# MAZE NAVIGATION AND PATH OPTIMIZATION FOR AN AUTONOMOUS ROBOT IN A SIMULATED ENVIRONMENT

**BY**

**KAMREN JAMES AND MIKAYLA LEWIS**

**DESIGN PROJECT REPORT**

**SUBMITTED TO THE FACULTY**

**OF THE**

**COLLEGE OF ENGINEERING**

**IN**

**PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DEGREE OF**

**BACHELOR OF SCIENCE**

**IN**

**MECHANICAL AND MANUFACTURING ENGINEERING**

**MAY 2024**

**COLLEGE OF ENGINEERING**
**TENNESSEE STATE UNIVERSITY**
**NASHVILLE, TENNESSEE**

**MAZE NAVIGATION AND PATH OPTIMIZATION FOR AN AUTONOMOUS ROBOT IN A SIMULATED ENVIRONMENT**

**DESIGN PROJECT REPORT**

**Approval Recommended:**

_____

**Project Advisor**

_____

**Date**

_____

**Course Instructor**

_____

**Date**

_____

**Department Head**

_____

**Date**

**Approved:**

_____

**Dean, College of Engineering**

_____

**Date**

## ABSTRACT

## MAZE NAVIGATION AND PATH OPTIMIZATION FOR AN AUTONOMOUS ROBOT IN A SIMULATED ENVIRONMENT

K. James & M. Lewis, Senior Mechanical and Manufacturing Engineering

The field of autonomous mapping and path planning for robots has gained attention in recent years due to the increasing demand for efficient and reliable autonomous systems. This paper will explore the existing efforts made to address the challenges associated with autonomous mapping and path planning. It discusses this project's significance in various industries and domains, such as disaster response, transportation, logistics, and robotics fields. The paper will also trace the historical development of autonomous mapping and path planning, highlighting the key milestones and technological advancements that have shaped the field. It also examines the factors contributing to the problem, including static/dynamic environments, limited sensor capabilities, and computational constraints. Furthermore, the paper identifies the necessary tools and technologies, such as advanced sensors, algorithms, machine learning, real-time data processing and software required to overcome these challenges. By addressing these challenges and advancing the capabilities of autonomous mapping and path planning, this project aims to unlock the full potential of autonomous systems to revolutionize various industries.

## DEDICATION

To my Mother Helen R. James & my Father Stephen L. James, through there endless support both financially and spiritually which allowed for me to get this far with my education. Here I dedicate this project to you two.

K.J.

To my Mother and Father Courtney Lewis and Michael Lewis for their continuing confidence in my abilities and the continuous faith they instilled in me to get me through my time at TSU. To my Grandmother who I will always cherish and look forward to seeing again one day, and to everyone who had a hand in making sure that I could enjoy my time in school, and to My sisters who are the reason I push myself so hard, I dedicate this project to you.

M.L.

**ACKNOWLEDGEMENT**

This project was written under the direction and supervision of Dr. Mohammad Habibi. We would like to express our sincere appreciation and gratitude to him for taking interest in our project and aiding and guiding us.

K.J, M.L

# TABLE OF CONTENTS

LIST OF FIGURES

FIGURE DESCRIPTION PAGE

## LIST OF TABLES

# NOMENCLATURE

ROS - Robot Operating System

VM - Virtual Machine

SLAM - Simultaneous Localization and Mapping

PRM - Probabilistic RoadMap

TD3 - Twin-Delayed Deep Deterministic Policy Gradient

DDPG - Deep Deterministic Policy Gradient

PPO - Proximal Policy Optimization

SAC - Soft Actor-Critic

RVIZ - Robot Visualization Description Format

URDF - Universal Robotic Description

SDF - Simulation Description Format

TF - Transform

XACRO - XML macro language

RQT - ROS Tool (Used to see active node graph)

ML- Machine Learning

# CHAPTER 1

## INTRODUCTION

Note Pages

1.Autonomous Robots and Systems

- A. Problem Identification
- B. Current Solutions
- C. Significance
- D. History
- E. Historical Perspectives
- F. Our efforts
- G. Tools Required
- H. Our contribution

# Chapter 1

## Introduction

In recent years, autonomous systems have witnessed significant growth and adoption throughout various industries. With the rise of this technology, it has increased the operating efficiency of some industries and is actively expanding to others. However, during this period of expansion, robots of this type need to have the capability to autonomously map, and path find their way throughout their environment. The capability of these systems to autonomously map their environment and plan the optimal paths is crucial to their employment. Given they can do this more effectively their reach in the industry/field and become limitless. This capstone research paper focuses on the development of machine learning algorithms and systems for autonomous mapping and path planning, aiming to improve efficiency, safety, and automation in diverse applications.

The importance of this research area lies in its potential to revolutionize industries such as transportation, logistics, agriculture and other fields such as disaster rescue and surveillance. Accurate mapping and path planning algorithms allow autonomous systems to optimize routes while avoiding obstacles, leading to increased productivity, reduced cost, and improved safety standards.

### *Section 1: Autonomous Robots and systems*

A. Problem Identification

The problem for this capstone project is to create a prototype of a robot that can autonomously map an environment, a maze in this case, then use machine learning to find the most optimal path to the destination within that environment. Additionally, there is the task of developing and training machine learning models to enable the robot to autonomously navigate the maze and optimize its path. Moreover, ensuring seamless communication between the software components and the physical robot poses another

significant challenge, particularly in terms of implementing wireless communication protocols. Thus, the overarching problem revolves around creating a cohesive and functional system that can effectively explore and navigate maze environments autonomously, leveraging machine learning techniques while overcoming various technical and logistical hurdles.

B. Current Solutions

The problem created by society today is the lack of efficient transportation and delivery methods. In China they have implemented AI (Artificial Intelligence) into delivery systems to help with efficiency. Some current efforts to solve this problem are the exploration and utilization of other programs and algorithms. Such methods include:

- SLAM (Simultaneous Localization and Mapping)
- Graph based approaches (Probabilistic roadmap, Dijkstra, A*, D* algorithms)
- Machine learning methods (Reinforcement, Supervised and Unsupervised learning)
- Sensor Fusion
- Genetic algorithms & Particle swarm optimization (RRT (Rapidly exploring and random tree)/A*)
- Collaborative methods (Communication between multiple systems)
- Deep Learning (TD3, DDPG, PPO)

Beginning with Simultaneous Localization and Mapping (SLAM), this technique has a crucial role in addressing the mapping portion of this problem. By using sensor data and advanced algorithms, SLAM allows the system to create accurate maps of its present environment in real-time. And another benefit of this method is that as well as creating maps based off the data while traversing the system is aware of where it is in space. This capability allows the system to be able to understand and navigate its surroundings effectively. [22]

Graph based approaches contribute to the path planning aspect of the problem. These methods help provide a structured representation of the environment and allow for efficient and optimized path planning. Graph based approaches include methods such as PRM (Probabilistic Roadmap) and Dijkstra's algorithms. These programs/algorithms variable where you can influence the path the system will chose to take based on manipulating the data. By using this method, autonomous systems can determine the most optimal paths based on various constraints and objectives. [25]

Machine learning and Deep learning methods assist the robot with making informed decisions in real time. Machine learning, algorithms can help improve the mapping accuracy and efficiency, while deep learning can assist in recognizing and avoiding obstacles during the path planning navigation.[16]

Sensor fusion involves combining the data from multiple sections to get a more comprehensive view of the current environment. The integration of sensor data enhances the accuracy of the map, localization, and obstacle detection leading to more effective path planning. [26]

Genetic algorithms and Particle swarm optimizations allow for the optimization of the path planning algorithm in place. By using these optimized algorithms, the system can search for the most efficient paths based on different objectives to make the best decision. Those objectives might include paths with the least number of turns, the path with the shortest distance traveled, the shorted calculated time of completion or the path that uses the least amount of energy. These types of algorithms also assist autonomous systems with dynamic and static environments. [5]

Collaborative methods can enhance the accuracy and efficiency of mapping and path planning. By utilizing multiple autonomous systems and coordinating data and individual actions. It is possible to gain fully accurate maps in both static and dynamic environments. Leading to more optimal path creation. [5][25]

C. Significance

The current contributions to creating an autonomous system involve addressing the challenges related to dynamic/static environments, sensor limitations, computational constraints, and public safety concerns. Researchers are focusing on developing machine learning algorithms and systems for autonomous mapping and path planning to improve efficiency, safety, and automation in various applications. Efforts to solve this problem include the exploration and utilization of various programs and algorithms such as SLAM, graph-based approaches, machine learning methods, sensor fusion, and collaborative methods. The significance of this research lies in its potential to revolutionize industries such as transportation, coordination, agriculture, disaster rescue, and surveillance by enabling autonomous systems to optimize routes, avoid obstacles, and increase productivity. The history of autonomous vehicles and robots' dates to the concept of self-propelled carts designed by Leonardo da Vinci and has evolved significantly with the advancement of machine learning technologies. [24] The reasons for the problem include dynamic environments, sensor limitations, computational constraints, and public safety concerns. [15] To contribute to this project, a background in Calculus, Circuits, Mechatronics, Computer-aided Design (CAD), and Python/C++ Programming is required. The project aims to develop sophisticated machine learning algorithms that enable precise autonomous mapping and efficient path planning, with a focus on addressing the inherent challenges and paving the way for broader adoption and integration of autonomous technologies in real-world applications.[12]

With our age of technological advancement, autonomous systems are needed to aid in the advancement of efficiency, safety, cost reduction and future scalability. With the advancement of autonomous robots that can path plan their own routes to get to their destination efficiently.

Optimal path planning and mapping are critical components in the successful implementation of autonomous systems. By automating the mapping process and generating optimal paths, these systems can navigate complex environments, reduce travel time, and enhance safety.

In the transportation industry, autonomous vehicles have the potential to transform the way we travel. Autonomous mapping and path planning enable these vehicles to navigate unfamiliar routes and dynamically adapt to changing road conditions. By minimizing travel time and optimizing routes, autonomous vehicles can enhance transportation efficiency, reduce congestion, and improve overall road safety. [5][6] Additionally, in organization and delivery services, efficient mapping and path planning can optimize delivery routes, reduce costs, and improve customer satisfaction by ensuring timely and reliable deliveries. [5][21] Furthermore, autonomous mapping and path planning are crucial in logistical warehouse operation. With the implementation of autonomous robots, they will lead to increased productivity, improved workflow and reduce operation costs. Moreover, as the demand for autonomous systems continues to grow, addressing the challenges in mapping and path planning becomes increasingly important. The ability to accurately map and plan paths in dynamic environments is essential for the safe and reliable operation of autonomous systems. By overcoming these challenges, this project opens doors to wider adoption and acceptance of autonomous systems in the transportation, organization, and robotic industries.

Figure 1.1: DaVinci's Cart

D. History

Integrating the rich history of autonomous vehicle development into the broader narrative of autonomous robots and vehicles reveals a fascinating journey marked by innovation, technological advancements, and the growing impact of artificial intelligence on various sectors. This extended narrative encapsulates the evolution from early visions to current implementations and future possibilities of autonomous systems.

The concept of automating transportation and tasks dates back centuries, notably to Leonardo da Vinci's envisioning of a self-propelled cart in the Renaissance, which

could follow pre-set paths. This early invention laid the groundwork for the exploration of automation in vehicles. The 20th century brought the idea closer to reality; the 1950s saw General Motors conceptualize the Firebird, equipped with an "electronic brain" for automating driving tasks, illustrating the burgeoning interest in vehicle autonomy. Research efforts in the 1970s and 1980s, particularly at academic institutions like Stanford University with the "Stanford Cart," pushed the boundaries of what was technologically possible, employing early computer vision techniques for autonomous navigation. The early 2000s saw a pivotal moment with the DARPA Grand Challenges, which significantly propelled forward the field of autonomous vehicles by demonstrating their potential in navigating complex environments without human intervention [27].

The rapid advancements in machine learning and artificial intelligence over the last two decades have been instrumental in the development of autonomous systems. From algorithms for predictive analysis to sophisticated models enabled by deep learning, AI has become the cornerstone of autonomous vehicle development, facilitating real-time decision-making and interaction with the environment.

