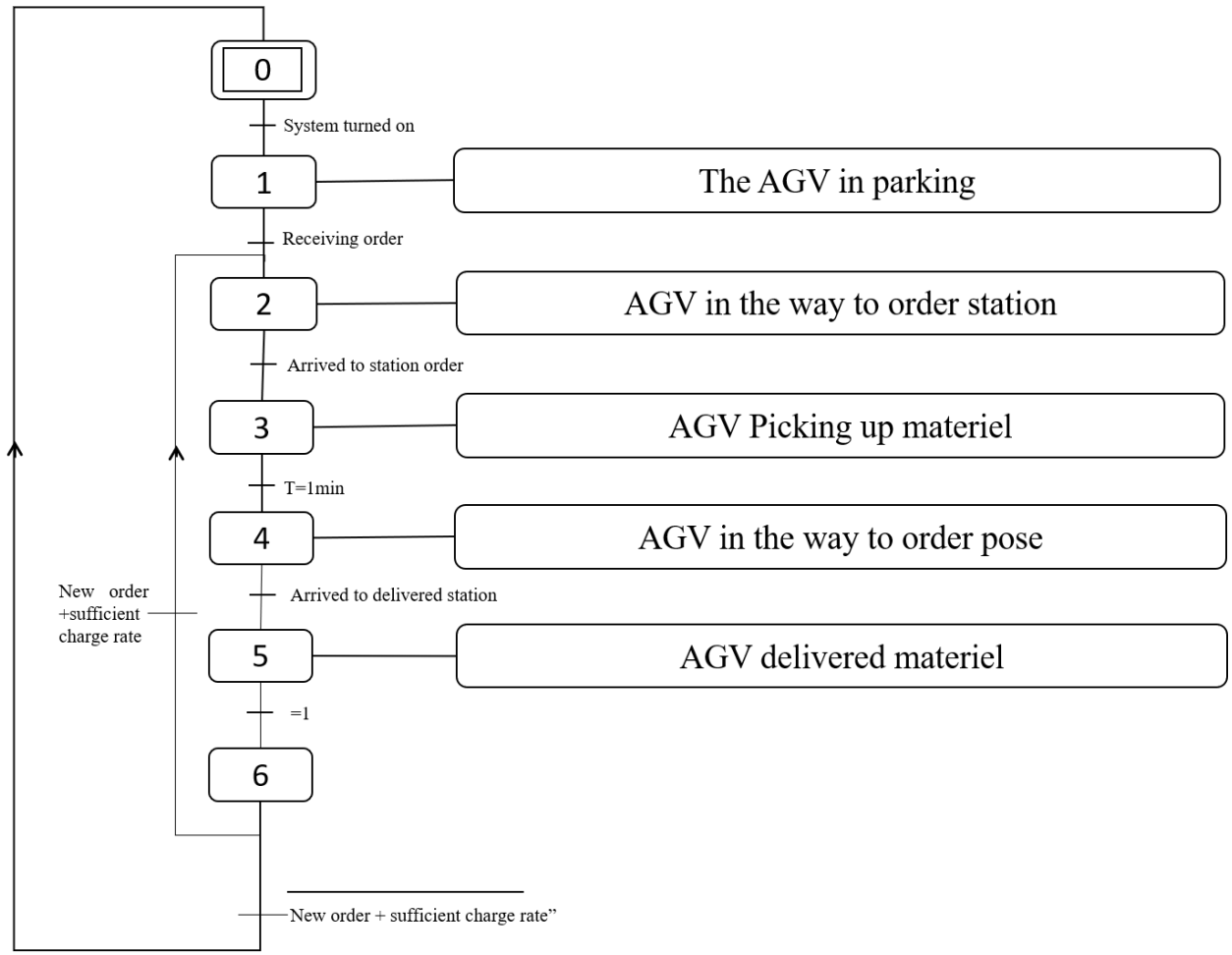


Figure III.10 *Grafcet*

III.5 Conception

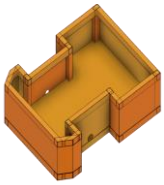
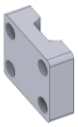
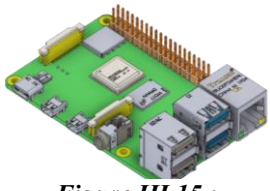
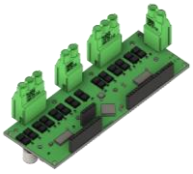
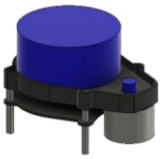
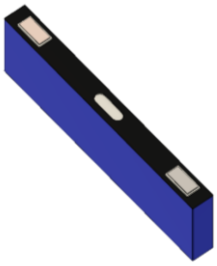
The AGV conception involves using the Fusion 360 program, taking full advantage of the software's capabilities. Starting with the design phase, we focus on creating a realistic 3D model of the AGV, detailing each component, from mechanical parts to sensor placements. Fusion 360's parametric design features allow us to quickly modify design elements as we refine our robot's structure and mechanisms.

Next, we simulate the physical properties of our AGV within the software, testing aspects such as stress points, weight distribution, and movement capabilities. This is made possible through Fusion 360's robust simulation tools, providing critical insight into how our design would behave in real-world conditions.

III.4.1 Presentation of components

In this part, we present the conception of the components used in the assembly conception of our robot AGV as presented in the table III.4

Table III.4: *Table Part component of AGV*

Components	Element 1	Element 2	Element 3	Element 4
Mechanic	 ***Figure III.11 :*** *Robot Structure*	 ***Figure III.12 :*** *Robot Caster wheels*	 ***Figure III.13 :*** *Bolts*	 ***Figure III.14 :*** *Motor Support*
Electronic	 ***Figure III.15 :*** *Rasbperry Pi-4*	 ***Figure III.16 :*** *Arduino Mega*	 ***Figure III.17*** *O-drive*	 ***Figure III.18 :*** *LiDAR Sensor*
Electric	 ***Figure III.19 :*** *Baterry 36V*	 ***Figure III.20 :*** *BLDC motor*	/	/

III.4.2 Assembling component

To complete the robot, we should assemble the different part components (mechanic, electric and electronic) as shown in figures III.21 to III.25.

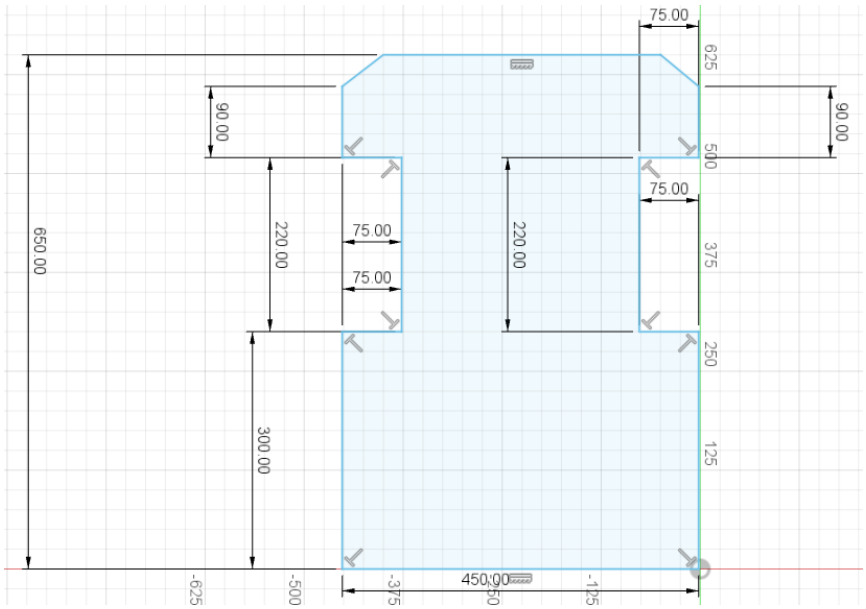


Figure III.21: *YB-425 Robot 2D conception*

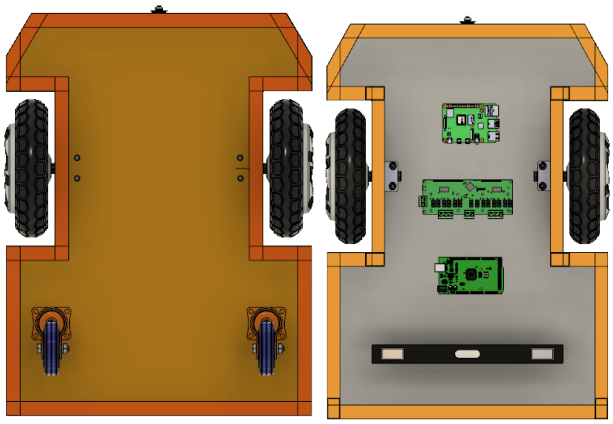


Figure III.22 *YB-425 Robot Bottom and Top*

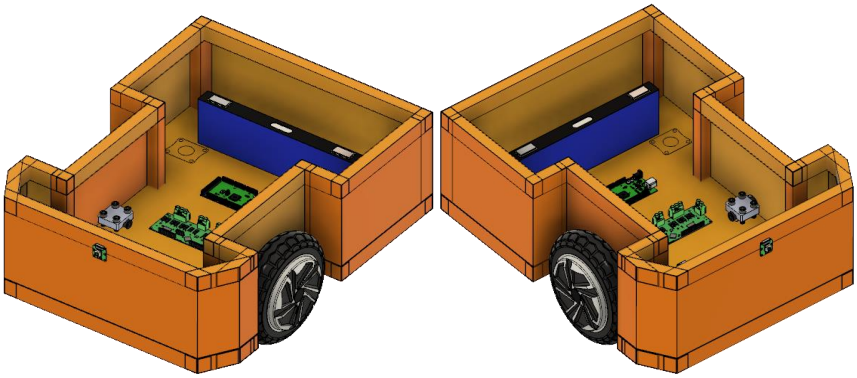


Figure III.23 *Front Side of YB-425 Robot*



Figure III.24 *Front and back of YB-425 Robot*

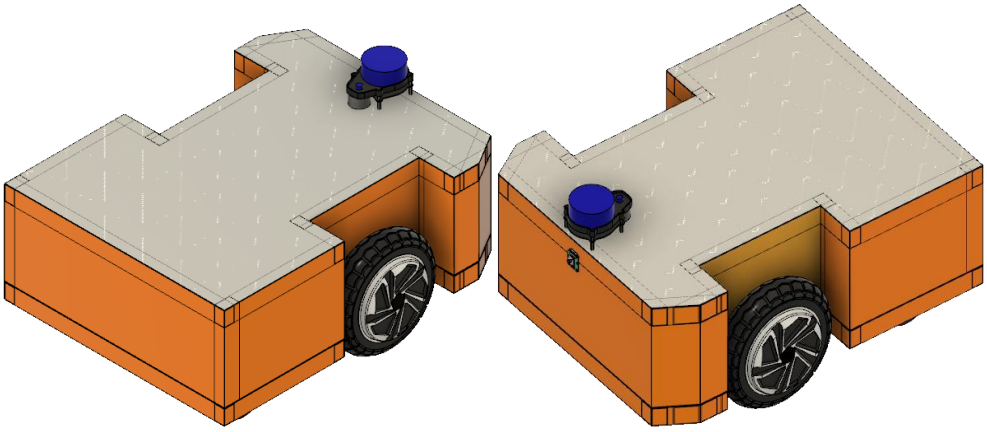


Figure III.25 *YB-42 Robot with LiDAR Sensor*

III.5 Strength of materials

The RDM study is carried out to ascertain the strength of the chassis to the various loads applied to it, namely the weights of its multiple components and the product transported. Among the materials used to design the AGV prototype, we chose steel because it is solid and supports charges.

First, we apply the load, representing the weight of the product. This load is applied on the high platform of the chassis to which the product will be attached.