Beyond the realm of personal and commercial transportation, autonomous technology has found applications in agriculture, logistics, healthcare, and environmental monitoring. Drones and autonomous tractors are revolutionizing crop management, while autonomous delivery robots streamline logistics operations. In healthcare, robots are tasked with disinfection, medication delivery, and surgical assistance, showcasing the versatility of autonomous systems in improving efficiency and care quality.

The narrative from da Vinci's cart to today's self-driving cars and beyond is a testament to the cumulative advancements in engineering, computer science, and AI. The DARPA Grand Challenges, along with investments from tech giants and automakers in the 2010s, have underscored the feasibility and potential of autonomous vehicles in transforming everyday life. The ongoing development of GPUs, cloud computing, and

sophisticated AI models continues to expand the capabilities and applications of autonomous systems.

As we look to the future, the integration of advanced machine learning techniques and computing resources promises to further elevate the autonomy and efficiency of robots and vehicles. This progression not only exemplifies a journey of technological innovation but also highlights the symbiotic relationship between advancements in AI and the practical implementation of autonomous systems across various domains. The evolution of autonomous vehicles and robots underscores a significant shift towards integrating these technologies into the fabric of daily life, with the potential to revolutionize transportation, agriculture, healthcare, and environmental monitoring. This journey from conceptual visions to practical applications and future possibilities marks an era of unprecedented innovation and transformation in autonomous systems.

Figure 1.2: DARPA Grand Challenge Vehicle

## E. Historical Perspectives

*Table 1.1: Initial Survey*

| Aspect | Past Advancements | Present Advancements | Future Advancements |
|---|---|---|---|
| Public Perception | -Little faith in what it could be<br>- Many Concerns on what it would turn into | -Increasing Acceptance<br>-More Trust in its viability | -Full acceptance and integration into society<br>-Norm for transportation and everyday needs |
| Safety | -Concerns about reliability<br>-Concerns about public safety | -Improvements in safety protocols | -Enhanced safety standards<br>-near-zero Accident and Incident rates |

| | | | |
|---|---|---|---|
| | | - Increased improvement in incident prevention | |
| Technology | -Basic Sensors<br>-Rudimentary Algorithms | -Advanced Sensors<br>-AI algorithms | -AI driven systems<br>-Enhanced capabilities and safety features |
| Economic Impact | -little to no impact<br>-Early stages of research and development | -Growing Market<br>-Potential for significant benefits for society | -Leading provider for transportation<br>- Substantial economic Implications |

The inception of autonomous technologies was met with reservation, as public trust was limited, reflecting a general uncertainty about the potential and eventual role of these systems within society. Research has indicated a gradual shift in this stance, with current trends showing a growing acceptance and a burgeoning confidence in their practicality. Scholarly projections suggest a future where autonomous technologies are seamlessly woven into the fabric of daily life, serving as standard tools for transportation and routine tasks.[20]

Early apprehensions about the safety and reliability of autonomous systems were underscored by research findings, which pointed to public safety concerns. Over time, advances in multi-sensor information fusion and algorithmic improvements, such as those highlighted by the Kalman filter approach, have led to substantial enhancements in safety protocols and incident prevention strategies. Future research aims to fortify these systems to the extent that they exhibit enhanced safety standards with accident and incident rates

approaching zero, representing a significant evolution from the initial safety qualms.[20][26]

Initial technological implementations, characterized by rudimentary algorithms and basic sensors, have been substantially refined. Current advancements, as indicated by case studies involving the A* algorithm and its derivatives, have introduced advanced sensors and AI algorithms to the forefront of autonomous navigation systems. Predictive models suggest a continuum of progress leading to fully AI-driven systems equipped with enhanced capabilities and safety features, transitioning from the basic constructs of early technology to the sophisticated architectures predicted for the future.[25][26]

The economic impact of autonomous robots and vehicles has transitioned from being minimal during the early exploratory phase of research and development to a burgeoning market with the potential for significant societal benefits. As reported in industry studies, this growth trajectory is poised to position autonomous systems as key drivers in transportation sectors, promising substantial economic implications and transforming them from emerging innovations into pivotal economic entities.[27]

In summation, the collective body of research underscores a narrative of transformation for autonomous robots and vehicles, detailing an arc from initial skepticism to burgeoning acceptance, from foundational technological elements to sophisticated, AI-driven systems, and from marginal economic influence to substantial societal impact.

F. Our efforts

There are numerous factors that contribute to the problem of creating an autonomous system. The first problem is dynamic environments with moving obstacles, consistently changing environmental conditions leads to unpredictable scenarios that drastically affect the accuracy of mapping and analyzing data that is needed to be gathered and then saved to pull from. Additional limitations lie in the sensors that are utilized. The resolution and the range of a lidar impacts the quality and reach of the sensor data. If your

sensor has a small radial range, it increases the difficulty of accurately mapping out the entire terrain that is needed. Furthermore, there are computational constraints when it comes to data processing. To be able to simultaneously map and navigate an environment uses a large amount of processing power that is often not accessible or feasible for most systems. One of the biggest issues that face the world of AI is the concern for public safety. Since the beginning of having technology there have always been concerns surrounding what they are capable of. Due to these concerns the public is not as open to trying new things regarding robots. This creates a problem because it begs the question of the ethical practices that go into giving AI its space. When putting this into vehicles and robots the concern becomes even deeper as now lives are being put in the hands of something that does not have the same level of discernibility that having a consciousness provides. Due to this being a relatively new field there are not as many regulations to determine what is acceptable and what is potentially detrimental.

G. Tools Required

The tools needed to contribute to this project is to have a background in Calculus, Circuits, Mechatronics, Computer aided Design (CAD), and Python/C++ Programing. With this background then our team can contribute to this problem. The reason for these backgrounds is as follows.

There are many tools needed to contribute to this project and field. There needs to be a background in Calculus, Circuits, Mechatronics, Computer aided Design (CAD), and Python/C++ Programing. With this background our team can contribute to this problem.

With proficiency in Calculus, we get a better understanding of the algorithms that are going to be required for the conversion of the sensor data to readable machine data. As well as understanding the path planning and machine learning algorithms that are to be implemented in the project later. Without a background in math that is at least to the calculus level then this part of this project will be rather difficult to overcome.

We hope to bring forth research in this field by first establishing what we will be doing in this project. Next is doing as much research as possible, specifically research into current methods into map generation, path planning, and on machine learning algorithms that are applicable to this project. Then to start work on the virtual model of the robot that will provide a representation of what is to be done for the prototype and to serve as the brains of the robot. After these initial steps are complete the following steps are how we plan to proceed:

- Create a virtual visual representation of the robot, that is either stored as a file or a subscribe-able topic.
- Make the model be able to be moved, this is going to be a differential drive (2-wheel drive and control) robot for connivence.
- Create a method for a maze to be created/spawned in a virtual environment.
- Manufacture a method for the virtual robot to travel the maze.

- Utilizing virtual sensors to find a way to map and save data.
- Develop a Q-learning (Reinforcement machine learning method) so the robot can learn the most efficient path to finish.
- Establish a wireless method of communication between the virtual robot and the physical robot.
- Establish a state-publisher method that can transmit sensor data.
- After these initial steps are completed then a hardware model can be created and implemented. However, for the prototype model only the wireless communication method needs to be added as well as the state-publisher. The virtual model will then be synchronized to what happens in the physical prototype and will be responsible for the path planning computation.

H. Our contribution

According to The Survey of Path Planning Algorithms for Mobile Robots by Joshua Siegel et. al, The D* algorithm is good for quick path planning approaches but fails to guarantee a solution for dynamic environments. The Rapidly Exploring Random Trees method excels in dynamic environments solutions. Based on their research it is best to combine these methods to create a meta-heuristic hybrid approach. This is what we will base our algorithm around. The D* is an improvement on the Dijkstra method and the Rapidly Exploring Random Trees method is a genetic method.

The Dijkstra method is a method of programming that allows for the shortest path to an objective to be found. This is achieved by assigning a cost to nodes. The cheaper it is price wise to get to the node the shorter and more ideal the path is. It has a visited and unvisited node mode. Once an unvisited node is the next one on the list to be checked it is added into the queue and assigned a value and then transferred to visited mode. These are achieved by combining these 2 equations.[46]

These combined with the rest of the Algorithm sends individual little markers to every direction. These scan the area and make note of any walls and obstacles that might stand in its way. Once the scan is complete it finds the nodes that are closest together to get from point A to point B. The advantages to using this method are of course finding the optimal solution, but the other upside to this method is that it will always find a path to the objective. There might need to be adjustments for making sure it is the most optimal method but that is a much easier task if you are sure that a result is given. D* is an improvement on this method. That allows you to pull from preexisting data instead of starting from the same starting point every time.[46]

The Rapidly Exploring Random Trees method uses tree-like nodes to get from point a to point b. This method can use the end points to create paths and then choose the shortest pathway to each. It can also generate a field of points initially and then create tree branches around selected nodes to create a shorter path to the destination and will

continue this method until the shortest path is obtained. However, since this project is about the robot being fully autonomous, we must implement a machine learning algorithm, specifically reinforcement learning.

## Chapter 2

## THEORETICAL BACKGROUND

Notes Pages

1. Purpose
    A. Importance
2. Machine Learning
    A. Supervised
    B. Unsupervised
    C. Reinforcement Learning
3. Software
    A. ROS
        I. Key Features
    B. RVIZ
    C. Gazebo
    D. Algorithms
4. Hardware
    A. Microcomputers
        I. Raspberry Pi
        II. Arduino
        III. Jetson Nano
    B. Motor Drivers
        I. Stepper
        II. Servo
        III. H-Bridge
    C. Motor Controllers
        I. Open Loop
        II. Closed Loop
        III. Variable Frequency
    D. Motors

I. Electric
II. Hydraulic
III. Electro-Hydraulic

E. Sources
I. Solid State
II. Lithium Ion
III. Lithium Polymer

F. Lidars
I. Solid State
II. Flash

5. Conclusions
6. Upcoming

# Chapter 2
# THEORETICAL BACKGROUND

The purpose of this project is to develop a more efficient algorithm and system in which we can make a robot n navigate and map an environment. As well as effectively traverse the mapped environment using a machine learning algorithm to move about the generated map. This project can be applied to many industries, such as the logistics industry and transportation industries. If this were to be applied to the logistics industry warehouse management can become automated. After a map is created with details of the locations of each product location in the warehouse. There would no longer be a employees needed to transport the goods around or in/out of the warehouse, increasing the efficiency of the task, reducing cost as well as bringing automation to a field in the industry that hasn't been thought of. The other industry in which this would make an immediate impact is the transportation industry. All the affected industries stand to benefit generously from our contribution.

*Section 1: Purpose*

## B. Importance

This project is important due to its significance regarding the advancement of autonomous systems but more importantly the applications that this can be applied to. After the development of a mapping and path planning robot this can be implemented in logistics as well as the transportation industries. The immediate benefit of this would be the increased efficiency of operation, improved safety by removing some of the human element during day-to-day operations, automation could be applied were before it wasn't a feasible option. In the end the end goal for this type of technology is not stopping at just maze finding and traversing robots. This tech can be used and implemented in many ways in the future. Companies such as Tesla, Waymo, Rivian and others are developing

vehicles that can traverse their environment in real time. Our system can help further that technology at an even faster speed.