The P1 load is calculated by multiplying the mass considered "M=65 **Kg**" with gravity "g=9.81 **N**"

$$P1 = M \times \boldsymbol{g} \quad \text{..... (III.6)}$$

$$P1 = 65 \times 9.81$$

$$P1 = 638\ \boldsymbol{N}$$

The figure III.26 represents the simulation results under Fusion 360 of the low platform with the load that applies P1 = 650 N.

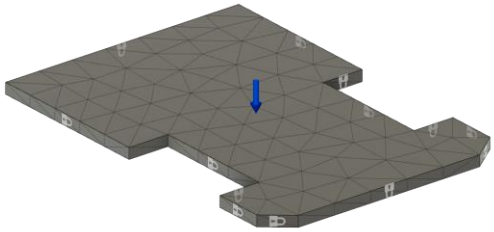


***Figure III.26**: The high platform with load applies*

We applied a force of 650 N, and the result of the study is presented in figure III.27:

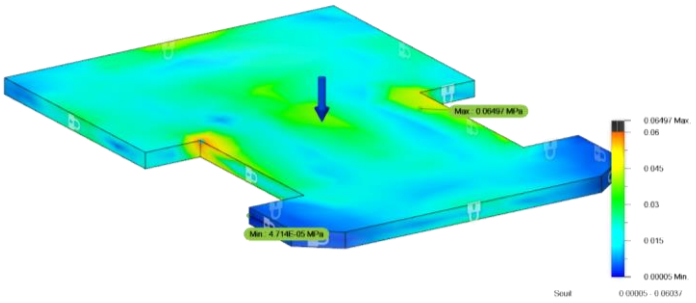


***Figure III.27:** The result of simulation*

From the figure III.27, note that the yellow angles in the edges are resistant to the strength applied by weight, and the platform did not notice any deformation because the platform thickness is reliable and bears strength.

The second load applied to the chassis represents the sum of the weights of the different components of the AGV components (motors, battery, chassis), as well as the weight of the supported product with a margin of tolerance to ensure the strength of the chassis in case of accidental introduction of a heavier product. This load is applied on the lower platform of the chassis. The P2 load is calculated by multiplying the sum of the masses of the different components and the product "M=80Kg" to gravity "g=9.81".

$$P2 = M \times g \quad \text{..... (III.7)}$$

$$P2 = 80 \times 9.81$$

$$P2 = 785\ \mathbf{N}$$

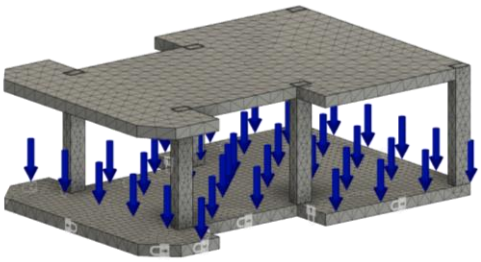


***Figure III.28**: The low platform with load applies*

We applied a force of 800 N, and the result of study is presented in figure III.29:

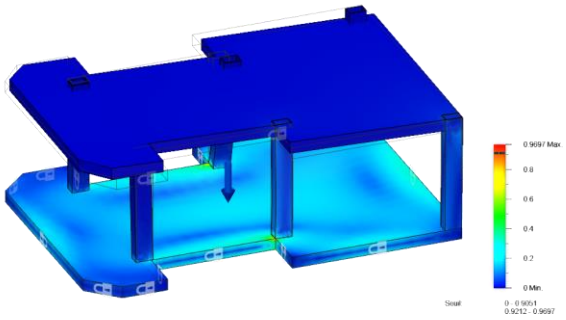


***Figure III.29:** The result of simulation*

In figure 29, we observed that there is a deformation of the chassis under the effect of the applied force, it is heavier, so the thickness of the platform must be modified to resist the applied loads as presented in figure III.30.

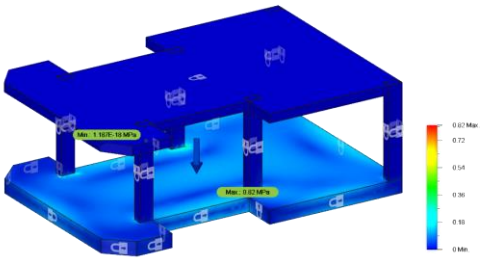


***Figure III.30:** Low platform after modification*

III.6 Conclusion

In conclusion, Chapter 3 has led us to comprehensively explore AGV design, focusing on functional analysis, Grafcet, and AGV conception in Fusion 360. We have delved into the mechanics of AGVs, gained insights into the importance of logical sequence and control systems,

and had the opportunity to create and visualize our AGV model in a digital environment. Additionally, we have examined the critical process of component selection, understanding the interplay of compatibility, performance, cost-effectiveness, and lifecycle in the success of AGV development. As we conclude this chapter, we are poised for the exciting journey ahead into AGV simulation in ROS (Robot Operating System) in Chapter 4, bridging the gap between theory and application for innovative AGV design.

Chapter IV. Simulation AGV in ROS

IV.1 Introduction

ROS provides tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across various robotic platforms. In the context of our project, we will utilize ROS to build a virtual environment and perform simulations for our AGV.

One of the key strengths of ROS is its ability to facilitate high-fidelity simulation, which allows us to recreate a real-world environment virtually and experiment with different scenarios without any risk. This ability makes it an invaluable tool for our project. Through a series of nodes, topics, and services, we can define and control every aspect of our robot's behavior while the ROS master ensures the nodes can locate and interact with each other seamlessly.

This simulation will serve as a testing ground, allowing us to iterate on and refine our robot's design and behavior before deploying it in the real world. With ROS as our foundation, we will build a virtual environment where we can visualize, test, and debug our project in a safe and controlled manner.

In this chapter, we deeply explore the pivotal role of simulation in Robotics, including essential elements such as the ROS master, node, publisher, along with tools like Gazebo and RViz. We will discuss the implementation of a ROS2 controller, and illustrate how a robot can be virtually represented in a Gazebo environment. Further, we'll delve into the process of creating a map and implementing localization techniques. We will also import the URDF file, which is a crucial step towards achieving autonomous navigation. These combined aspects offer a comprehensive understanding of simulation's importance in robotics, providing a controlled platform for designing, testing, and fine-tuning robotic systems before they're put to real-world use

IV.2 The Significance of Simulation in Robotics

Simulation plays a critical role in robotics for a multitude of reasons. With the complexity of robotic systems and their interactions with the environment, it is often impractical, expensive, and sometimes hazardous to test all scenarios in the real world. This is where simulation comes into the picture, bridging the gap between theory and practical implementation.

In robotics, simulation serves as a virtual playground for robots. It provides a safe and controlled environment where robots can be tested and trained without the fear of causing real-world damage. Furthermore, we can recreate a wide variety of scenarios in this environment, which might be too difficult, dangerous, or even impossible to replicate in reality.

A well-built simulator can mimic the physics of the real world, which includes modeling the robot's movement, sensory data, and the environment in which it operates. Such high-fidelity simulation is invaluable for verifying algorithms, testing system behavior, and refining the design based on the results.

ROS is a software development platform that facilitates inter-process communication within intelligent platforms. It operates through a system of nodes, enabling the exchange of functional

messages within a robotic architecture. ROS system comprises five essential components: ROS Master, nodes, publishers, subscribers, and topics.

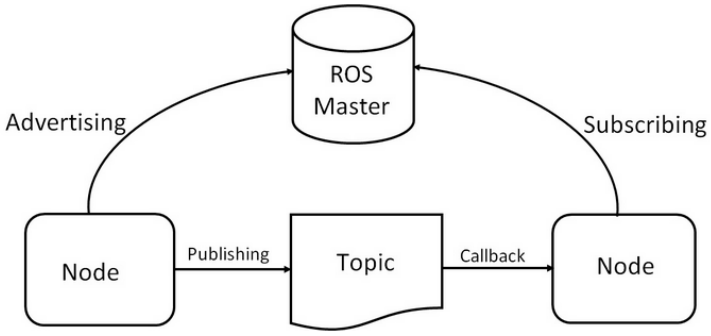


Figure IV.1: *A typical ROS Model: System components.*

IV.2.1 ROS Master

The ROS Master is responsible for managing names and registration services to the nodes within a ROS system. The ROS Master monitors publishers and subscribers to ensure associated topics and services are provided within the robotic system. The ROS Master also enables location and communication between nodes within the robotics system.

IV.2.2. Node

Nodes are individual components or processes that perform computation in a ROS system. A robot's functionality is divided into separate nodes, which can run independently and communicate with each other. For example, one node might handle image processing, another could control the robot's motion, while another node could handle obstacle detection.

```
#spawen the robot in gazebo
start_gazebo_ros_spawner_cmd = Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    arguments=[
        '-entity', YB425_MODEL,
        '-file', urdf_path,
        '-x', x_pose,
        '-y', y_pose,
        '-z', '0.01'
    ],
    output='screen',
)
```

Figure IV.2: *An Example of a Spawner Node.*

The Spawner Node in figure IV.2 is used to spawns the URDF file into the Gazebo simulation environment. This process plays a crucial role in the instantiation of simulation models.

IV.2.3 Publisher

In the context of a ROS system, a publisher refers to a message broadcasted by a node or a specific topic.

IV.2.4 Subscriber

A subscriber in a ROS system is a message that a node or topic receives.

IV.2.5 Topic

Topics serve as communication pipelines through which nodes exchange data. Each node can publish messages to a topic, subscribe to a topic to receive messages, or both. Topics enable many-to-many relationships between nodes, allowing data to be disseminated as needed throughout the system.

IV.2.6 Services

Services allow for synchronous communication between nodes in ROS. A node sends a request to another node, performs a task, and sends a response back. This type of communication is typically used when a task needs to be performed on demand, such as activating a sensor or moving a joint to a specific position. Services only provide data when a client specifically calls them. A visual representation of these behaviors is shown in Figure IV.3.

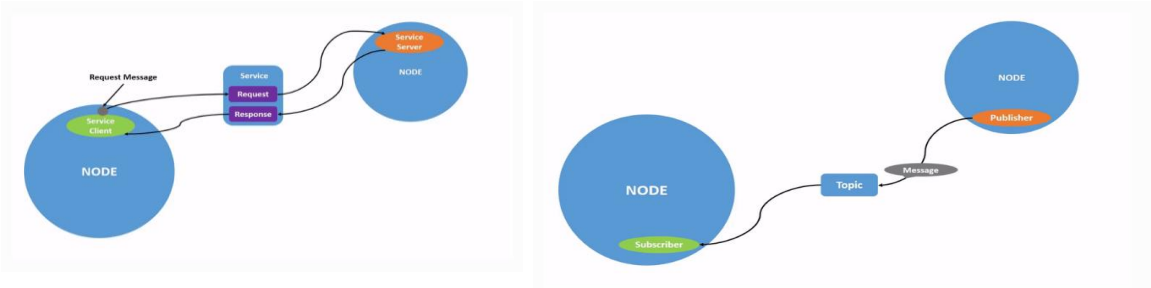


Figure IV.3 *Visual Representation of Node, Service, and Topic Architecture*

IV.2.7 Gazebo

Gazebo is a powerful 3D dynamic simulator that accurately and efficiently simulates robots in complex indoor and outdoor environments. It offers a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Integrated with ROS, Gazebo provides a comprehensive suite of tools to simulate robots, including existing models, sensor data, and control schemes.