*Section 2: Machine Learning*

In artificial intelligence there are 2 different methods to attain progress. One of the best ways we have found is called machine learning. Machine learning is the idea that instead of explicitly telling Artificial Intelligence (AI) what to do and how to do it, the program will run over time, gain experience and be able to devise the most efficient and best way to complete the task. Machine Learning is based in data. Without data there is no material to train on, without material there is no foundation to learn on. There are many different forms of machine learning that include: Supervised, Unsupervised semi-supervised which are essentially a middle ground between supervised and unsupervised and will not be talked about in this paper.[16][19]

A. Supervised

Supervised learning alike the other forms is based on training over time to eventually get to the best outcome. There are two main types of Supervised learning: regression and data mining. There are 2 types of Regression: Linear Regression and Logistic Regression. Linear Regression gives a relationship to independent and dependent variables which helps with making projections. These projections can predict the way a data set will progress overtime and tell us what the data means. Logistic Regression is used to assess data that is already had. It takes binary inputs to then predict its next move. Data mining comes in many different forms including Random Forest and K-Nearest neighbor. Random Forest uses tree like nodes on all data points which are then used to find the closest pathway to the desired point. Once every data point has been explored then the most efficient tree pathway is used. Finally, K-nearest neighbor values data by how close it is to existing known points. Normally it calculates the Euclidean distance between a

known point and an unknown point to assign it an average and store it to then be used for other data points.[17]

B. Unsupervised

Unsupervised Learning differs from Supervised learning because of the type of data it chooses to use for its information. Supervised data has specific places to start from when looking for a certain point. Unsupervised learning specializes in unlabeled data which is based off not having specific pairs or beginning points beforehand. There are many types of Unsupervised learning as well including autoencoders, probabilistic clustering, and Hierarchical clustering. Autoencoders is the idea of taking the unlabeled data given and inputting it into something called a latent space that deciphers the data given and then refers it back to what was originally given. This is helpful because it takes what is essential from the data and determines its value to the system and then stores it. Probabilistic clustering takes the data and asses its vitality to the situation given. A data point is taken analyzed and is assessed based on its probability to be the best solution to the task. This allows to keep key points that are possibly the right answer in a separate set rather than having to reconsider all the points every single time. Finally Hierarchical clustering is similar to probabilistic clustering in the way that it asses the data. The difference between the two is instead of just throwing all the probabilities in one set they are organized from least likely to most likely in a chart of sorts. This prevents the program from having to reassess the probabilities every time it narrows down the set, now it can go through all the points once and in the end pick the best rated position and use that one.[18]

C. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning that stands out due to its dynamic approach to training an AI agent. Unlike other forms of machine learning, where the algorithm is provided with labeled data, RL learns through interaction with an environment. This process involves the AI agent taking actions and receiving feedback in

the form of rewards or penalties based on the outcome of those actions. Over time, through trial and error, the agent learns to associate certain actions with maximizing cumulative rewards. In the context of our project, where the goal is to develop an autonomous maze exploration and path planning robot, reinforcement learning emerges as a promising choice. By utilizing RL, we aim to empower our robot to learn how to navigate the maze autonomously, leveraging its ability to iteratively improve its decision-making based on past experiences and feedback. With a clear understanding of the robot's objectives and environment, reinforcement learning offers a pathway to achieve efficient and adaptive behavior, enhancing the robot's ability to fulfill its intended tasks with minimal human intervention.

*Section 3: Software*

The Autonomous Maze Exploration and Optimal Path Planning Robot represents a synthesis of complex software and hardware interactions. The technology stack comprising Linux Mint, ROS2 Foxy, Gazebo, and Rviz serves as the backbone for this integration. This chapter elucidates the rationale for selecting these technologies and summarizes their functions within the context of the project. Some systems include Linux Mint, ROS2, Gazebo and Rviz.

The purpose of using Linux Mint is for its stability, user-centric design, and compatibility with development tools. As an operating system, it provides a secure and reliable foundation for developing and running robotic applications. Linux Mint functions as the interface between the user and the robot's hardware, managing resources and ensuring smooth execution of programs. Its compatibility with ROS2 Foxy is especially beneficial, providing a seamless experience for software development and deployment.

Figure 2.1: Linux Mint

ROS2 Foxy is the framework that facilitates complex robotic interactions. It was selected for its robust communication protocols, real-time support, and modular design, crucial for contemporary robotic systems. And ROS2 Foxy acts as the middleware that manages

message passing between different parts of the robotic system, from sensor data acquisition to motor control. Its architecture enables the development of scalable and maintainable code, essential for the project's success.

Gazebo was chosen for its advanced simulation capabilities, which are necessary for testing the robot in a controlled yet realistic virtual environment. Gazebo provides a platform to simulate the robot within detailed, dynamic environments. It allows developers to test navigation algorithms and sensor integrations without the risk and expense of real-world trials, accelerating development and ensuring robustness.

Rviz is utilized for its real-time visualization capabilities. It allows developers and stakeholders to visually monitor the robot's internal state and understand its interactions with the environment. Rviz also serves as the interface for visualizing sensor data such as LiDAR and camera feeds, as well as the robot's pose and trajectory. This insight is invaluable for debugging and optimizing the robot's navigation algorithms.

The selection of Linux Mint, ROS2 Foxy, Gazebo, and Rviz for the Autonomous Maze Exploration and Optimal Path Planning Robot is a strategic decision grounded in the specific needs of the project. This combination of software offers a superior balance of usability, functionality, and integration over alternative solutions, making it the preferred suite for this endeavor.

When evaluating Linux Mint against other operating systems like Ubuntu or Fedora, its advantage lies in its user-friendly approach and out-of-the-box support for a broad range of hardware. Linux Mint is often praised for its stability and ease of use, which is paramount for complex developmental tasks in robotics, where system crashes or compatibility issues can cause significant delays.

ROS2 Foxy is the evolution of ROS1, providing improvements in communication, security, and real-time support. While ROS1 is still widely used, ROS2 Foxy's architectural advancements make it a better choice for future-proofing the project, ensuring compatibility with upcoming technologies and standards in robotics.

Gazebo, as a simulator, offers more sophisticated physics engines and environmental interactions than alternatives like Microsoft's AirSim or the CoppeliaSim (formerly V-

REP). While these are robust simulators, Gazebo's deep integration with ROS and its open-source community support provides unparalleled benefits for developing and testing complex robotic behaviors.

*Table 2.1: Software Alternatives*

| Software | Preferred (Rating 1-5) | Alternative | Alternative Rating |
|---|---|---|---|
| OS | Linux Mint (5) | Ubuntu (4) | Fedora (3) |
| Middleware | ROS2 Foxy (5) | ROS1 (4) | - |
| Simulation | Gazebo (5) | AirSim (3) | CoppeliaSim (4) |
| Visualization | RViz (5) | Simulink (3) | Unity3D (4) |

RViz's role as a visualization tool is unparalleled within the ROS ecosystem. Alternatives such as MATLAB's Simulink or Unity3D offer visualization capabilities, but they lack the seamless integration and real-time data handling that Rviz provides when working within a ROS-based framework.

The concerted use of Linux Mint, ROS2 Foxy, Gazebo, and Rviz provides a comprehensive technology stack that supports every stage of the autonomous robot's development lifecycle. Linux Mint ensures a robust operating system platform, while ROS2 Foxy provides the necessary communication infrastructure for modular and complex robotic systems. Gazebo offers a virtual proving ground for safely and efficiently testing the robot's capabilities, and Rviz supplies the visualization tools required for an in-depth analysis of the robot's performance. Together, these technologies form an integrated framework for developing an autonomous robot capable of navigating mazes with precision and adaptability.

A. ROS

The Robot Operating System (ROS) is a system designed for the advancement of the robotics industry. ROS is an open-source framework/middleware designed for building and controlling robotic systems. ROS provides a collection of software libraries, tools, other various conventions that help aid in the development of robotic applications. ROS also offers a distributed architecture that allows for the modular development of robotic systems. Along with a communication infrastructure for communication, enabling different software components, referred to as node, to exchange data and messages.

Figure 2.2: ROS OPERATING SYSTEM

These nodes can be written in a variety of different programming languages, however the primary 3 that are used are HTML, C++, and Python. Using these languages all nodes can communicate with each other on a network. ROS has gained popularity in the robotics community due to its flexibility. As time went on there have been several distributions of ROS and now a new version ROS2 is now available. ROS1 is catered more-so to single machine communication, while ROS2 has a looser structure that allows for multi-machine communication and scalability across networks.

*I. Key Features*

Key features that distinguish ROS from other platforms are its unique messaging system, package development system, visualization and debugging tools, arrangement of libraries.

1. Messaging System - ROS uses a publish-subscribe system of communication between nodes. Nodes can publish messages to specific topics, and others can subscribe to those topics to receive messages. This loose communication method allows for the connection between nodes to exchange sensor data, control commands and other relative information.

2. Package Management - ROS organizes code and other resources called packages. Based on the version of ROS use the package structure differs. ROS1 has a restrictive package structure and operates off a master slave system. Meaning you need one master node to be activated to access the others. ROS2 has a looser package and communication structure that allows for communication between multiple nodes without the slave system.
3. Visualization and Debugging Tools - ROS also includes various visualization and debugging tools to help aid in the development and analysis of robotic systems. These tools allow for the visualization of sensor data, robot models, and the behavior of nodes. Some software includes Rviz for the visualization of 3D robot models and sensor captured data. And Gazebo for simulating robots and virtual environments. All virtual simulators vary compared to that reality, but it is important to use simulators like that of Gazebo to partially determine the behavior of the robot system.
4. Libraries - ROS provides a wide range of libraries and tools for common tasks needed for the robotic system. Such as controlling hardware, interacting with sensors and actuators, performing navigation and mapping operations, and implementation of algorithms. Some examples of these libraries that will be in use of this project is the Robot Model (URDF) library and the ROS control libraries. The URDF library is required to manufacture a robot model in the form of a URDF.

B. RVIZ

ROS visualization (Rviz) is a powerful 3D visualization tool that plays a integral part of the Robot Operating System. It provides a graphical user interface for visualizing and debugging robotic systems and their sensor data. Rviz can do the following:

1. Visualization of Robot Models - RVIZ allows for the user to load and display 3D models of the robots, including the links, joints and other features of the robot system. These models are typically defined using the Unified Robot Description Format (URDF). This is the format that is present through most ROS systems and can easily be used in different nodes. Using this format Rviz gives the user the ability to view and interact with the robot model, enabling the user to inspect the structure and kinematics of the system.
2. Sensor Data Visualization - Rviz enables the visualization of sensor data produced by various types of sensors normally used in robotics systems. Such as cameras, laser scanners (LiDAR), and depth sensors. Using Rviz you can view sensor data in real-time allowing for you to analyze the environment around the robot and get an idea on how the robot perceives the environment. This is a feature that must be used to get an understanding of the sensor data.
3. Displaying the Robot State - Rviz is a component of ROS that allows for you to view the real-time information about the robot. Such information includes the robots state which includes the joint positions, velocities, angles. This feature allows for the verification of the robot's internal state during operation or during the systems development.
4. Displaying the Robot Trajectories - Rviz can visualize the planned/executed robot trajectories allowing you to see the path that the robot is following. This is useful for the debugging of motion planning algorithms or tracking trajectory performance.
5. Interaction and User controls - RVIZ provides interactive controls that allow the user to the robot model, change perspective of the camera, and change other elements of the system. Using built in elements such as the state publisher or the TF planner you can manipulate the robot model to inspect the robot behavior.
6. Configuration and customization - Rviz offers customization options to tailor to the user's needs. You can configure the appearance of the robot model, by

changing the color, style, and other visualization settings, and to save and load configurations. This feature allows you to capture images of your system.

7. And provide integration with other ROS tools - Rviz also seamlessly integrates with other ROS tools and libraries. One example of this is you can configure Rviz to display the live output of SLAM (Simultaneous Localization and Mapping) algorithms and point cloud data by visualizing lidar sensor data. Using this you can visualize this information as well as add on additional overlays such as maps, grids or coordinate frames.

Its primary function is to provide a graphical interface for displaying sensor data, robot state, and diagnostics in real-time. Rviz excels in rendering a visual representation of a robot's perception and interaction with its environment, making it invaluable for debugging and monitoring purposes. It allows users to interact with the data by adjusting viewpoints, visualizing trajectories, and examining the spatial relationships of sensor data within the environment.

No Physics Simulation: Rviz does not simulate physical interactions or dynamics. It visualizes pre-existing data and state information without generating or altering them.

C. Gazebo

The structure of the robot can be broken down into systems. Firstly, there is the operating system that the robot would be using as base. Second is the package structure, this is where the code for the robot and where the locations of the saved maps as well as all digital information would be housed. And then there is the system in which the physical components of the robot are connected. These are the base systems that are responsible of the motion and operation of the robot.