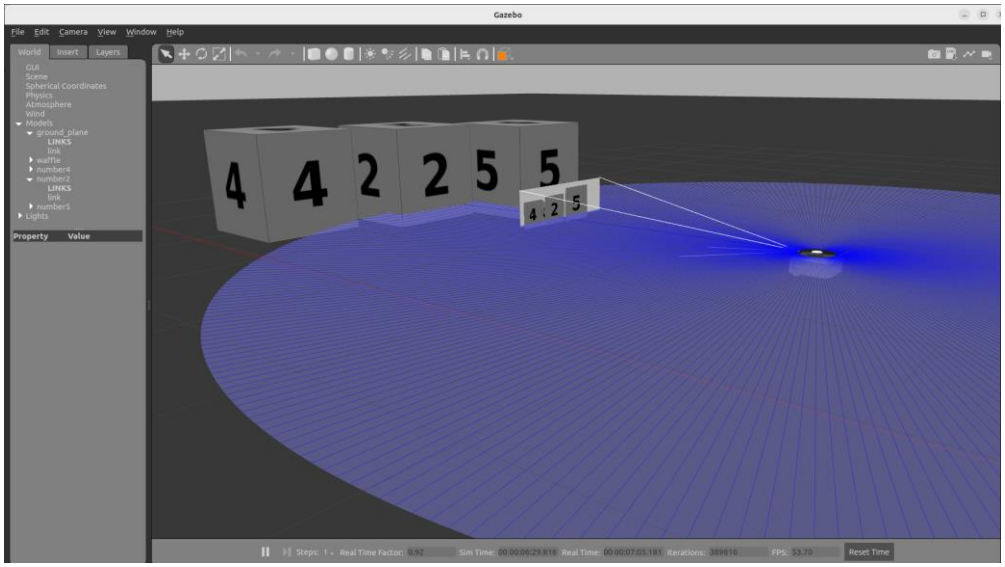


Figure IV.4 *Simulation of YB425 Robot with Camera and LiDAR in Gazebo Environment*

IV.2.8 RViz

RViz (ROS Visualization) is a 3D visualizer for displaying sensor data and state information from ROS. It provides a view into the robot's perception and current state, allowing us to see what the robot "sees" and how it's interacting with the environment (Figures IV.5 & IV.6).

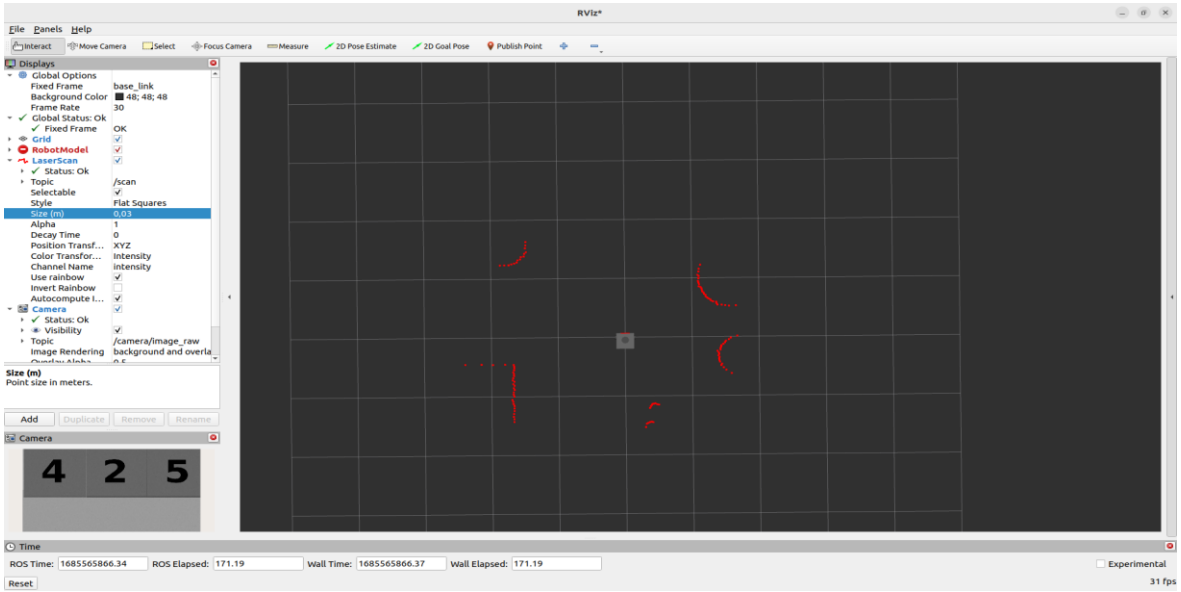


Figure IV.5 *Visualization of YB425 Robot with Camera View and LiDAR Obstacle Detection in RViz Simulation Environment*

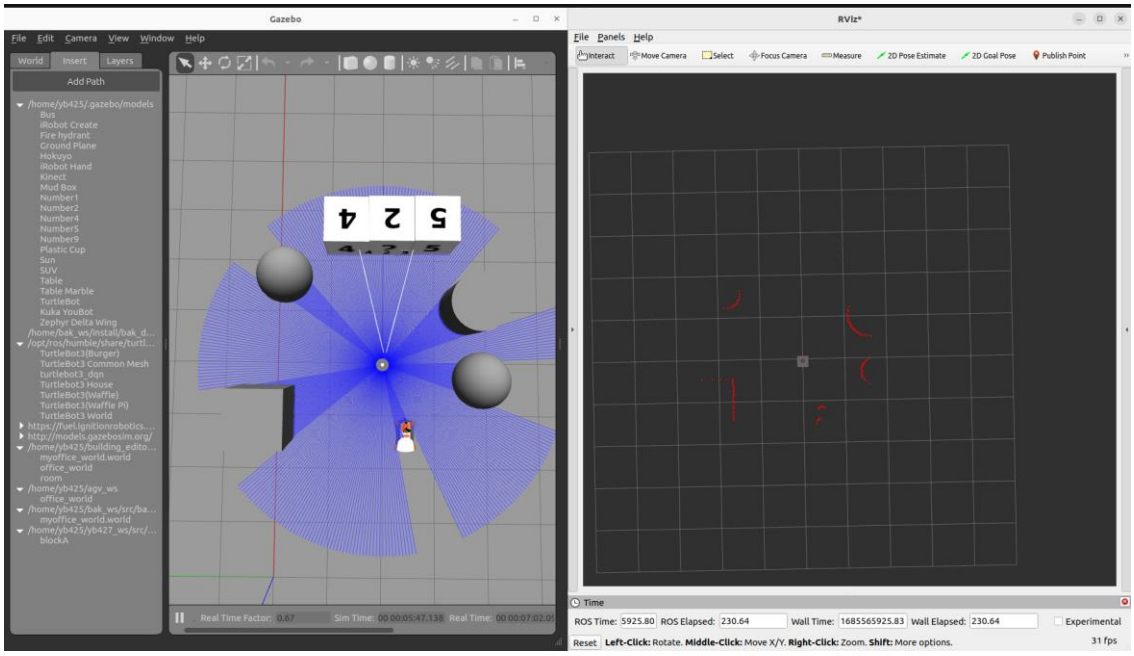


Figure IV.6 *Visualization of YB425 Robot in Gazebo and RViz Environments*

Discussion of the Figures

The Gazebo simulation environment window displays a scene with several visual elements. This scene in Figure IV.6 includes a blue vector that represents LiDAR beams. LiDAR is a remote sensing technology that uses laser beams to measure distances and create a 3D environment representation.

The blue vector in the Gazebo window (Figure IV.4) indicates the direction and orientation of the LiDAR sensor mounted on top of our robot. As the robot moves, the LiDAR beams scan the surroundings, collecting information about the distances to objects in its field of view.

Additionally, there is a red point cloud visible in the RViz window (Figure IV.5). This point cloud represents the obstacles located behind our robot. The LiDAR sensor's measurements generate the point cloud and provide a detailed representation of the objects' positions in 3D space.

Moreover, the Gazebo window displays shapes that resemble obstacles in the simulated environment. These shapes can represent various objects or structures our robot needs to navigate around or interact with during its operation. They serve as visual references to help us assess the robot's behavior and performance in different scenarios.

By having the LiDAR beams, the red point cloud of obstacles, and the shapes representing obstacles in the Gazebo window (Figure IV.6), we can effectively simulate and analyze the robot's perception and decision-making capabilities in different environments. This allows us to develop and test algorithms and control strategies for autonomous navigation and obstacle avoidance, enhancing the robot's overall performance and safety.

By harnessing these tools, we can construct and visualize a virtual environment, design our robotic system within it, and implement the required functionalities. In the next section, we will focus on how to represent our robot using URDF.

IV.3 Robot Representation

A critical aspect of working with robots in a virtual environment is how we represent them digitally. This is where the Unified Robot Description Format (URDF) comes into play. URDF is an XML representation of a robot model describing each element and its relationships.



Figure IV.7 *URDF XML Representation of the YB425 Robot*

The URDF file typically includes:

Links: These represent the rigid parts of a robot. An example of the corresponding program is presented in (Figure IV.8).

```
<link name="wheel_left_link">
  <visual>
    <origin xyz="0 0 0" rpy="1.57 0 0"/>
    <geometry>
      <mesh filename="package://bak_description/models/yb425_common/meshes/tire.dae" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="dark"/>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder length="0.018" radius="0.033"/>
    </geometry>
  </collision>

  <inertial>
    <origin xyz="0 0 0" />
    <mass value="2.8498940e-02" />
    <inertia ixx="1.1175580e-05" ixy="-4.2369783e-11" ixz="-5.9381719e-09"
             iyy="1.1192413e-05" iyz="-1.4400107e-11"
             izz="2.0712558e-05" />
    </inertial>
</link>
```

Figure IV.9: *Representation of the Left Wheel as a Rigid Link in the Robot URDF Code*

Joints: These define the connection and relative movement between two links. An example of the corresponding program is presented in Figure IV.10.

```
<joint name="wheel_left_joint" type="continuous">
  <parent link="base_link"/>
  <child link="wheel_left_link"/>
  <origin xyz="0.0 0.144 0.023" rpy="-1.57 0 0"/>
  <axis xyz="0 0 1"/>
</joint>
```

Figure IV.11 *Left Wheel Joint Connecting the Wheel (Link) with the Robot Base Link (Chassis) in the Robot URDF Code*

Sensors: These define any sensors attached to the robot, such as cameras or Lidar. An example of the Camera Sensor Configuration is presented in Figure IV. 12.

```
<joint name="camera_joint" type="fixed">
  <origin xyz="0.064 -0.065 0.094" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="camera_link"/>
</joint>

<link name="camera_link">
  <visual>
   <origin xyz="0 0 0" rpy="1.57 0 1.57"/>
    <geometry>
     <mesh filename="package://bak_description/models/yb425_common/meshes/r200.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0.003 0.065 0.007" rpy="0 0 0"/>
    <geometry>
      <box size="0.012 0.132 0.020"/>
    </geometry>
  </collision>
</link>
```

Figure IV. 13 *Camera Sensor Configuration in the Robot URDF Code*

By defining these elements and their relationships in the URDF file, we construct a full digital representation of our robot. This file serves as a blueprint for ROS to understand and interact with our robot's structure.

In our project, we created a URDF file that accurately depicts our robot, its joints, sensors, actuators, and other relevant components. This URDF file plays a pivotal role in the simulation and control of our robot within the virtual environment (Figure IV.14).

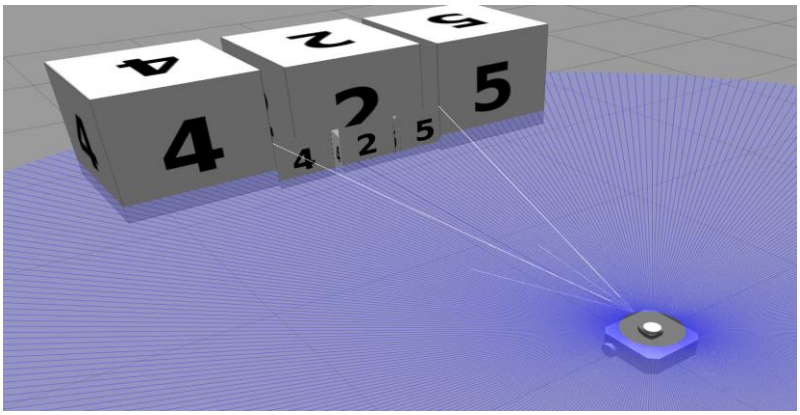


Figure IV.15 *Digital Representation of Full Structure of URDF code in Gazebo Simulation: YB425 Robot with LiDAR at top and Front Camera*

IV.4 Implementing ROS 2 Controllers

Once we have a URDF representation of our robot, the next step is to bring it to life by defining its movements and responses. A central component of this process is the implementation of ROS controllers.

Controllers are crucial in defining the behavior of our robot by translating commands into actions. They take in command messages and use them to control the robot's actuators. This can include, for example, driving a mobile robot's wheels or moving a manipulator robot's arm.

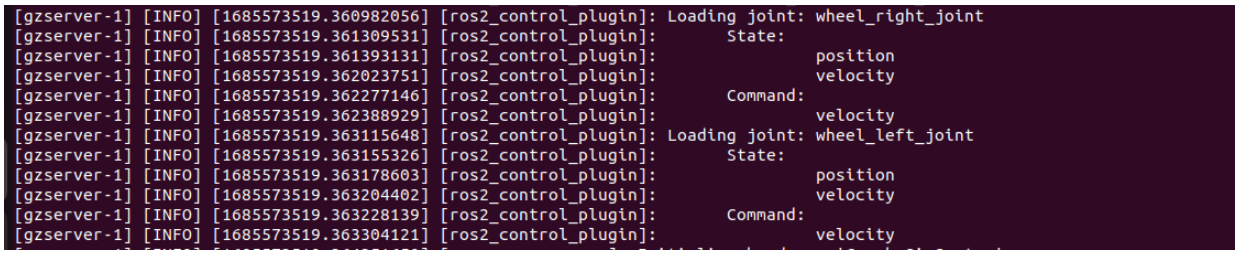


Figure IV.16: *Adding Controller for Left and Right Wheels to Enable Joystick Commands - Velocity and Position Control*

In our project, we employed ROS controllers to enable the joystick control of our robot. By mapping joystick inputs to specific commands for the robot, we can manually control the robot's movements in the virtual environment. This mapping process is achieved through a combination of ROS nodes and topics, with nodes interpreting joystick inputs and publishing corresponding command messages on designated topics.

```
#launch the joystick
joy_node = launch_ros.actions.Node(
    package='joy',
    executable='joy_node',
    parameters=[joy_params],
)
teleop_node = launch_ros.actions.Node(
    package='teleop_twist_joy',
    executable='teleop_node',
    name='teleop_node',
    parameters=[joy_params],
    remappings=[('/cmd_vel','/diff_drive_base_controller/cmd_vel_unstamped')],
)
```

Figure IV.17 *Code for Node Loading the Joystick Controller Command*

The implementation of ROS controllers allows us to interact with our robot in a flexible and intuitive way.

IV.5 Virtual Environment Gazebo

Creating a realistic and useful virtual environment is a critical step in simulating and testing our robotic system. For this, we turn to Gazebo, a powerful simulation tool that integrates seamlessly with ROS.

Gazebo provides a robust framework for simulating complex and dynamic environments in 3D. It includes realistic representations of physics, lighting, and textures, providing our robot with a virtual environment that closely mirrors the real world. It also allows for the simulation of sensor data, which is invaluable for testing our robot's perception and response systems.

In our project, we used Gazebo to construct a virtual representation of our robot's operational environment. After loading our robot's URDF file into Gazebo, we were able to visualize our robot within this environment and observe how it interacts with its surroundings.

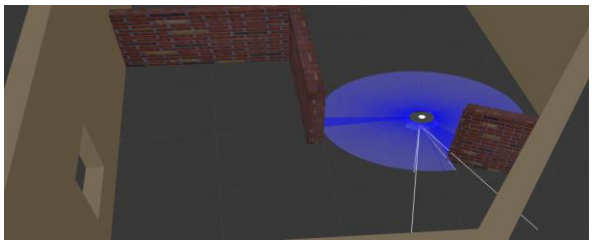


Figure IV.18 *Robot Navigation and Obstacle Sensing in the Virtual Environment*

By creating and manipulating our environment in Gazebo, we can perform extensive testing and refinement of our robot's functionalities, all within a safe and controlled setting. However, while Gazebo is excellent for simulating and observing the robot's environment as a whole, we also need a tool that allows us to visualize specific aspects of our robot's state and sensor data. For this, we utilize RViz, which we will discuss in the following section.

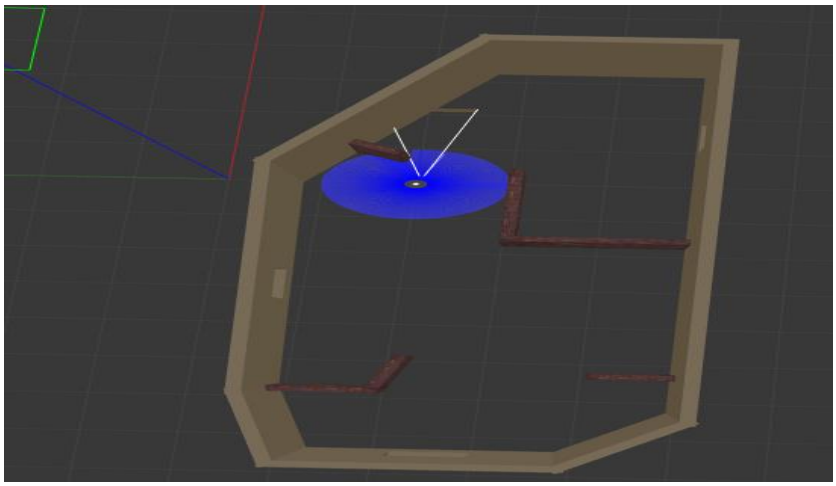


Figure IV.19 *Virtual Environment of the Project - Gazebo Simulation of Real-Life Room Design*

IV.6 Map Creation and Localization SLAM

As we continue through the construction and control of our robot, it becomes clear that understanding the environment is key to successful navigation. A robot that can accurately map its environment and know its location within that map is capable of autonomous navigation and operation. In the realm of ROS, the SLAM Toolbox provides us with this capability.

SLAM Toolbox is a powerful ROS package that enables our robot to create a map of its environment while keeping track of its location within this map. It does this by using the data from the robot's sensors (such as LiDAR or depth cameras) to perceive the surroundings and incrementally construct a map while simultaneously using this same data to estimate the robot's position and orientation within the mapped environment.

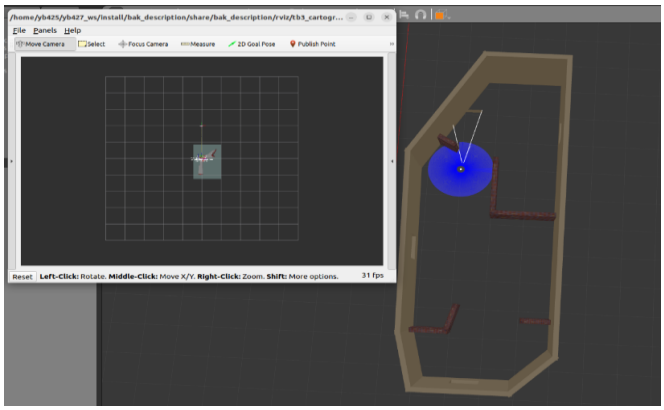


Figure IV.20 *Map Creation of the Virtual Environment using SLAM Toolbox (Cartography)*

In our project, we used the SLAM Toolbox to generate a map of our Gazebo-simulated environment. By feeding the Toolbox with sensor data from our robot, we were able to construct an accurate map of our virtual environment and track our robot's position within it.

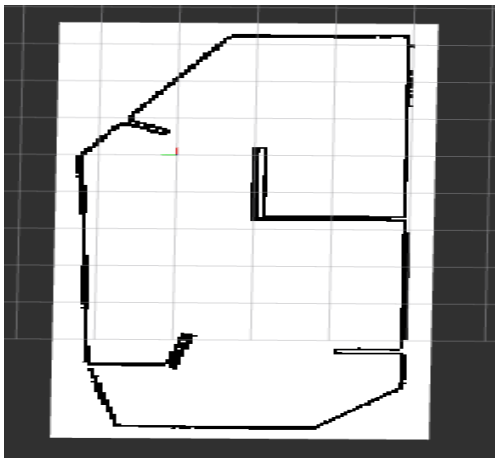


Figure IV.21 *Full Map of the Environment for Autonomous Navigation*

This ability to create a map and localize within it is critical for our next step: implementing autonomous navigation. We will explore this in the following section, where we discuss the Navigation2 (Nav2) package.

IV.7 Achieving Autonomous Navigation

Once we have a map of our environment and the ability to localize our robot within that map, the next step is to enable the robot to navigate autonomously. This is achieved through the use of the Navigation2 (Nav2) package.

Start by launching the robot model in a simulation environment. This step involves running the necessary launch files associated with the specific robot. These files typically set parameters, load configurations, and start nodes related to the robot. After that, we use SLAM to load the previously created map into RViz to set the 2D Pose Estimate. This informs the localization algorithm of the robot's rough initial location and orientation on the map.

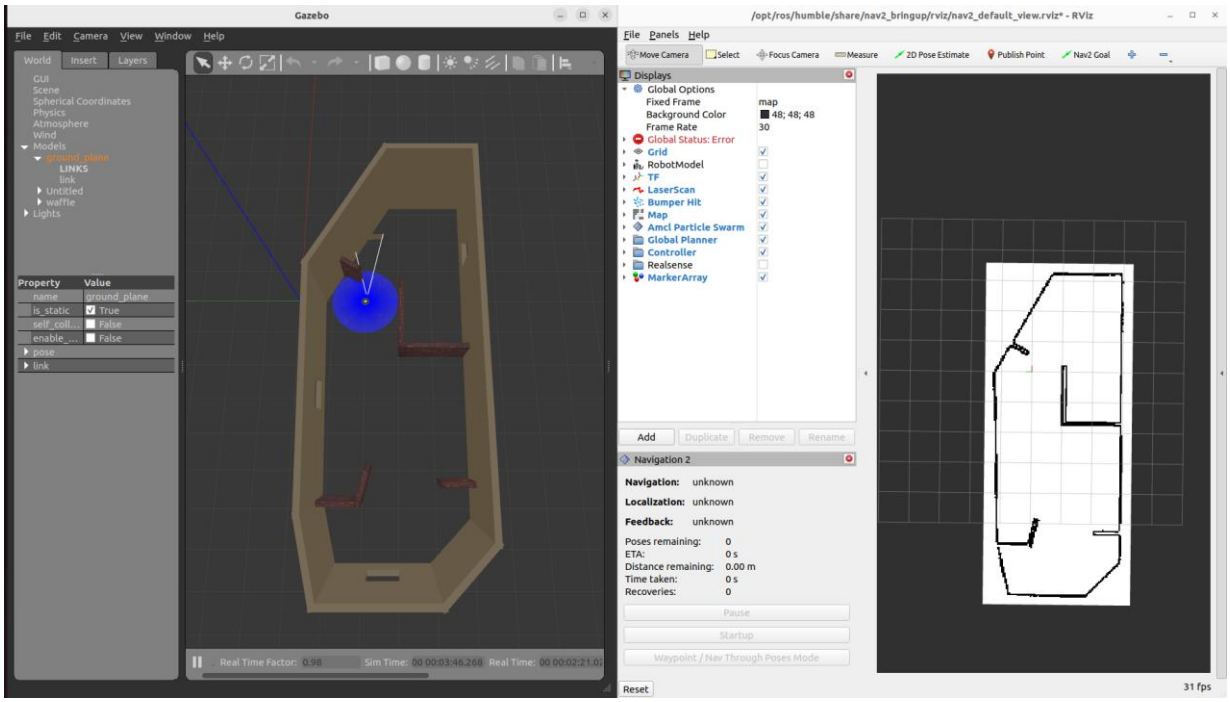


Figure IV.22 *Robot and Virtual Environment Loading (Right) and Map Localization (Left)*

Nav2 is a powerful tool within the ROS ecosystem that provides robust, flexible, and easy-to-use functionalities for autonomous navigation. It includes modules for path planning, path following, and obstacle avoidance.