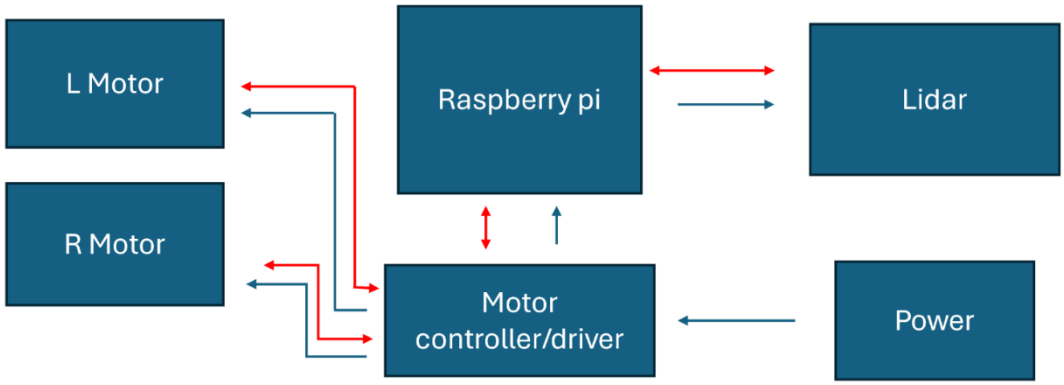
Starting with the operating system, due to previous projects that operate in a similar fashion to that of ours. We have chosen Linux as our operating system. In particular, the distribution of Linux called ubuntu 20.04. The reason for this distribution is due to a few reasons. First reason is because other researchers have used this version of Linux as an operating system for the reason of convenience and for a larger number of accessible

libraries that they could pull from. It is an older version of Linux that many in this field use and it is also recommended by many robot hobbyists and enthusiasts to get started. And the second reason is as mentioned previously, me and my group members have little experience in the field/technology. So, it is best to try and use the most user-friendly software as possible.

Next is the package structure/file structure of the code for the robot. The software that we are using as a base is called ROS (Robot Operating System). It is a system that is used by many robots in the industry. Using this software, it is possible to view the robot virtually, simulate the robot in various situations, and the coding language it uses is python, html, and C++. These languages don't require changes in syntax when it comes to the distribution they are implemented as well as with the libraries in which they utilize which makes compiling code easier as well as developing it.

The structure of the robot is going to be rather simple to reduce modes of failure and to make the programing structure more manageable. The components in the robot are a battery, TTY motors, a motor driver, motor controller, raspberry pi(brain), chassis, lidar(sensor) and wheels. The reason for these components is as follows. Starting with the battery, this is a 12-volt 4-amp battery that will provide all the components of the robot with power enabling them to function. There is no other purpose the battery provides; the current battery was taken from a RC airplane it is rather large. Later, it would be better to get a battery of a smaller form factor or possibly one of smaller size and capacity however for this project this should provide more than enough energy.

*Chart 2.1: Potential Hardware connections*




In the domain of autonomous robotic systems, the configuration of component connections is determined by the necessity for a streamlined flow of control signals and informational exchange. The Raspberry Pi serves as the central processing entity, its role being to receive and process data, which is facilitated through a serial over USB connection with the RPLIDAR A1 sensor. The choice of this connection method is predicated on its widespread adoption and reliability as a hardware interface, enabling the central unit to assimilate environmental data with efficiency.

Subsequently, the motor controller is engaged by the Raspberry Pi, whereby it acts as a translator of high-level directives into lower-level control signals apt for the motor driver's consumption. This translation is critical, as it bridges the gap between command issuance and physical actuation. The motor driver, receiving its instructions from the motor controller, is tasked with the regulation of power to the motors. This regulation is paramount in dictating the precision of the robot's movements and its velocity. Power provision is the domain of the battery, supplying the necessary electrical current to both the motor driver and the motors themselves. This provision is essential for the autonomous aspect of the robot, permitting operation independently of tethered power sources.

The schematic of connections is designed with the objective of optimizing the robot's operational efficacy. It is essential for each component to maintain effective

communication pathways with its counterparts. This ensures responsiveness to navigational commands, real-time processing of environmental inputs, and the maintenance of control over the robot's locomotive functions. Such a systematic arrangement of components underpins the effective functioning of an autonomous robotic system, ensuring its ability to navigate and perform tasks with autonomy and precision.

The most common operating system that is used for autonomous systems as well as small scale robotics systems is Linux. The specific distribution is not important however the overall system is. Linux is an operating system that is tailored to developer and program since it allows more freedom when it comes to using different coding languages, which allows more projects to be completed. As it pertains to this field in robotics, it is important since it allows scalability and communication in the robotics and manufacturing industries. However, after a Linux distribution is installed (using Ubuntu for the purposes of this project (The reason for this distribution is since it is the easiest one to install on my present computer and microcomputer)).
Important Note: Whichever, distribution of Ubuntu that is installed on the system (the version you can install is dependent on the kernel of your computer).

The software is implemented by installing the Linux distribution to the computer that will handle the graphical/computation parts of the project. And the same version of the build must be installed on the system. With the same system being installed on both systems, communication via a ssh protocol or by a shared IP on a private or semi-private network. The reason for this is for better communication between the robot and its sensor data and the computer.

*Table 2.2: Software Selection*

| Feature/Aspect | Gazebo | RViz | Additional Environment Example |
|---|---|---|---|
| **Primary Use** | High-fidelity simulation of robots and environments, including physics-based dynamics. | 3D visualization tool for ROS, focusing on displaying sensor data and robot state. | **Unity3D**: Game engine with strong support for VR/AR and realistic simulations. |
| **Physics Simulation** | Yes, provides integrated physics engines (ODE, Bullet, etc.) for realistic simulations. | No, primarily used for visualization without physics simulation. | Yes, supports physics simulation through its engine. |
| **Integration with ROS** | Native support for ROS/ROS2, allowing direct integration with robotic software stacks. | Designed as a part of the ROS ecosystem, offering seamless integration for visualization. | Possible through plugins or middleware but not inherently designed for ROS. |
| **Realism** | High, with support for lighting, textures, and environmental effects. | Medium, focuses on representing data and state rather than realistic rendering. | High, especially with advanced graphics and environmental effects. |
| **Learning Curve** | Moderate to high, depending on the complexity of simulations and familiarity with ROS. | Moderate, with a focus on data visualization concepts. | High, given its broad range of features beyond robotics. |
| **Community Support** | Strong, especially within the robotics community. | Strong, as an integral tool within the ROS ecosystem. | Very strong, with a vast user base in gaming and interactive media. |
| **Customization** | High, with the ability to create detailed models and environments. | Medium, limited to visualization aspects. | Very high, with extensive support for custom assets and scripts. |

For this project to reduce the strain on the robot's system, division of the work is necessary. Based on similar projects involving navigation, the systems are divided into 2

parts. There is an on-board computer on the robot that is responsible for the robots' movements as well as sending the input & saved data from the lidar to the other system. The other system is responsible for processing the data and sending it back to the robot to follow the command input on how to navigate the environment. The reason for this is to increase the performance of the overall process, the raspberry pi can be responsible for all the computation required of this project however, due to the low processing power, it would be more optimal for most of the computations be offloaded to an exterior computer with more power.

D. Algorithms

There are different types of algorithms that can be used in this project and that used in the industry. There are several types of applicable algorithms that can be used for this project. Due to the vast number of algorithms that were created for the advancement of robotics, they can be divided into three categories, Sample-based motion planning, Graph search, and machine learning algorithms. There are many variations of each type, each providing their own advantages and disadvantages. Beginning with the sample-based motion planning. Sample-based solvers, function by randomly sampling points and attempting to connecting points to form a path. The algorithm begins by starting with the initial position and then takes random actions (plotting points) to explore the entire area. The algorithm keeps track of these states and will build a graph representing the maze. This will continue till it has fully mapped the maze. An algorithm that takes use of this approach is RRT (Rapidly Expanding Random Tress and PRM (Probabilistic Roadmaps). RRT and PRM are similar in nature, much like the standard approach of the algorithm. Graph search algorithms differ from the sample-based solver algorithms. Graph search algorithms construct a graph representation of the maze and use graph search techniques to find the optimal path of travel. Such techniques include algorithms such as Dijkstra's algorithm, A* and D*. Most of these algorithms are variations one another, each algorithm with slightly different variations.

For the machine learning algorithms there exist more categories. These categories influence the learning speed of the robot, the error of the system, stability and performance of the system. Such policies include Deterministic, Stochastic, Gaussian, Categorical, SoftMax, Epsilon, Boltzmann, Adaptive and Off Policies. There are many more policies that exist these are just the few that I believe are relevant to this project. But before you can dive in and learn about the different policies and their patient uses, there is another important policy type that one must understand first. That is on-policy learning and off-policy learning. On-policy learning is when the learning agent learnings the value of the function according to the given policy. The agent follows the current polices that it has learned and experienced and learns based on this information alone. On-policies only learn off information that was presented to it at the start. An example of this is a robot learning how to navigate a maze. As the robot navigates the maze learning how to solve the maze it updates its policies based on the rewards it receives. The key point of this that the robot using this policy learns from the factions that it takes, its learning is dependent on its policy of exploration. Off-policy learning is like that of the on-learning. The difference is that it operates off two polices now compared to one. The first policy the behavior policy that dictates the movement of the robot, then there is an additional policy that is learning policy. Using these policies the robot learns by taking actions according to its learned behavior and it also by modifying the Q-values in the policy to receive the maximum reward from its next state. Meaning that the robot can learn an optimal policy (the fastest path to the exist) and work on suboptimal policy behavior (implement random behavior to explore random states).

Deterministic policies, operate off the idea that the output state always takes from input state and vice versa. The general goal of this policy is to learn the optimal controls for all possible states so that the learning agent can receive the maximum reward. Thus, leading to faster and more efficient learning. However, this policy doesn't perform well in all circumstances, these circumstances include dynamic environment, determining multiple optimal actions per given state, and when encountered with high-dimensional environments. These environments don't fair too well this policy at times due to it

sometimes succumbing to looping suboptimal/unstable behavior. There is a change for it to overcome this behavior however it will take a long time compared to other policies, that overcome this through other methods.

Stochastic Policies are a little more complicated compared to deterministic policies. Every state is selected based on a non-zero probability according to a probability distribution function. It would first derive a rule for each state, and the sum of the probabilities for each given probability must some to one, then the most probable state is selected as the output.

$$Q(s,a) = r + \gamma \cdot \min_{i=1,2} Q_{\text{target},i}(s, \mu(s) + \epsilon)$$

Figure 2.3: TD3 algorithm

When it comes to the machine learning algorithm, we believe that TD3 is the best method in which the robot can learn to find the best path available. Twin Delayed Deep Deterministic policy gradient algorithm (TD3) is a popular reinforcement learning algorithm designed with the intension for continuous action spaces. A continuous action space simply means that the robot can choose between actions from a continuous range of values rather than choosing from static predefined actions. This means that the robot can have more fluid and precise movement in its environment.

- Q (s, a) represent the action-value function, estimating the expected return of taking an action(a) in state (s)
- r is the immediate reward received after acting (a) in state (s)
- y is the discount factor, valuing future rewards
- $Q_{target}$, is the target Critic networks used to estimate the value of the next state s
- u(s) is the action proposed by the target Actor network for the next state.

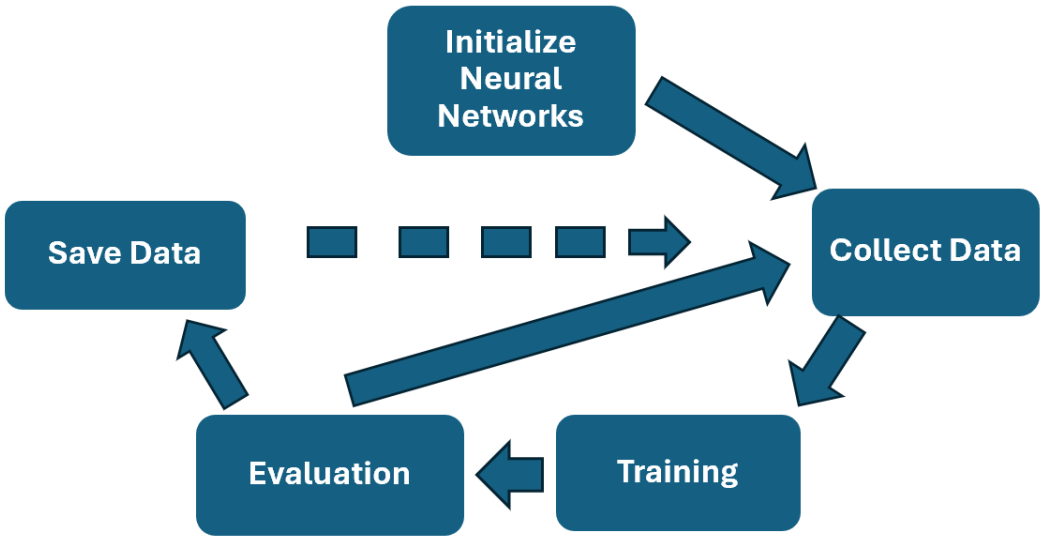
The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm operates through a series of interconnected steps aimed at iteratively improving the performance

of the actor and critic networks. Initially, the algorithm initializes both the critic and actor networks, which are responsible for approximating the value function and policy, respectively. These networks are typically deep neural networks, initialized with random weights.