With Nav2, we can provide our robot with a destination, and plan a path to that destination, navigate the robot along that path, and adjust the path as needed to avoid any obstacles encountered along the way.

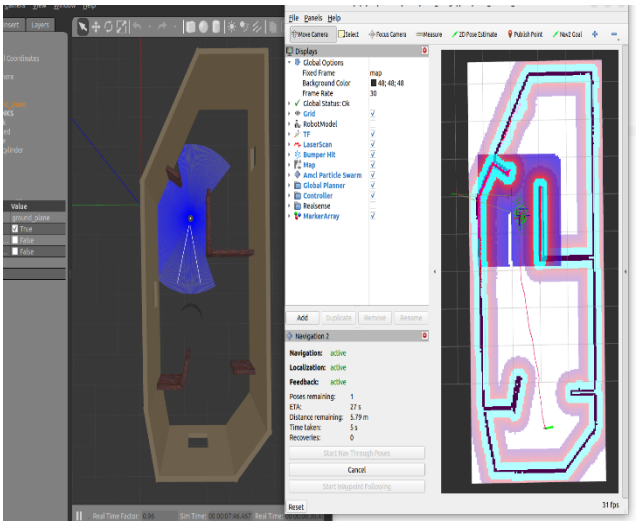


Figure IV.23 *YB425 Robot Reaching the Goal via Shortest Path (A* Algorithm)*

In the previous figure, the robot successfully reached a specific goal mission.

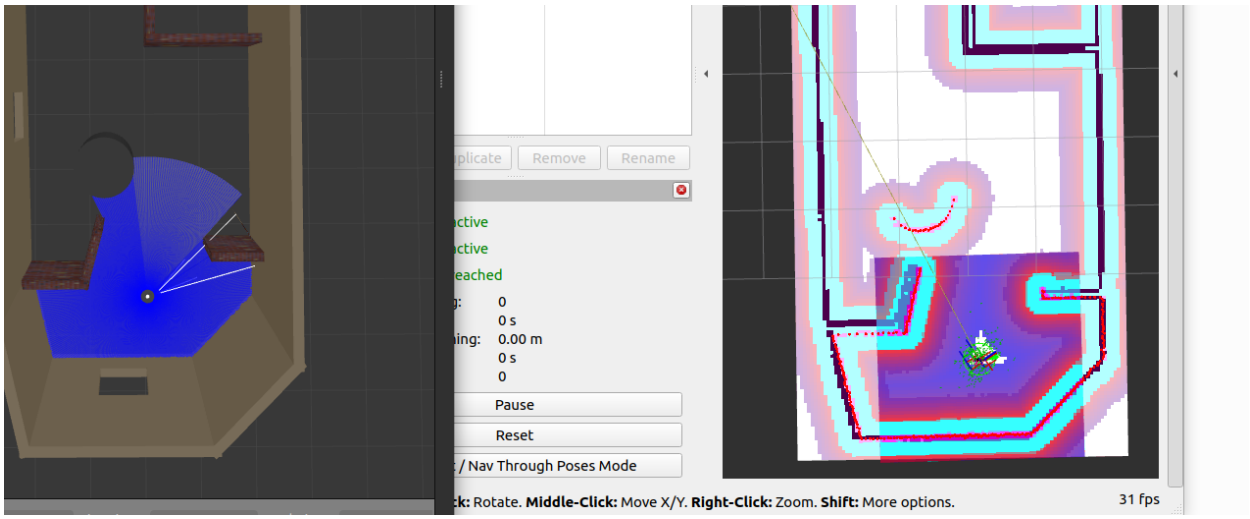


Figure IV.24 *YB425 Robot Path Planning with Obstacle Avoidance*

In our project, we used the Nav2 package to enable our robot to navigate the environment we created in Gazebo autonomously. With the map generated by SLAM Toolbox and the localization data from our robot's sensors, Nav2 successfully guided our robot from point A to point B, adjusting its path dynamically in response to any obstacles in its path.

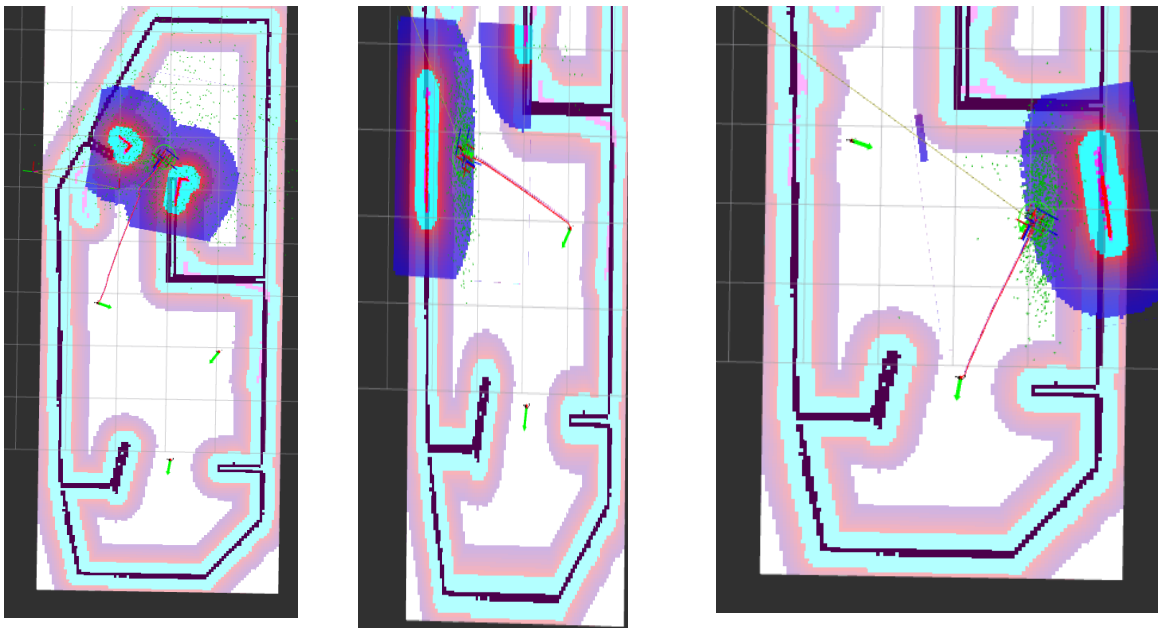


Figure IV.25 *YB425 Robot Path Planning with Multiple Goals and Dynamic Path Generation*

In the figure IV.26, the robot successfully reached multiple goal missions. This ability to navigate autonomously is a significant milestone in our robotic project, demonstrating the potential of our system to operate effectively and independently within a real-world environment.

IV.8 Conclusion

This chapter presents an in-depth walkthrough of the process of constructing an autonomous robot simulation using ROS. It details every step from the initial concept to the accomplishment of autonomous navigation. This detailed study significantly enriches the field of robotics, underlining the flexibility and potential of ROS in revolutionizing robotics research. Moreover, these simulations provide us with the ability to visualize the robot in real-world conditions. In the upcoming chapter, we will bring these concepts to life as we materialize the YB425 Robot.

Chapter V. Realization of AGV

V.1 Introduction

Upon completion of the study and simulation stages, we will dedicate our efforts to the realization phase of the AGV to validate the virtual environment. In this chapter, we transition from structural design to functional verification, closely examining the AGV's core components. The next steps include establishing the circuitry for the electrical and electronic components and testing the AGV motors, which are crucial for its operational effectiveness and reliability. Finally, we'll actualize real-world autonomous navigation, bringing our theoretical and simulated studies into tangible reality.

V.2 AGV Assembly Steps

The AGV is an intricate system that consists of four main subsystems: the mechanical, electronic, electric, and software subsystems. After the design and simulation phases, where various design tools are utilized to visualize and assess the performance of the AGV subsystems, the construction and assembly of the AGV commence. The mechanical subsystem involves the physical construction of the vehicle; taking into account aspects such as structure; the material choice; and the weight distribution. The electronic subsystem manages interfacing between different components, sensors, and controllers, all of which play critical roles in the operation of the AGV. The electric subsystem primarily concerns power management, encompassing the battery, wiring, and energy supply to various components. Lastly, the software subsystem encapsulates the programming logic and algorithms, including those necessary for autonomous navigation. This system interprets sensor data, makes decisions based on that data, and outputs commands to the actuators. The collaborative interaction of these subsystems results in a fully functional AGV capable of navigating its environment autonomously and performing its intended tasks.

V.2.1 Mechanical subsystem

The mechanical components designed: of a chassis, wheels, and engine mounts, figure V.1 represents the assembled AGV.



Figure V.1 *AGV structure*



Figure V.2 *AGV wheels*

V.2.2 Electrical Subsystem

This section outlines the electrical circuit (Figure V.3) connecting the brushless DC (BLDC) motors to the Odrive v3.6 controller, which operates at 56 volts and can manage a peak current of 120A per motor. The power supply for this setup is a 36-volt battery. Each BLDC motor incorporates three phases, designated as A, B, and C, alongside a Hall effect sensor system, which includes a 5V supply line and signal lines for the A, B, and C phases and a ground (GND). This configuration allows us to control and monitor the operation of the motors precisely.

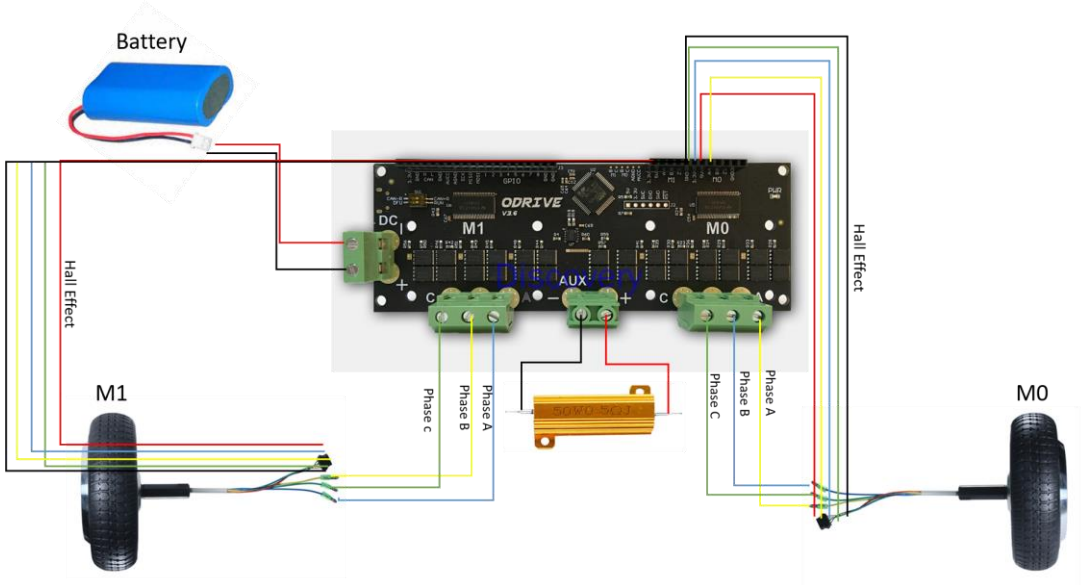


Figure V.3: *Electric circuit of motor*

V.2.3 Electronic subsystem

The various sensors and electronic components used are connected to the Odrive v3.6 (Figure V.4), Raspberry Pi board (Figure V.5), Lidar (Figure V.6)

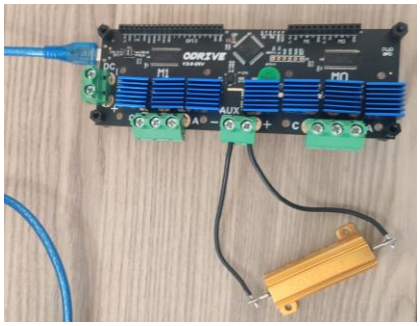


Figure V.4*:Odrive v3.6*



Figure V.5: *Raspberry Pi*



Figure V.6: *LiDAR Sensor*

To realize the electronic circuit for autonomous robot navigation, various components are connected to the Raspberry Pi, which acts as the central processing unit or microcontroller. Raspberry Pi's flexibility, computing power, and wide range of interfacing options make it an excellent choice.