Once the networks are set up, the algorithm begins the data collection phase. During this phase, the initial policy generated by the actor network is used to interact with the environment, gathering experience tuples consisting of states, actions, rewards, and next states. These tuples form the basis of the training data for both the critic and actor networks.

After collecting sufficient data, the algorithm enters the training phase. Here, it utilizes the collected experience tuples to update the parameters of both the critic and actor networks. The critic network is trained to minimize the temporal difference (TD) error between its predicted Q-values and the target Q-values, which are computed using the Bellman equation. Simultaneously, the actor network is trained to maximize the expected cumulative reward by adjusting its parameters to produce actions that lead to higher Q-values.

*Chart 2.2:TD3 Simplified Method*



Following training, the trained model is evaluated using a predefined reward function specific to the project's objectives. This evaluation serves to assess the performance of the model and determine whether it meets the desired criteria. Based on the evaluation results, the trained model may be saved if its performance meets the specified threshold or discarded if it fails to meet expectations.

The cyclical nature of the TD3 algorithm allows for continuous refinement of the actor and critic networks over multiple iterations. By iteratively collecting data, training the networks, and evaluating the resulting model, the algorithm progressively improves the policy, enabling the robot to navigate the maze more effectively. This iterative process leverages the power of deep learning to adapt and learn from past experiences, ultimately leading to more robust and efficient decision-making in complex environments like the maze.

Reinforcement learning algorithms, such as TD3 and DDPG, are suited for continuous action spaces as they can be trained to learn to approximate and optimize policies that can operate within a confined range of actions. These algorithms use function approximators, like deep neural networks, to learn the mapping between the robot’s current operating state and the optimal actions that need to take place. But optimizing the policy through training in an environment and receiving feedback in the form of rewards or punishments, the robot can learn to select the actions that maximize its performance for its task.

Continuous actions spaces provide increased flexibility and expressiveness in robotic control. By allowing robots to perform complex, and precise movements. However, additional challenges are also created in the way of optimization due to the continuous and multi-dimensional nature of the action space. These algorithms try to address some of these present issues by using overestimation bias and policy exploration factors that account for some of the errors that are prevalent in the system.

TD3 introduces some key enhancements over its predecessor DDPG to improve its stability and performance. One of these enhancements is the introduction of twin critics, (policies), which help to reduce the overestimation bias in the Q-value estimates.

However, given the complexities of this project and the needs required of the virtual and prototype model it is best that we utilize deep learning along with the TD3 algorithm in this project. It's a practical choice because as the robot moves and learns, it's building a map of the maze, getting closer to its target with every iteration.

*Section 4: Hardware*

When dealing with AI hardware, a variety of components are need in the construction and functionality of its systems. Microcomputers are employed as the central processing units, orchestrating the operations of the device by executing software algorithms. Motor drivers and motor controllers are utilized to modulate and direct the power to motors, which are responsible for the movement and mechanical actions of the system. Batteries are integrated as the primary source of energy, supplying the necessary power for all components to function effectively. Lastly, Lidars are incorporated to provide spatial awareness and environmental perception, enabling the system to navigate and interact with its surroundings with a high degree of accuracy. Each of these components plays a critical role in the sophisticated orchestration that underpins AI hardware, contributing to its ability to perform complex tasks autonomously.

A. Microcomputers

In the landscape of microcomputers, devices such as the Raspberry Pi, Arduino, and Jetson Nano have been developed to cater to a wide range of applications, from educational purposes to sophisticated computing tasks. By the Raspberry Pi Foundation, the Raspberry Pi was created to promote computer science in schools, offering a platform for learning programming and digital making. The Arduino, characterized by its ease of use for beginners and its open-source hardware and software, is utilized primarily for electronic projects and prototyping, aiming to make the technology accessible to a broad audience. Meanwhile, the Jetson

Figure 2.4: Raspberry Pi

Nano, developed by NVIDIA, is targeted towards edge computing, artificial intelligence, and machine learning projects, providing considerable computational power to handle complex algorithms and processing. Each of these microcomputers has been embraced by hobbyists, educators, and professionals alike, enabling innovation and creativity in a variety of fields. Through their development and widespread adoption, the barriers to entry for engaging with technology and computing have been significantly lowered, fostering a culture of learning and experimentation.[28][29]

*I. Raspberry Pi*

Raspberry Pi is widely recognized as a compact, affordable computing device that has been embraced by enthusiasts, educators, and professionals alike for a variety of projects and educational purposes. Developed by the Raspberry Pi Foundation, it is designed to promote the teaching of basic computer science in schools and developing countries. With its small size, low power consumption, and versatility, the Raspberry Pi is often used as a platform for experimentation, learning programming languages, building hardware projects, and even serving as a media center or web server. Over the years, several models and versions have been released, each improving on the capabilities and features of its predecessors, thereby ensuring that computing is made accessible to a broader audience. The impact of the Raspberry Pi on learning and innovation in the field of computer science cannot be overstated, as it has been instrumental in demystifying computing and making technology more accessible to people around the world.[28]

*II. Arduino*

Arduino is celebrated for its accessibility and versatility in the realm of electronic projects and prototyping. Developed with an emphasis on simplicity and ease of use for beginners, it consists of both open-source hardware and software. The core of Arduino's appeal lies in its user-friendly development environment, which allows for the straightforward writing of code and uploading it to the microcontroller. This has led to its widespread adoption in educational contexts, maker communities, and among hobbyists seeking to explore the possibilities of electronics and programming. Various models of Arduino boards have been introduced over the years, each designed to cater to different project requirements and skill levels. As a result, a rich ecosystem of add-on modules, known as shields, has been developed, enabling the extension of the platform's capabilities to include functionalities such as internet connectivity, motor control, and sensor integration. Through the collective efforts of its community, Arduino has played a pivotal role in democratizing access to technology, inspiring a generation of creators to innovate and share their creations.[29]

Figure 2.5: Arduino Uno

*III. Jetson Nano*

The Jetson Nano, developed by NVIDIA, is recognized as a small, powerful computer specifically designed to power the next wave of edge computing and AI applications. Aimed at developers, researchers, and enthusiasts in the field of artificial intelligence, it has been equipped with the capability to run multiple neural networks in parallel for applications such as image classification, object detection, segmentation, and speech processing. The platform's accessibility is ensured by its affordability and the


Figure 2.6: Jetson Nano

provision of comprehensive development tools, including pre-trained AI models and software libraries, which are made available to facilitate the development of AI projects. By leveraging the processing power of the Jetson Nano, users are enabled to bring AI to a variety of devices and applications, making advanced computing accessible to a wider audience. The impact of the Jetson Nano in the domain of AI and machine learning is significant, as it serves as a bridge, lowering the barrier to entry for those looking to explore the potential of AI technologies.[30]

*Table: 2.3: Component Selection*

| Component | Pros and Cons | Chosen |
|---|---|---|
| Raspberry Pi | Pros: Highly versatile, supports extensive programming languages, large community support. Cons: Limited IO speed or real-time applications. | Yes |
| Arduino | Pros: Ideal for beginners, extensive libraries and community support, great for small projects. Cons: Limited processing power and memory, not suitable for complex tasks. | No |
| Jetson Nano | Pros: Powerful GPU for Al and machine learning projects, supports advanced computing needs. Cons: Higher cost and power consumption than other microcontrollers. | No |

B. Motor Drivers

When it comes to Autonomous technology, motor drivers play a crucial role in facilitating precise control over motors, including stepper, servo, and those operated by H-bridge circuits. Stepper motor drivers are utilized for their ability to control the position and speed of stepper motors with high precision, enabling their use in applications requiring exact movements. Servo motor drivers, on the other hand, are employed to control servo motors, which are prized for their ability to maintain precise control of angular or linear position, velocity, and acceleration. H-bridge motor drivers are designed to enable the direction of the current through the motor to be changed, allowing for the motor to be driven forward or in reverse. These drivers are integral to the operation of various types of motors, each serving a specific function in the control and implementation of motor-based projects. Using stepper, servo, and H-bridge motor drivers, the execution of complex movements and tasks in automated systems and robotics is made possible, highlighting their importance in the development, and functioning of these technologies.[31][32][33]

*I. Stepper*

Stepper motors are highly valued for their ability to convert electrical pulses into discrete mechanical movements, making them indispensable in applications requiring precise control of position and speed. The characteristic of producing a high torque at low speeds, coupled with the capability to hold their position when not moving, makes them particularly useful in situations where control and precision are paramount. Unlike conventional motors, steppers are operated by digital pulses, which means that their rotation can be precisely controlled in terms of both angle and speed, allowing for exact positioning and repeatability of movements. This feature is extensively utilized in CNC machines, 3D

Figure 2.7: Stepper

printers, and robotics. The design and functionality of stepper motors enable them to be driven without the need for a feedback system to control position, which simplifies their application in controlled environments. Consequently, stepper motors have been widely adopted in industries where precise motion control is required, underlining their importance in the advancement of automated and precision machinery. [31]

*II. Servo*

Servo motors, either AC or DC, are characterized by their ability to provide exact control of angular or linear position, velocity, and acceleration. This is achieved using a feedback loop that continuously adjusts for irregularities between the motor's actual position and the commanded position. This characteristic makes servo motors highly sought after in applications where precision and control are critical, such as in robotics, radio-controlled vehicles, and aircraft. servo motors are capable of smooth and continuous movement, which allows for more precise control over motion. The feedback mechanism typically involves a potentiometer or an encoder, ensuring that precise positioning is maintained throughout the operation. Additionally, servo motors are known for their efficiency and versatility, being able to match the speed and torque requirements of the task at hand. The incorporation of servo motors into various systems has been instrumental in enhancing the performance and capabilities of automated and remote-controlled devices, highlighting their pivotal role in modern technology applications.[32]

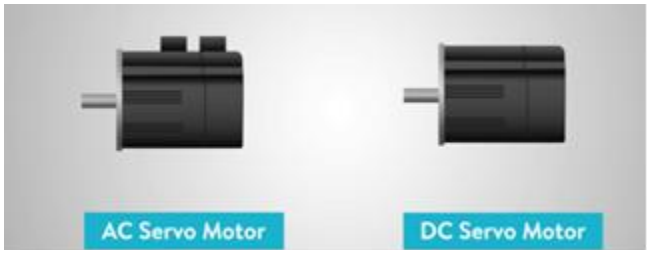


Figure 2.8: AC/DC servos

*III. H-Bridge*

The H-bridge circuit is widely recognized for its pivotal role in controlling the direction and speed of DC motors, enabling the motor to be driven in both forward and reverse directions. This functionality is achieved through the strategic arrangement of four switches (which can be transistors or relays) that form the shape of an "H," allowing current to flow through the motor in either direction. The control of these switches, often facilitated by microcontrollers or other control devices, permits the modulation of motor speed via pulse-width modulation (PWM) signals. By adjusting the duration of these signals, the effective voltage across the motor is varied, thereby controlling its speed. The versatility and effectiveness of H-bridge circuits have made them indispensable in the design of electronic projects involving motor control, such as robotic vehicles, automation systems, and anywhere precise control over motor movement is required. The deployment of H-bridge circuits underscores their importance in the advancement of technologies that require the dynamic control of motion.[33]