Once the hardware is connected, the Raspberry Pi is configured to manage the data flow between the sensors and the Odrive. The ROS is essential in this configuration. ROS, a flexible and distributed framework for robot software development, is installed on the Raspberry Pi. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

Once ROS is installed and configured, the Raspberry Pi can take input data from the sensors, process it as required, and then output the necessary control signals to the ODrive. This allows the robot to navigate its environment autonomously, making decisions based on sensor data and executing actions via the control of its motors.

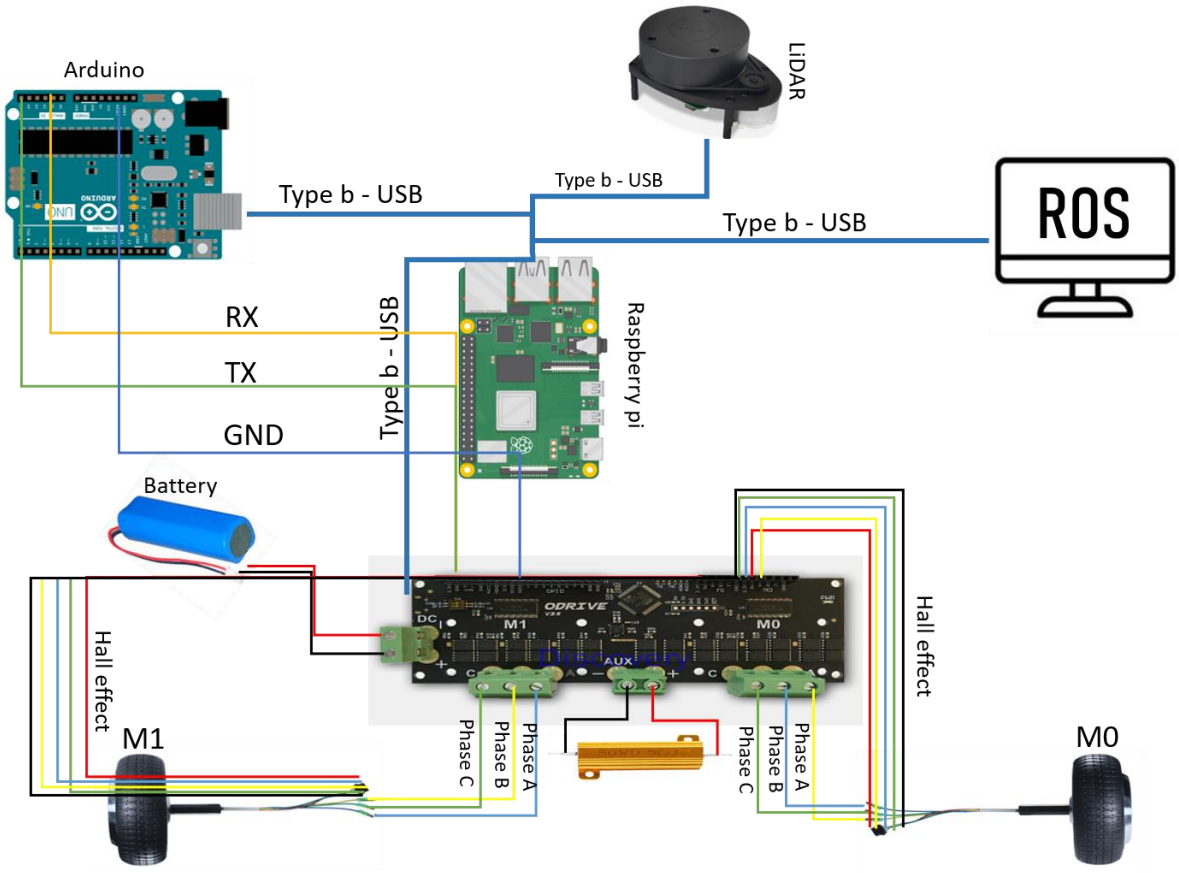


Figure V.7: *Electronic circuit*

V.3 Motor Test:

To test the efficiency of the chosen motor, we will test the following parameters:

V.3.1 Equilibration and speed motors

Upon completing the assembly of our AGV, we deem it necessary to test the motors to ensure optimal performance rigorously. We evaluate several parameters, such as speed and rotations per minute (RPM), using a microcontroller like the Raspberry Pi to supervise their operation. We incrementally increase the voltage supplied to the motors to measure the corresponding speed and RPM, establishing a clear relationship between the input voltage and motor output. We also test the direction of rotation by dispatching a command to the motors and observing their response. Suppose we encounter anomalies such as low speed, incorrect RPM, or incorrect rotation direction. In that case, we will embark on troubleshooting, which may involve reviewing the wiring, power

supply, motor controller, or the motor itself. For safety precautions, we ensure the AGV is securely stationed to prevent accidental movement during testing. Once the testing is successful, we proceed to subsequent stages like refining the motor control algorithms or integrating sensor inputs for autonomous navigation.

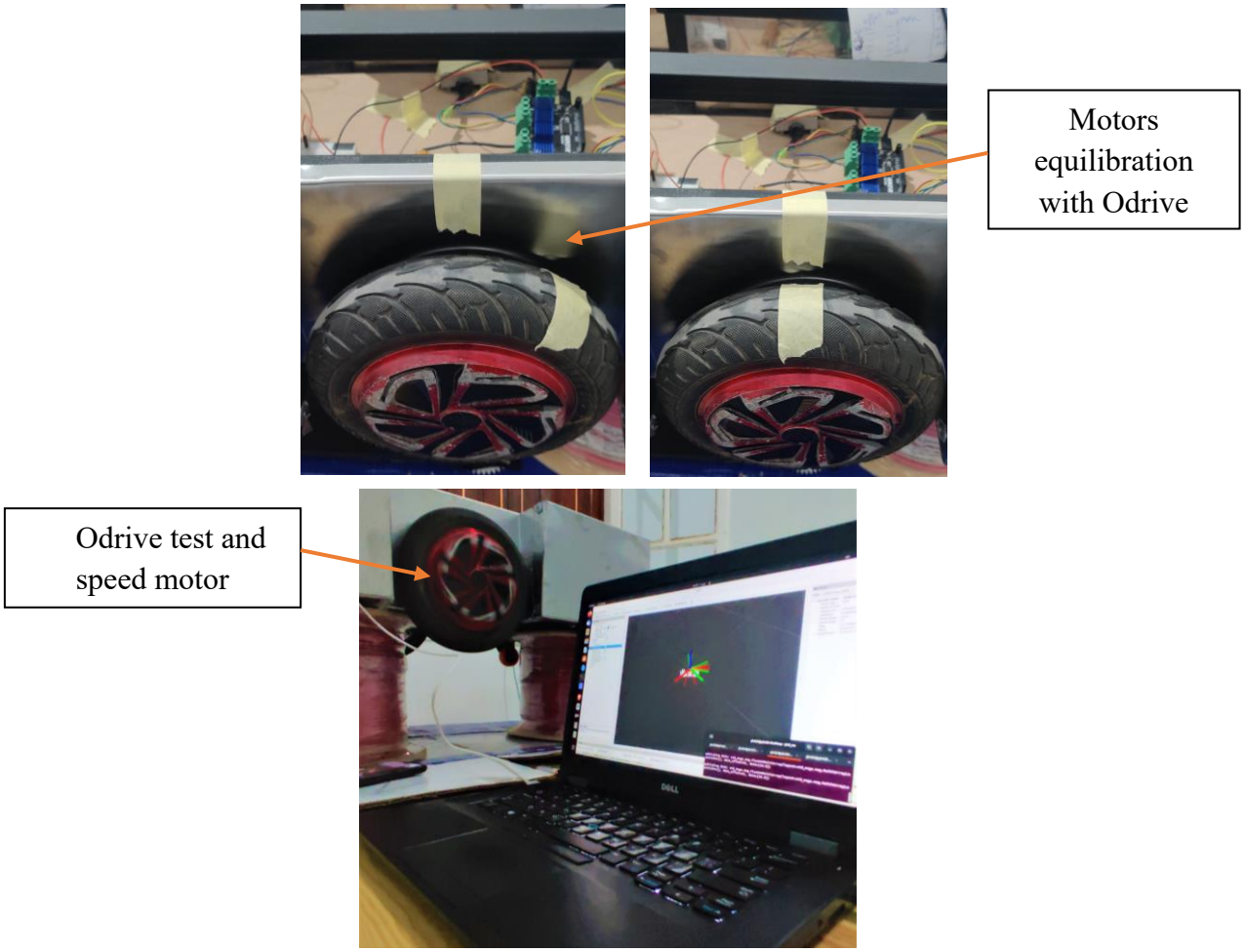


Figure V.8: *Different test of motors*

V.3.2 Motors problems:

After thorough testing, a significant issue we've identified is the pervasive noise interference between motors which causes one motor to stop. This issue becomes form encoder of M0, which influence the M1, and this issue gives an error in odrive firmware in the name of ENCODER_ERROR_ILLEGAL_HALL_STATE=0x0010 can disrupt performance and potentially affect system integrity. Addressing this problem may involve improving insulation and introducing noise-damping measures.

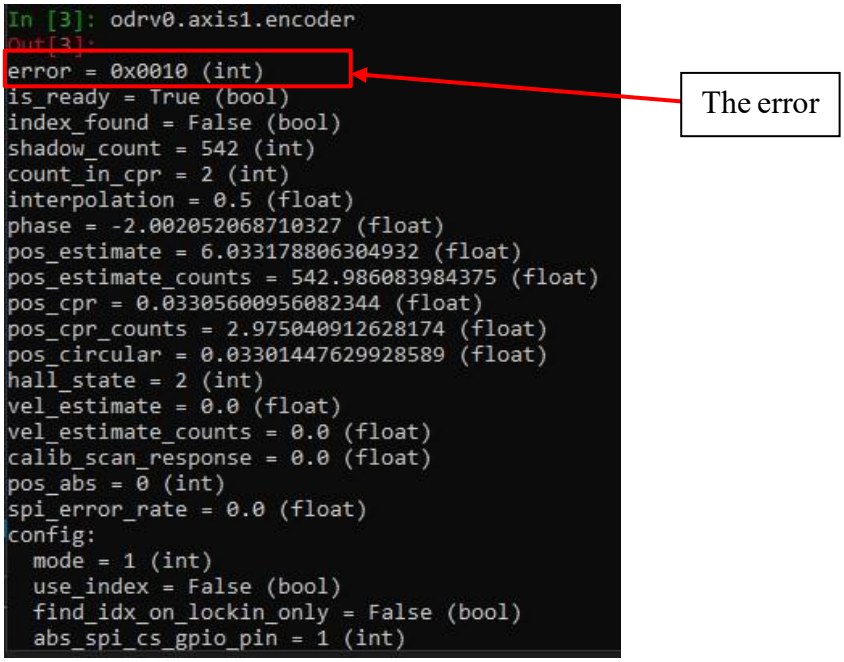


Figure V.9: *Error of noise between motor*

Solution:

After conducting comprehensive research, our solution to address the motor noise issue centers on adding a capacitor to the pin of the Hall Effect sensor in the ODrive motor controller. This alteration has been implemented on both motors, M1 and M0, and specifically, a capacitor is added between the sensor pin and the grounding (GND). By installing these capacitors, we aim to filter the high-frequency noise affecting sensor readings, thereby improving our motor system's overall stability and reliability. The chosen value of the capacitor is 47nF, a value found to be effective in noise reduction after careful experimentation and evaluation.

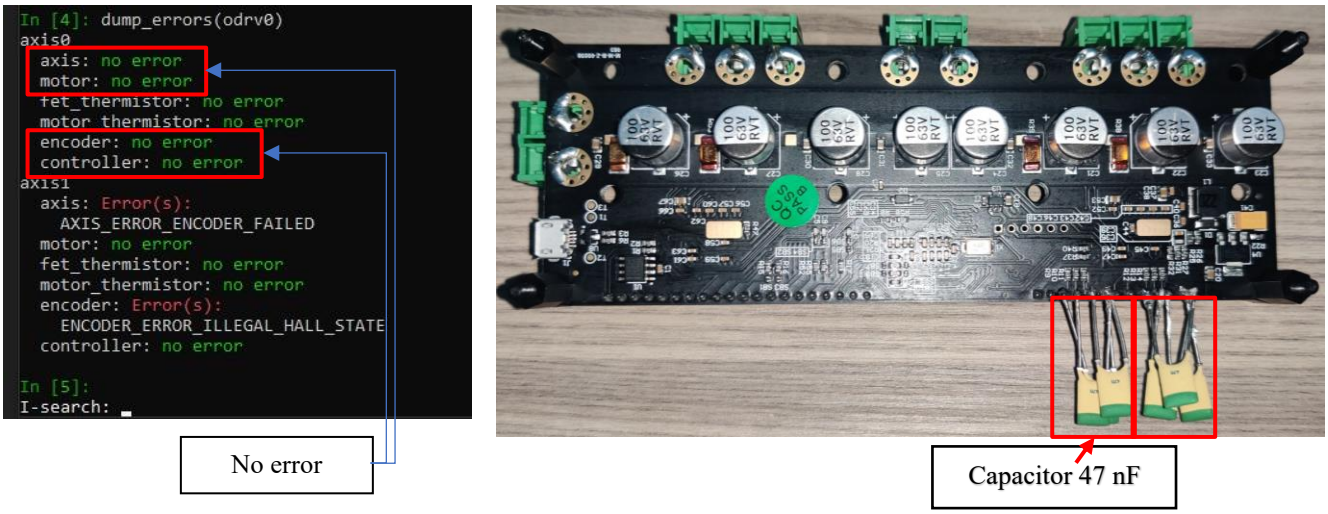


Figure V.10: *Solve problem of noise error*

V.4 Manuel control

To control our AGV, we have narrowed the options to two primary methods: a physical joystick and a mobile application. A joystick offers hands-on, immediate control, useful in scenarios requiring precision and real-time responsiveness. Meanwhile, the mobile application provides a more flexible control interface, allowing operation from a distance and possibly integrating additional features like path planning or real-time diagnostic information.

V.4.1 Joystick control

Utilizing a joystick for mobile robot control provides direct, tactile feedback, which can be crucial for tasks requiring fine motor control and precision. This control method transforms the user's hand movements into robot motion, offering an intuitive, responsive interface. Moreover, a joystick can support multiple degrees of freedom, allowing more complex maneuvering of the robot, such as forward/backward motion, turning, and even more sophisticated operations depending on the robot's capabilities.

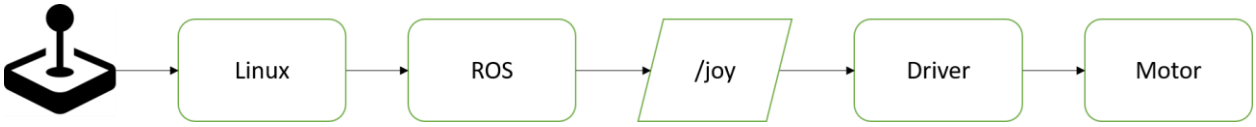


Figure V.11: *Joystick Manuel control diagram*

V.4.2 Application control

The app, which can be accessed from any smartphone, allows users to operate the robot remotely by Wi-Fi connection, expanding the scope of control. Beyond simple movement commands, the mobile app can provide a host of additional features such as real-time status updates, diagnostics, pre-programmed maneuvers, and even sophisticated path-planning capabilities, thereby enhancing the overall functionality of the robot.

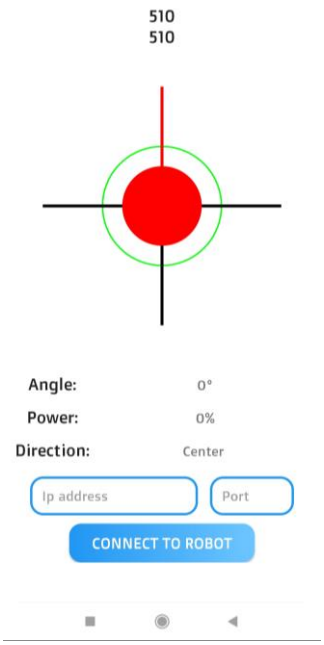


Figure V.12: *App mobile interface*

V.5 Visualization

RViz is a 3D visualization tool within the ROS environment, offering an intuitive interface for visualizing a variety of sensor data and robotic states. It supports real-time and recorded data, displaying it in a dynamic, navigable 3D view that can aid in debugging and system comprehension.

V.5.1 Lidar visualization

RViz a 3D visualization tool for ROS (Robot Operating System), offers powerful capabilities for visualizing LiDAR data. It displays real-time or recorded point cloud data captured by a LiDAR sensor, facilitating the interpretation of complex spatial information. LiDAR visualization involves the creation of 3D representations of the environment using (LiDAR) data.

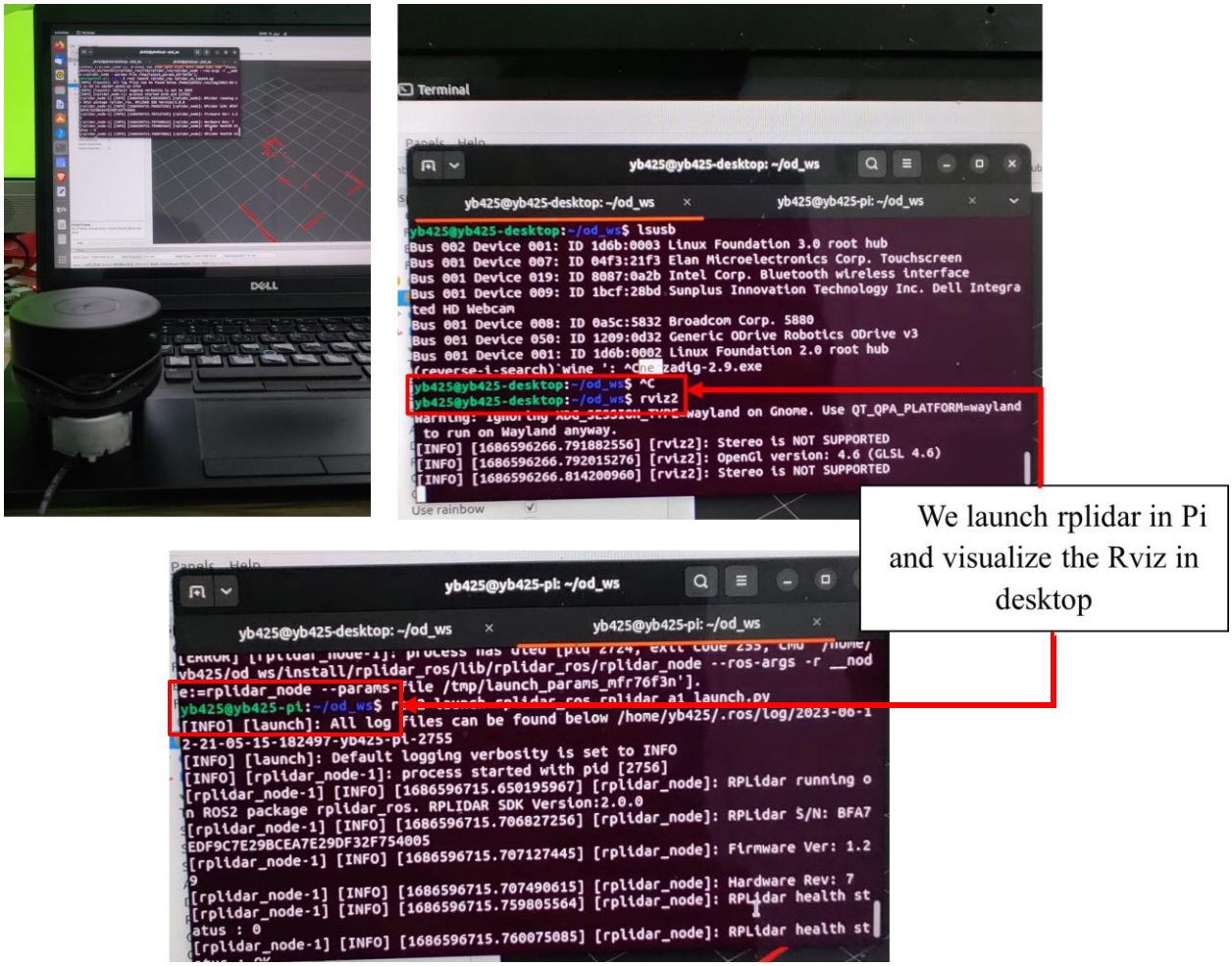


***Figure V.13**: Visualization of LiDAR in RViz*

V.5.2 Camera visualization

Camera visualization in ROS (Robot Operating System) involves using tools such as RViz and image-view to interpret and display image data from a robot's cameras. These tools provide live streaming or playback functionalities for effectively analyzing and debugging your robot's perception.

On the other hand, image-view is a simple image viewer for ROS. It subscribes to an image topic and displays the image content in a window. This tool is particularly useful for checking the raw output of a camera node.

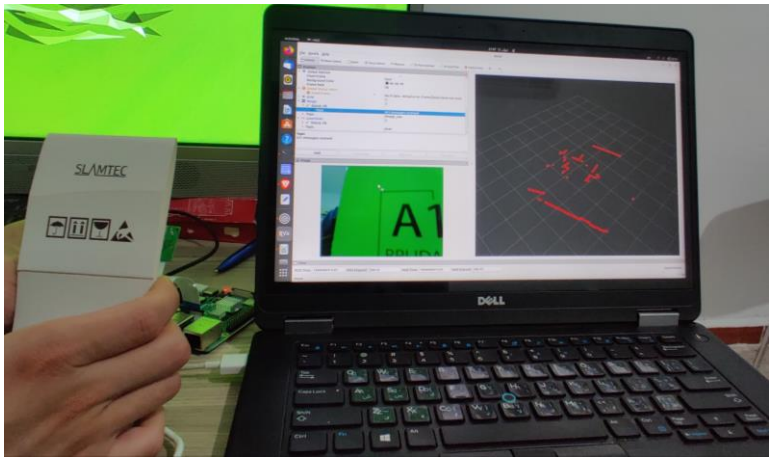


Figure V.14: *visualization camera in RViz*

V.6 Real Robot Test

We test the YB-425 robot with Manuel control and camera visualization for a real time

V.6.1 Manuel control:

In this section, we preview the implementation of manual control for our real AGV using a joystick within a ROS environment. The joystick provides an intuitive, user-friendly interface for AGV control, while its integration into the ROS framework presents both unique challenges and learning opportunities.

The AGV's manual control is facilitated via the joystick, which sends commands to the ROS node responsible for handling AGV motion. This node deciphers the joystick's input into motor commands that are understood and executed by the ODrive motor controller governing the AGV's movements. The joystick allows for a range of controls, including initiating forward and backward movements, enabling turns, and commanding stops. the figure V.15 show the control of our real robot with joystick



Figure V. 15 *controlling real robot with joystick*

V.6.2 Camera visualization in the robot:

In this part of our project, we implemented real-time video streaming for the AGV using a camera mounted on its top section. This camera serves two crucial purposes.