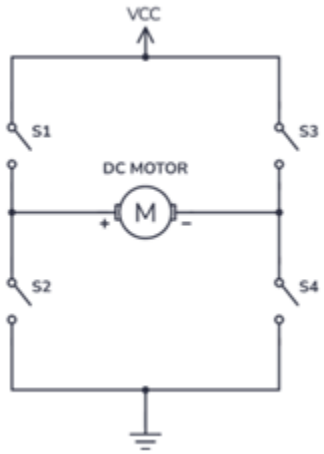



Figure 2.9:H-bridge

*Table 2.4: Component verification*

| Components | Pros and Cons | Chosen |
|---|---|---|
| Stepper | Pros: Provides precise control over stepper motors, ideal for applications requiring exact positioning. Cons. Requires complex programming for effective use | No |

| | | |
|---|---|---|
| Servo | Pros: Easy to integrate and use, offers precise positioning for small to medium loads. Cons: Limited by torque capabilities and not suitable for heavy loads. | No |
| H-Bridge | Pros: Simple design, capable of driving motors in both directions, useful for various application cons: Can generate significant heat, efficiency may be a concern. | Yes |

C. Motor Controllers

Motor controllers are essential components in the regulation of motor performance, encompassing a variety of types such as open-loop, closed-loop, and variable frequency drives (VFDs), each designed to meet specific control needs. Open-loop controllers are utilized for their simplicity and cost-effectiveness, operating without feedback to regulate motor speed, making them suitable for applications where precision is not critical. In contrast, closed-loop controllers are characterized by their use of feedback mechanisms to adjust the motor's operation continuously, ensuring precise control over speed, position, or torque, which is vital in applications demanding high accuracy. Variable frequency drives, on the other hand, are specifically designed to control the speed of AC motors by varying the frequency and voltage supplied to the motor, offering enhanced efficiency and control in applications ranging from industrial processes to HVAC systems. By adjusting the power supplied to the motors, VFDs enable significant energy savings and improved performance. The diverse functionalities

of open-loop, closed-loop, and variable frequency drives highlight their integral role in optimizing motor control across a wide range of applications, from simple mechanisms to complex industrial systems.[34][35][36]

*I. Open-Loop*

Open-loop control systems are characterized by the absence of feedback for regulating their operation. In these systems, the input signal is processed to generate an output, but the actual outcome is not compared against the desired result to adjust. This approach is utilized due to its simplicity and cost-effectiveness in applications where precision in control is not the primary concern or where the output can be predicted reliably from the input. Open-loop systems are commonly employed in situations where the relationship between input and output is straightforward and predictable, such as in basic motor speed controllers, where a fixed input leads to a predictable speed, or in heating systems where a specific input determines the heat output. The advantage of open-loop systems lies in their straightforward design and operation, eliminating the complexity and expense associated with feedback mechanisms. However, their effectiveness is limited by changes in conditions or load, which can lead to variations in performance, highlighting the trade-off between simplicity and precision in control strategies.[34][35]

II. Closed-Loop

Closed-loop control systems, known for their precision and adaptability, are distinguished by the integration of feedback mechanisms that continuously monitor and adjust the system's output to match the desired setpoint. In these systems, the output is constantly measured and compared to the input command, with any deviation resulting in adjustments to minimize the error. This feedback loop enables the system to automatically correct for disturbances or changes in operating conditions, ensuring consistent performance and accuracy. Closed-loop systems are extensively applied in various domains, including temperature control systems, where precise maintenance of a

set temperature is critical, and in robotic systems, where exact positioning is essential. The capability of closed-loop systems to adapt to external changes and internal variations makes them highly effective for applications requiring strict control over output parameters. The reliance on feedback to achieve desired outcomes underscores the sophistication and complexity of closed-loop systems in contrast to their open-loop counterparts, facilitating enhanced control and efficiency in dynamic environments.[34][35]

*III. Variable Frequency Drives*

Variable Frequency Drives (VFDs) are recognized for their critical role in the control of AC motor speed and torque by varying the motor's electrical power supply frequency and voltage. Through the adjustment of these parameters, VFDs enable motors to operate more efficiently and with greater control over a wide range of speeds, significantly reducing energy consumption and mechanical stress. This capability is particularly beneficial in applications where the motor's load conditions vary, such as in pumping, ventilation, and machine tool operations. The use of VFDs also contributes to the extension of the lifespan of motors and mechanical components by allowing for soft start capabilities, thereby reducing the inrush current and mechanical shock during startup. Furthermore, the precise control afforded by VFDs enhances process control in industrial applications, leading to improved product quality and operational efficiency. The adoption of VFDs across various sectors underscores their importance in achieving energy efficiency, operational flexibility, and cost savings in motor-driven systems.[36]

*Table 2.5: Component Type comparisons*

| Component | Pros and Cons | Chosen |
|---|---|---|
| Open-Loop | Pros: Simple and cost-effective solution for applications where precision is not critical. Cons: Lacks | No |

| | | |
|---|---|---|
| | feedback for accuracy, not suitable for precision tasks. | |
| Closed-Loop | Pros: Offers high accuracy and control through feedback, can adjust to changes in load dynamically, Cons: More complex and expensive than open-loop systems. | Yes |
| Variable Frequency Drive | Pros: Efficiently controls the speed of AC motors, can significantly reduce energy consumption, Cons: Implementation can be complex and costly. | No |

D. Motors

Motors, encompassing electric, hydraulic, and electro-hydraulic types, are pivotal in the conversion of energy into mechanical motion, each utilizing distinct mechanisms to achieve this transformation. Electric motors are characterized by their use of electrical energy to generate mechanical rotation, widely adopted for their efficiency and reliability in a multitude of applications ranging from household appliances to industrial machinery. Hydraulic motors, in contrast, operate on the principle of fluid power conversion, where pressurized hydraulic fluid is used to produce rotational motion or linear displacement, favored in heavy-duty applications for their high torque output at low speeds. Electro-hydraulic motors combine elements of both electric and hydraulic technologies, leveraging electrical control to manage hydraulic systems, thereby offering

precise control, efficiency, and adaptability in applications requiring sophisticated motion control solutions. Through the utilization of these diverse motor types, a broad spectrum of operational requirements and application-specific challenges are addressed, highlighting the critical role motors play in the advancement of modern mechanical and industrial technologies.[37][38][39]

*I. Electric*

Electric motors are widely utilized for their ability to convert electrical energy into mechanical energy with high efficiency and reliability. In these motors, the interaction between the magnetic field generated by the armature and the external field within the motor results in the generation of force, which produces rotation. Various types of electric motors, including AC (alternating current) and DC (direct current) motors, are employed depending on the specific requirements of the application, such as speed control, torque, and power level. The versatility of electric motors allows for their extensive use across numerous domains, from powering small household appliances to driving the machinery in large industrial plants. The design and operational principles of electric motors ensure that they are a crucial component in a multitude of electrical machinery and devices, facilitating the conversion of electrical energy into useful mechanical work. The widespread adoption of electric motors underscores their importance in modern technology, contributing significantly to the efficiency and functionality of electrical systems and devices.[37]

*II. Hydraulic*

Hydraulic motors are recognized for their ability to transform hydraulic energy into mechanical energy, employing the flow and pressure of hydraulic fluid to generate rotation or linear motion. These motors are distinguished by their robustness and high torque output at low speeds, making them particularly suitable for heavy-duty applications where substantial force is required. The principle of operation involves the hydraulic fluid acting on the motor's vanes, pistons, or gears, depending on the type of hydraulic motor, to create motion and force that is then transmitted to perform work. Given their efficiency and reliability, hydraulic motors are extensively used in a variety of settings, including construction equipment, industrial machinery, and maritime applications. The adaptability of hydraulic motors to deliver powerful and controlled movement under demanding conditions highlights their critical role in systems where precision and strength are paramount. Through their deployment in diverse applications, hydraulic motors demonstrate their indispensable contribution to the operation of complex mechanical systems, emphasizing the significance of hydraulic power in modern industry and engineering.[38]

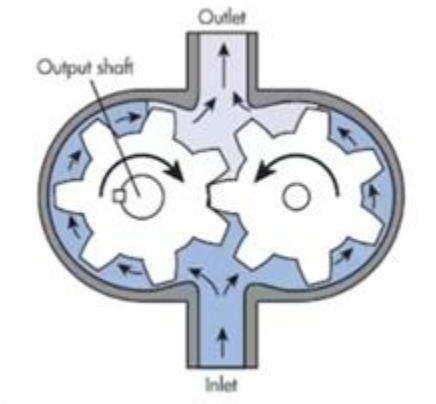



Figure 2.10: Hydraulic motor

*III. Electro-Hydraulic*

Electro-hydraulic systems are acknowledged for their integration of electrical and hydraulic mechanisms to achieve superior control and efficiency in the conversion of electrical energy into mechanical force. In these systems, electrical signals are utilized to control the flow and pressure of hydraulic fluid, enabling precise manipulation of hydraulic motors and cylinders. This fusion of technologies allows for the fine-tuning of movements and the application of significant force with high accuracy, making electro-hydraulic systems especially valuable in applications requiring detailed motion control,

such as in robotics, aerospace, and manufacturing processes. The ability to programmatically adjust hydraulic responses through electrical inputs provides a level of versatility and precision that is difficult to achieve with purely hydraulic or mechanical systems. Consequently, electro-hydraulic systems have become integral in the design and operation of complex machinery, offering a sophisticated solution that leverages the strengths of both electrical and hydraulic power to meet demanding operational requirements. The widespread adoption of electro-hydraulic technology underscores its importance in enhancing the capability and efficiency of modern mechanical and industrial systems.[39]

E. Sources

*Sources*, including solid-state, lithium-ion, and lithium-polymer types, are essential in providing portable power to a wide array of electronic devices, each employing different materials and technologies to store and release energy. Solid-state batteries are characterized by their use of solid electrolytes and electrodes, which are known for their potential to offer higher energy densities, enhanced safety, and longer lifespans compared to their counterparts with liquid or gel electrolytes. Lithium-ion batteries, widely adopted in consumer electronics, electric vehicles, and renewable energy systems, utilize liquid electrolytic solutions and are prized for their high energy density, rechargeability, and efficiency. Lithium-polymer batteries, a variation of lithium-ion technology, employ a solid polymer electrolyte that allows for thinner, lighter, and more flexible designs, making them particularly suitable for portable electronics where form factor is critical. The evolution and diversification of battery technologies, as represented by solid-state, lithium-ion, and lithium-polymer batteries, have been instrumental in advancing the performance, reliability, and safety of portable power sources, highlighting their pivotal role in the proliferation of modern electronic devices and systems.[40][41][42]

*I. Solid State*

Solid-state batteries are distinguished by their incorporation of solid electrolytes, as opposed to the liquid or gel electrolytes found in conventional batteries, offering significant advancements in safety, energy density, and longevity. In these batteries, the risk of leakage or combustion is greatly reduced due to the absence of liquid components, which inherently enhances their safety profile. Additionally, the solid electrolyte facilitates a more stable electrochemical environment, allowing for the use of more energy-dense electrode materials and potentially extending the battery's lifespan. This stability also contributes to a higher resistance to temperature variations and physical damage. The development and research into solid-state batteries are driven by the demand for safer, more efficient, and longer-lasting power sources for applications ranging from portable electronics to electric vehicles. Despite the technical challenges associated with manufacturing and scalability, the potential benefits of solid-state batteries, such as increased energy density and improved safety features, position them as a promising direction for future energy storage solutions. The ongoing advancements in solid-state battery technology underscore its potential to significantly impact the landscape of energy storage and power delivery in the coming years.[40]

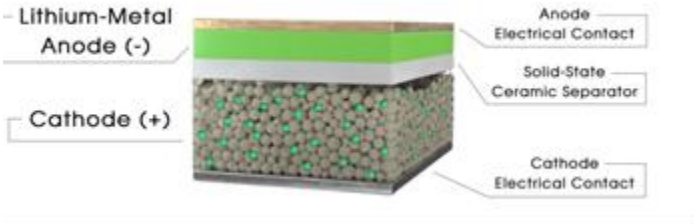



Figure 2.12: Solid State battery

II. Lithium-Ion

Lithium-ion batteries are celebrated for their high energy density, rechargeability, and versatility, making them a popular choice across a wide range of applications, from portable electronics to electric vehicles and renewable energy storage systems.

These batteries operate on the principle of lithium ions moving between the anode and cathode, through an electrolyte, during charging and discharging cycles. The ability to efficiently store and release energy has led to their widespread adoption, providing a reliable power source for devices requiring long-lasting, compact, and lightweight batteries. Furthermore, the advancements in lithium-ion technology have contributed to improvements in power capacity, longevity, and safety, though challenges related to thermal stability and resource availability persist. The significance of lithium-ion batteries in the modern technological landscape is underscored by their role in enabling the mobility of electronic devices, the expansion of renewable energy applications, and the progression towards electrification of transportation, illustrating their critical contribution to energy storage and management solutions.[41]

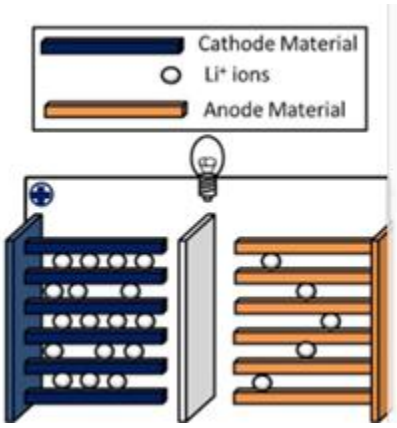



Figure 2.13: Lithium-ion battery

*III. Lithium-Polymer*

Lithium-polymer batteries, characterized by their use of a solid polymer electrolyte instead of a liquid electrolyte, are noted for their flexible form factors, lightweight design, and safety profile. This technology enables the batteries to be manufactured in a variety of shapes and sizes, catering to the specific needs of compact and portable devices where space and weight are critical considerations. The solid polymer electrolyte significantly reduces the risk of leakage and enhances safety, making lithium-polymer batteries a preferred choice for consumer electronics, such as smartphones, tablets, and laptops.

Moreover, these batteries exhibit a lower chance of experiencing electrolyte leakage compared to their lithium-ion counterparts, contributing to their reputation for safety. Despite generally offering a slightly lower energy density than lithium-ion batteries, the versatility and adaptability of lithium-polymer batteries have solidified their position in the market, particularly in applications requiring custom battery shapes and sizes. The development of lithium-polymer technology continues to evolve, focusing on improving energy density, extending lifespan, and reducing costs, reflecting its ongoing importance in the advancement of portable power solutions.[42]

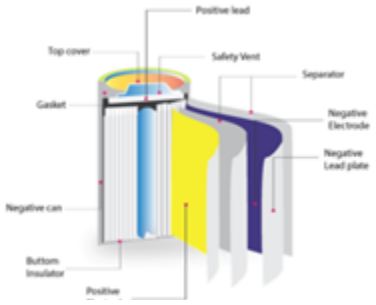


Figure 2.14: Lithium-polymer battery

*Table 2.6: Battery Type Comparison*

| Component | Pros and Cons | Chosen |
|---|---|---|
| Solid State | Pros: Offers higher energy density and improved safety features compared to other battery types. Cons: Still in development, with higher costs and manufacturing challenges. | No |
| Lithium-Ion | Pros: Widely used with a proven track record, offers high energy density and rechargeability. Cons: Potential safety risks with overheating and fires. | No |

| | | |
|---|---|---|
| Lithium-Polymer | Pros: Flexible design options, lighter and can be shaped more freely than Li-ion batteries. Cons: Generally, offers lower energy density than lithium-ion. | Yes |

F. Lidars

Lidar technology, encompassing solid-state, flash, and hybrid variants, plays a crucial role in enabling precise and detailed three-dimensional mapping and object detection across various applications, from autonomous vehicles to geographical surveying. Solid-state lidars are distinguished by their lack of moving parts, offering enhanced durability and reliability, making them well-suited for integration into vehicles and mobile devices. Flash lidar systems, by emitting a wide beam to illuminate the entire scene simultaneously, allow for rapid capture of the environment without the need for mechanical scanning, which is advantageous for applications requiring high-speed detection. Hybrid lidars combine elements of both mechanical and solid-state technologies, aiming to leverage the advantages of each to achieve both high resolution and wide coverage. This amalgamation of technologies within the lidar spectrum is instrumental in catering to diverse operational requirements, ensuring versatility in deployment. The continuous evolution and specialization of lidar systems underscore their vital contribution to advancements in remote sensing, automation, and safety, reflecting their integral role in the development of cutting-edge technologies*.[43][44][45]*

*I. Solid State*

Solid-state lidars are recognized for their innovative design, which eliminates moving parts by utilizing optical phased arrays or other mechanisms for beam steering, thereby enhancing their durability and reliability. This feature is particularly advantageous for applications in challenging environments or where maintenance access is limited, such as in autonomous vehicles and drones. By directing laser pulses electronically, solid-state lidars can rapidly scan their surroundings without the mechanical wear and tear associated with traditional spinning lidar systems. The compactness and robustness of solid-state lidars make them well-suited for integration into a variety of platforms, demanding less space and offering greater resistance to vibrations and impacts. Although the field of view and resolution offered by early solid-state lidars were less than those of their mechanical counterparts, advancements in technology continue to bridge this gap, improving their performance and applicability. The development of solid-state lidar technology is driven by the need for efficient, reliable, and cost-effective solutions for precise environmental sensing and mapping, underscoring its significance in the progression of autonomous navigation and safety systems.[43]

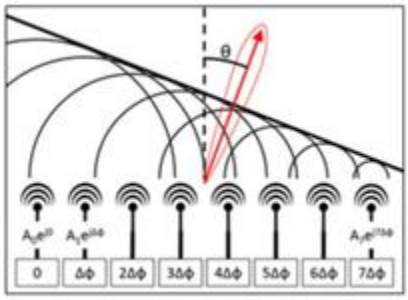



Figure 2.15: Solid State Lidar

*II. Flash*

Flash lidars are characterized by their method of illuminating an entire scene with a single, broad pulse of light, rather than scanning the environment point by point. This approach enables the simultaneous capture of distance information across the entire field of view, facilitating rapid image acquisition and reducing the complexity associated with moving parts. The absence

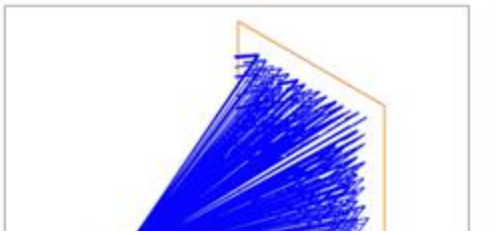


Figure 2.16: Flash lidar

of mechanical components not only contributes to the robustness and compactness of flash lidar systems but also enhances their reliability and longevity, making them particularly suitable for applications requiring fast, real-time 3D imaging, such as in some autonomous vehicle systems, industrial applications, and certain types of robotics. Despite traditionally facing challenges in range and resolution when compared to more conventional scanning lidar systems, advancements in sensor technology and signal processing continue to improve the capabilities of flash lidars. Their ability to provide instant environmental mapping is invaluable in scenarios where speed and responsiveness are critical, highlighting the important role flash lidars play in the evolution of sensing technology for autonomous navigation and safety applications.[44]

*III. Hybrid*

Hybrid lidars integrate the functionalities of both mechanical scanning and solid-state technologies. They are designed to capitalize on the strengths of each approach to offer a comprehensive sensing solution. By combining the wide field of view provided by mechanical components with the reliability and compactness of solid-state beam steering, hybrid lidars can deliver detailed environmental scans with high resolution and accuracy. This fusion allows for enhanced performance in terms of coverage area and detail, making them adaptable to a variety of applications, from autonomous driving to advanced security systems. The incorporation of mechanical elements enables a broader scanning range, while the solid-state aspects contribute to the system's durability and response speed. As a result, hybrid lidars present a balanced option for users requiring both extensive scanning capabilities and the robustness associated with solid-state designs. The development of hybrid lidar technology reflects the ongoing efforts to optimize remote sensing tools, ensuring they meet the evolving demands of precision, efficiency, and resilience in complex operational environments.[45]

*Table 2.7: Lidar Type Comparison*

| Component | Pros and Cons | Chosen |
|---|---|---|
| Solid- State | Crop Prose easy integration coins: view at size resolution can be limited compared to other types. | No |
| Flash | Pros: Capable of instantaneous full-field measurements, no moving parts for increased reliability. Cons: Limited range and detail resolution compared to scanning Lidars. | Yes |
| Hybrid | Pros: Combines wide scanning range with solid-state reliability, offering detailed environmental data. Cons: Can be complex and expensive, incorporating moving parts. | No |

*Section 5: Conclusion*

The theoretical foundation outlined in this section provides a comprehensive understanding of the principles, technologies, and methodologies essential for the development of an Autonomous Maze Exploration and Optimal Path Planning Robot. The discussion covered the purpose and significance of the project, elaborating on how it aims to advance the field of autonomous systems by addressing challenges in efficient navigation and mapping. Through the exploration of software frameworks, particularly

ROS, and simulation environments such as RVIZ and Gazebo, this section underscores the critical role of these tools in facilitating the design, visualization, and testing of robotic systems in virtual environments before physical implementation.

The algorithms section provided insight into the various computational strategies that can be employed to enable the robot to intelligently navigate and make decisions in dynamic environments. The emphasis on machine learning, especially reinforcement learning algorithms such as TD3, highlighted the project's innovative approach to enabling the robot to learn and improve its path planning capabilities through interaction with the environment.

Furthermore, the hardware requirements discussion clarified the essential components needed to build the physical model of the robot. This includes the integration of sensors, motors, and computational units such as the Raspberry Pi, each serving a distinct purpose to ensure the robot's functionality and its ability to autonomously navigate and map mazes.

In summary, the theoretical background provided in this section lays a solid foundation for the practical implementation of the project. It not only details the technological and computational frameworks that will be utilized but also rationalizes their selection in the context of the project's objectives. This section has set the stage for the subsequent phases of design, development, and testing, ensuring that the project is grounded in robust theoretical principles that guide its execution towards achieving efficient, autonomous maze exploration and optimal path planning.

### *Section 6: Upcoming*

The upcoming Chapter 3 is to transition from the theoretical framework established in previous sections to the practical execution phase of our project. This chapter will delve into the implementation process of creating a virtual model of the Autonomous Maze Exploration and Optimal Path Planning Robot, showcasing the methodology and technical steps involved in bringing the theoretical aspects of the project to life within a simulated environment.

Key areas of chapter include:

- Implementation Strategy: Detailed description of how the virtual model is constructed, including the translation of the robot's design into a digital format that can be manipulated and tested in simulation software.
- Simulation Methodology: Exploration of the tools and techniques used to simulate the robot's environment, such as the construction of virtual mazes and the integration of sensor data to replicate real-world conditions. This section will highlight the use of Gazebo and RVIZ for realistic simulation and visualization of the robot's interactions within the maze.
- Algorithm Testing and Optimization: Discussion on how the virtual model facilitates the testing and refinement of navigation and path planning algorithms. Emphasis will be placed on the role of simulation in evaluating the robot's performance, enabling adjustments and optimizations before physical prototype development.
- Data Collection and Analysis: Explanation of methods used to gather and analyze data from the virtual model. This includes how the robot's sensor inputs, movement decisions, and path efficiency are captured and utilized to improve the algorithm's effectiveness.
- Integration with Hardware Components: Outline of the planned approach to transition from virtual modeling to the creation of a hardware prototype. This will cover how insights gained from simulations inform the assembly and programming of the physical robot.

Chapter 3 promises to offer a comprehensive guide on the practical steps taken to validate the design and functionality of the Autonomous Maze Exploration and Optimal Path Planning Robot through simulation. By providing a bridge between theoretical concepts and their application, this chapter aims to demonstrate the iterative process of design, simulation, evaluation, and refinement that is central to the development of complex robotic systems. Through this approach, we anticipate not only to prove the

feasibility of our design but also to enhance its efficiency and reliability in solving the challenges associated with autonomous maze exploration and optimal path planning.