Firstly, it enables the visualization of the AGV's front-side environment in real-time, allowing operators to see what the robot sees, which can be particularly beneficial in remote control or supervision scenarios. Secondly, the camera assists in determining the robot's proximity to obstacles. By interpreting the visual data captured by the camera, we can ascertain the robot's distance from objects and obstacles in its path. The camera, integrated into the ROS environment, streams video data to a dedicated ROS node. This node processes the video feed, enabling real-time visualization, and performs simple object detection tasks to identify potential obstacles. Based on the size and position of detected objects in the camera's field of view, we estimate the robot's distance from these obstacles. The figure V.16 show the video streamed from the camera of robot.



Figure V.16 *the video streamed from camera of robot via mobile phone*

You can also access to the robot camera via multiple device



Figure V.17 *multiple device access to camera*

V.7 Conclusion

This chapter began with the task of putting together various components and setting up both electrical and mechanical subsystems. Building the circuits was a challenging task, but with every step, our understanding of the AGV's operations expanded and our problem-solving skills grew.

Despite facing many obstacles, we reached an important stage: controlling the robot using a joystick while simultaneously visualizing camera and LiDAR feeds. This development brought us closer to our main goal of autonomous navigation and gave us hands-on control and direct sensor input from the AGV.

As we conclude this chapter, we not only reflect on what we have achieved, but also look forward to the exciting potential of fully autonomous AGV navigation. Our exploration into the world of robotics continues, informed and enriched by the journey we have undertaken so far.

GENERAL CONCLUSION

In this work, we immersed ourselves in the intricate realm of robotics, focusing our efforts on the development and optimization of Automated Guided Vehicles (AGVs). The aim was to create a highly efficient AGV that could swiftly adapt to environmental changes and find the most efficient route to its destination. Our approach involved the utilization of three innovative technologies: The Robot Operating System (ROS), Simultaneous Localization and Mapping (SLAM), and the A* pathfinding algorithm.

Our journey comprised both triumphs and challenges. During the simulation phase, we successfully implemented auto-navigation and map-building features, marking a considerable advancement towards an autonomous and adaptable AGV. However, during the realization phase, we faced the inherent complexities of implementing ROS on real hardware and experienced resource limitations, which necessitated a scaled-down approach.

Despite the hurdles, we made substantial progress. We were able to incorporate manual controls, integrate camera vision and LiDAR technology for object detection and distance measurement, and establish joystick control, allowing us to navigate closer to our envisioned AGV. One perspective we have introduced is the use of QR code readers and camera vision to further enhance navigation. This step, coupled with our work on LiDAR, would make our AGV equipped for both indoor and outdoor operations.

Although we didn’t fully realize the autonomous navigation and mapping functionalities, our work shed light on the challenges associated with these technologies. We not only enhanced our understanding but also identified potential solutions to these challenges, offering a solid foundation for future work in this domain.

The journey has given us an AGV marked by intelligence, adaptability, and swift movement. While there are areas that warrant further improvement, the journey doesn’t end here. As technologies evolve, particularly ROS, SLAM, and A*, we look forward to continuing to refine our AGV, pushing its performance boundaries even further. We remain committed to the principles of Industry 4.0, with an enduring aim to create a system that can adapt, learn, and efficiently navigate the dynamic world it operates in.

References

[1]. Robot in the operating Room the history. Douglas R. Ewing, MD. s.l. : seminaires in laparoscopic surgery, 2004, Vols. 63-71.

[2]. Handbook o robotics. Siciliano, B, & Khatib. 2016, springer.

[3]. Embedded system design . Marwedel, peter. 2018.

[4]. Malinowski, A., & Yu, H. (2011). Comparison of Embedded System Design for Industrial Applications. IEEE Transactions on Industrial Informatics, 7(2), 244–254.

[5]. Handbook of Unmanned Aerial vehicles . kimon P.valavanis, George J vachtsevanos. s.l. : springer .

[6]. handbook of automation . shimon Y, Nof. s.l. : springer, 2009.

[7]. Asurvery in the design and control of automated guided vehicle system. orha, alexandru Petru Vancea and ioan. s.l. : carpathian journal o electronic and computer engineering, 2019.

[8]. Art-slam: Accurate real-time 6d of lidar slam . Matteucci, Matteo Frosi and Matteo. s.l. : IEEE robotics and Automation letters , 2022.

[9]. Guided motion planning In Robotic Research. Alterovitz, Gu Ye And Ron. s.l. : springer, 2017.

[10]. Inertial sensors technologies for navigation applications : State of the art and future trends . Youssef, Naser EL-sheimy and ahmed. s.l. : satellite Navigation , 2020.

[11]. Mobile Robot localization using GPS. Guo-Sheng Cai, Huei-Yung Lin,and Shih-Fenkao. 2019.

[12]. Boyu Gao, Haoxiang Lang,and Jing Ren Stereo visual slam for autonomous vehicles. s.l. : IEEE , 2020.

[13]. present and future of simultaneous localization and mapping . Cesar Cadena, Luca Carlone. Henry Carrillo, Yasir Latif, Davide Scaramuzza ... s.l. : IEEE, 2016.

[14]. A survery of motion planning algorithm from the prespective of autonomous UAV guidance . chad Goerzen Zhaodan Kong, and Bernard Mettler. s.l. : journal of intelligence and Robotic Systems , 2010.

[15]. Wavefront and A-star Algorithms for mobile robot path planning . Zidane, Issa & Ibrahim, Khalil. 2018 .

[16]. Quigly, M, Conley,K , Gerkey, B , Faust, J .... ROS : an open-source Robot operating system. 2009.

[17]. introduction to ROS. ROS

[18]. concepts. ROS.org

[19]. Isermann, Rolf. Mechatronic System. london : Springer London, 2003. 1.

Annex

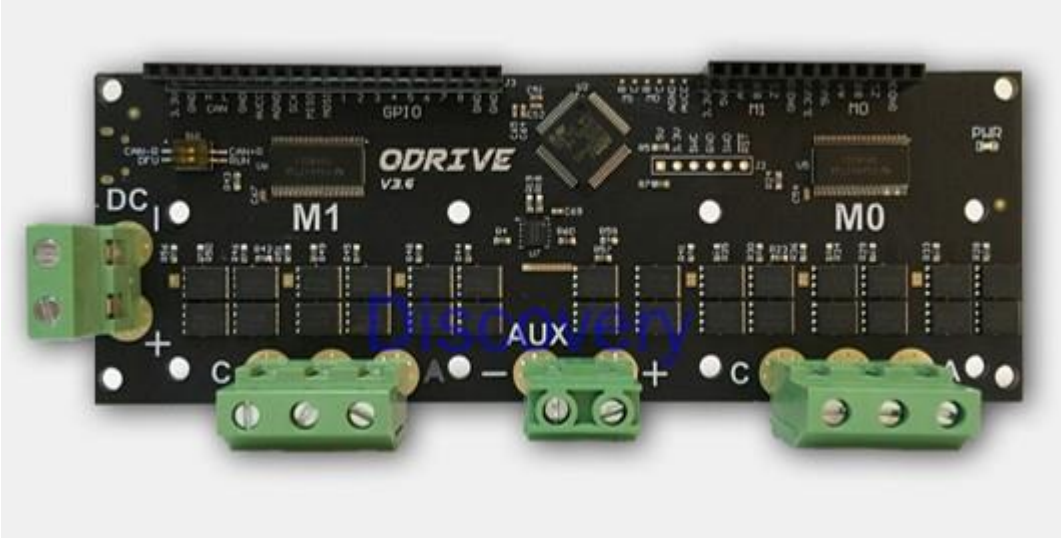
Annex A: Fusion 360

Fusion 360 is a cloud-based 3D modeling software developed by Autodesk. It's a CAD/CAM/CAE (Computer-Aided Design, Manufacturing, and Engineering) tool that combines multiple functions in one software package. It has capabilities for :

- **3D Modeling:** This is the central feature of Fusion 360, which allows users to create and edit 3D models of products or components. This includes creating solid bodies, modifying them with tools like extrusion, loft, sweep, and more, and assembling them into more complex structures.
- **2D Drawing:** Fusion 360 also enables creating 2D drawings for manufacturing, including dimensioned drawings and exploded views.
- **Simulation:** You can simulate the real-world performance of your design using stress analysis, thermal analysis, and other tools.
- **CAM (Computer Aided Manufacturing):** Fusion 360 includes CAM tools so you can plan, simulate, and generate tool paths for a wide variety of manufacturing processes, including milling, lathing, and more.
- **Collaboration:** As a cloud-based software, Fusion 360 has strong collaboration features. Multiple people can work on a design at the same time, and changes are synced to the cloud.
- **Rendering:** Fusion 360 includes photorealistic rendering capabilities. You can apply materials and lighting to your models and render them as still images or animations to show off your design.
- **Generative Design:** This is a relatively new feature in Fusion 360. It uses artificial intelligence to generate design options based on your goals and constraints.
- **Electronics:** Fusion 360 now includes the ability to design PCBs (Printed Circuit Boards), allowing for an integrated design experience.

Annex B: Odrive v3.6

- **High Performance**: Odrive is designed to deliver high amounts of torque to electric motors quickly and efficiently. This makes it suitable for demanding applications like robotics, electric vehicles, and CNC machines.
- **Open Source:** The schematics and firmware for Odrive are open-source. This means that anyone can look at, modify, and improve upon the controller's design and software.
- **Brushless Motor Support:** Odrive supports brushless DC (BLDC) motors, which are more efficient, quieter, and longer-lasting than brushed DC motors.
- **Field Oriented Control (FOC):** This is a method of motor control that optimizes torque output and power efficiency. It's particularly useful for applications where precision is important.
- **Position, Velocity, and Torque Control:** Odrive can control motor position, velocity, and torque separately or in combination. This provides a lot of flexibility in terms of how the motor can be used.
- **Encoder Support:** Odrive supports various types of encoders, which are sensors that measure the position or speed of the motor's shaft.
- **Regenerative Braking:** This feature allows the Odrive to convert the kinetic energy of a decelerating motor back into electrical energy, improving overall efficiency.



Annex C: URDF File

The Unified Robot Description Format (URDF) file is a key component in the robot simulation process. It is an XML file format used in ROS (Robot Operating System) to describe all elements of a robot. Here's a basic outline of what a URDF file represents:

Structure of a Robot: The URDF file represents the robot's physical layout. It describes the links (rigid bodies such as robotic arms, wheels, chassis) and joints (pivot points like elbows, wrists, or wheels' connections to the chassis) that make up the robot's structure.

Visual Aspects: It also depicts the visual characteristics of a robot, including its shape, color, and size.

Physical Properties: The URDF file outlines the physical properties of the robot, such as mass, inertia, and friction characteristics.

Sensor and Actuator Descriptions: The file can also incorporate descriptions of sensors and actuators attached to the robot.

The URDF file is crucial as it provides a complete description of a robot that is used by the robot state publisher to publish the state of the robot to tf, which is used by many higher-level tools such as RViz, motion planners, etc.

In ROS, you can find the URDF file usually in a dedicated "URDF" directory within the robot's package directory. To use a URDF file in your simulation, you would usually load it at runtime when launching your robot's nodes, often using a ROS parameter.

Annex D: RP LiDAR

RP LiDAR, manufactured by Slamtec, is a type of Light Detection and Ranging (LiDAR) sensor often used in robotics for its ability to map environments and perform localization tasks. It emits laser beams and then measures the time it takes for the light to return to calculate distances to surrounding objects, creating a 2D or 3D representation of the environment.

When using an RP LiDAR sensor with ROS, you will often interact with two types of files:

Driver: The driver is a piece of software that enables ROS to communicate with the LiDAR sensor. In the case of RP LiDAR, the commonly used driver is the **rplidar_ros** package, which you can find on GitHub or through ROS's package manager, **apt**.

Sensor Data Files (Bag Files): When you use a LiDAR sensor with ROS, the sensor's readings are typically saved in a .bag file. This is a ROS-specific format for recording and playing back ROS message data. You can record sensor data by running the **rosbag record** command while your sensor is active, and you can replay it with **rosbag play**.