# CHAPTER 3

# VIRTUAL MODEL

Notes

1. Implementation

2. Simulation

    A. Actor Policy Actor: The Decision Maker

    B. Critic Policy Actor: The Evaluator

    C. Differences between Actor and Critic

3. TD3

4. Gazebo

5. Prototype

    A. Interaction of Components in the Hardware Model

    B. Interaction of Hardware Model and Software Model

6. Prototype Model

    A. Sensor Integration and Data Acquisition

        I. Saving Sensor Data

B. Raspberry Pi: The Core
C. Motor Controller and Motor Driver: Movement Commands
D. Motors: The Actuation System
E. Complete Component Interaction

7.Conclusion

# CHAPTER 3
# VIRTUAL MODEL

The endeavor to cultivate an autonomous maze exploration robot propels the fusion of Deep Reinforcement Learning (DRL) algorithms with the Robotics Operating System (ROS2) framework and the Gazebo simulation environment. This integration encourages a structured approach to training a virtual robot model, employing a training node script that facilitates the interaction between computational virtual intelligence and robotic perception and actuation mechanisms. This section will outline the operational dynamics of the training node, explaining its role in navigating the robot through virtual mazes with increasing complexity.

## *Section 1: Implementation*

The integration of deep reinforcement learning (DRL) with the Robotics Operating System (ROS2) and Gazebo simulation for autonomous navigation tasks requires a comprehensive blend of knowledge/experience in several advanced domains. Primarily, an adept understanding of DRL principles, specifically the Twin Delayed DDPG (TD3) algorithm, is crucial. This encompasses not only the theoretical basis of actor-critic based models and experience replay mechanisms but also practical skills in implementing these models using computational frameworks like PyTorch. Such knowledge is essential for constructing and refining the intelligent agents that dictate the robot's navigational decisions.

Moreover, proficiency in ROS2 is indispensable for smooth and seamless interaction between the robot's sensory inputs and actuator outputs within a simulated environment. This requires familiarity with the creation and management of nodes, topics, and services in ROS2, ensuring that data flow and robotic control strategies are efficiently executed. Additionally, an understanding of the Gazebo simulation

environment is required to create realistic training scenarios for the robot, involving the manipulation of virtual terrains and obstacles to challenge and hone the robot's navigational abilities.

In essence, leveraging the training node code for autonomous navigation within complex mazes demands a multidisciplinary skill set that spans DRL algorithms, robotic systems integration via ROS2, and environmental simulation through Gazebo. Mastery of these domains not only enables the effective use and modification of the training node but also propels further innovations in robotic autonomy. This interdisciplinary approach underscores the evolving landscape of robotics, where the synthesis of machine learning and robotic control opens new avenues for exploration and application in autonomous systems.

*Section 2: Simulation*

There are many components that are required for the successful implementation of deep learning/machine learning networks regardless of the task in which they are supposed to carry out. As stated previously knowledge in deep learning networks, advanced applications of PyTorch, ROS2 and general programming knowledge are required. Here in this section of this chapter we will help break down the most important sections of the code into understandable chunks. First, we will be starting with the Gazebo environment.

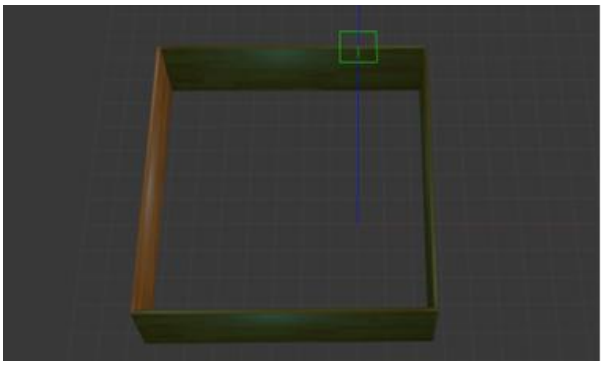


Figure 3.1: Initialize Maze

Without an environment as a foundation for the virtual testing and training to take place then there is no plausible way to proceed with this project. Below is a figure of the of the virtual environment. First figure on the left displays the first step of the maze

creation process while the figure on the right is the completed maze in Gazebo. From here we can then move on to the training node, analysis.

Within the architecture of the training node, the Actor network emerges as a pivotal element, tasked with the generation of actions based on the current state of the environment. The Actor is defined in Figure C1.

This neural network is composed of three linear layers, interspersed with ReLU activations to introduce non-linearity, culminating in a tanh activation function to produce actions within a bounded range. The Actor's role is to map observations (or states) from the environment directly to actions, facilitating continuous control over the robot's movements. This direct mapping enables the robot to learn complex behaviors for navigating through mazes, adapting its strategy to maximize efficiency and safety.

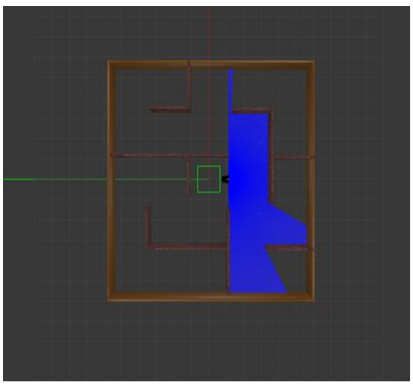


Figure 3.2: Complete Maze Model

In the realm of autonomous navigation, the orchestration between Actor and Critic policy actors underpins the operational efficiency and decision-making acuity of the system. Grounded in the framework of Deep Reinforcement Learning (DRL), specifically the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, these components play pivotal roles in refining the navigational intelligence of autonomous robots. This section delves into the nuanced functionalities and synergistic interactions of the Actor and Critic within the context of autonomous maze exploration and optimal path planning.

A. Actor Policy Actor: The Decision-Maker

The Actor component, a neural network model, functions as the decision-maker of the system. It is tasked with mapping the observed state of the environment directly to

actions, guiding the robot's movements through the maze. Architecturally, the Actor comprises multiple layers of neurons, each layer designed to process and transform the input state into a higher-level representation. The culmination of this process is the generation of action outputs, constrained within a bounded range through the application of a hyperbolic tangent (tanh) activation function at the output layer. This ensures that the actions produced by the Actor are viable within the physical capabilities of the robot. The Actor's efficacy lies in its ability to learn the most effective actions to take in any given state, a capability honed through continuous interaction with the environment and subsequent training iterations.

For the case of Figure 18, this actor class is responsible for initializing the actor network as well as determining what it generates. Here The Actor network generates actions for a given state, guiding the agent's behavior. The first layer processes the state input, producing an 800-dimensional vector, which then feeds into a second layer, producing a 600-dimensional vector. This is followed by a final linear layer that combines with the action dimension. The final layer passes through a Tanh function, bounding the output to ensure the action values lie within a valid range. The forward function specifies how the Actor network's layers interact. The state input is processed through each layer in sequence, with batch normalization applied after the first two layers for stability. The final output is transformed by a Tanh function to yield an action.

```
class Actor(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(Actor, self).__init__()
        self.layer_1 = nn.Linear(state_dim, 800)
        self.bn1 = nn.BatchNorm1d(800)
        self.layer_2 = nn.Linear(800, 600)
        self.bn2 = nn.BatchNorm1d(600)
        self.layer_3 = nn.Linear(600, action_dim)
        self.tanh = nn.Tanh()

    def forward(self, s):
        if s.size(0) > 1:
            s = F.relu(self.bn1(self.layer_1(s)))
            s = F.relu(self.bn2(self.layer_2(s)))
        else:
            s = F.relu(self.layer_1(s))
            s = F.relu(self.layer_2(s))
        a = self.tanh(self.layer_3(s))
        return a
```

Figure 3.3: Actor Network Policy

B. Critic Policy Actor: The Evaluator

```python
class Critic(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(Critic, self).__init__()

        # First Critic Network
        self.layer_1 = nn.Linear(state_dim, 800)
        self.bn1 = nn.BatchNorm1d(800)
        self.layer_2_s = nn.Linear(800, 600)
        self.layer_2_a = nn.Linear(action_dim, 600)
        self.bn2 = nn.BatchNorm1d(600)
        self.layer_3 = nn.Linear(600, 1)

        # Second Critic Network
        self.layer_4 = nn.Linear(state_dim, 800)
        self.bn4 = nn.BatchNorm1d(800)
        self.layer_5_s = nn.Linear(800, 600)
        self.layer_5_a = nn.Linear(action_dim, 600)
        self.bn5 = nn.BatchNorm1d(600)
        self.layer_6 = nn.Linear(600, 1)

    def forward(self, s, a):
        # Applying batch normalization only if batch size > 1 for the first critic network
        if s.size(0) > 1:
            s1 = F.relu(self.bn1(self.layer_1(s)))
            s1 = F.relu(self.bn2(self.layer_2_s(s1) + self.layer_2_a(a)))
        else:
            s1 = F.relu(self.layer_1(s))
            s1 = F.relu(self.layer_2_s(s1) + self.layer_2_a(a))
        q1 = self.layer_3(s1)

        # Applying batch normalization only if batch size > 1 for the second critic network
        if s.size(0) > 1:
            s2 = F.relu(self.bn4(self.layer_4(s)))
            s2 = F.relu(self.bn5(self.layer_5_s(s2) + self.layer_5_a(a)))
        else:
            s2 = F.relu(self.layer_4(s))
            s2 = F.relu(self.layer_5_s(s2) + self.layer_5_a(a))
        q2 = self.layer_6(s2)
```

Figure 3.4: Critic Network Policy

In contrast, the Critic serves as the system's evaluator, assessing the quality of actions proposed by the Actor by estimating the future rewards these actions are likely to yield. The Critic's architecture mirrors that of the Actor but with a critical distinction: it accepts both the state and the action as inputs, merging these to provide a value estimation. This estimation reflects the expected cumulative reward, guiding the Actor towards strategies that maximize the system's objectives. The dual-network configuration of the Critic in the TD3 algorithm—comprising two separate Critic networks—aims to mitigate the overestimation of future rewards, a common challenge in DRL that can lead to suboptimal policy convergence. By selecting the lesser of the two value estimations for policy updates, the system ensures a conservative approach to value approximation, enhancing the learning stability.

C. Differences Between Actor and Critic

The interplay between the Actor and Critic is characterized by a continuous feedback loop. As the Actor proposes actions based on the current state, the Critic evaluates these actions, providing feedback that informs the subsequent adjustments to the Actor's policy. This dynamic is further refined through the implementation of target networks—a concept wherein slower-updating copies of the Actor and Critic networks are employed to calculate target values, reducing the temporal correlations in updates, and fostering stable learning progression. The synergy between the Actor and Critic, facilitated by mechanisms such as soft target updates and policy noise, underscores the adaptability and robustness of the autonomous navigation system, enabling the robot to navigate complex environments with increasing competence over time.

To sum it up, the Actor and Critic policy actors constitute the core of the autonomous navigation system's learning mechanism, embodying the principles of exploration and exploitation in the quest for optimal navigation strategies. Through their intricate design and interdependent functionalities, these components drive the advancement of autonomous systems, enabling nuanced understanding and interaction with the surrounding environment.

*Section 3: TD3*

Next is the TD3 function that is responsible for the learning of the system. The policies that were previously talked about merely sort through the data that is collected from the lidar. Next is the TD3 function that is responsible for the learning of the system. The policies that were previously talked about merely sort through the data that is collected from the lidar.

Next is the TD3 function that is responsible for the learning of the system. The policies that were previously talked about merely sort through the data that is collected from the lidar. The Twin Delayed Deep Deterministic Policy Gradient (TD3) framework underpins the learning mechanism, orchestrating the update cycle for both the Actor and the Critic networks. Highlighted by its dual Critic designTD3 mitigates the overestimation bias inherent in Q-learning algorithms, enhancing the stability of the learning process.

```
class td3(object):
    def __init__(self, state_dim, action_dim, max_action):
        self.actor = Actor(state_dim, action_dim).to(device)
        ...
        self.critic = Critic(state_dim, action_dim).to(device)
        ...

    def train(self, replay_buffer, iterations, ...):
        for it in range(iterations):
            ...
            # Update the policy by maximizing the Q-value
            actor_loss = -self.critic.Q1(state, self.actor(state)).mean()
            ...
            # Delayed policy updates
            if it % policy_freq == 0:
                self.actor_optimizer.zero_grad()
                actor_loss.backward()
                self.actor_optimizer.step()
                ...
```

Figure 3.5: Initializing the TD3 class.

The TD3 algorithm, an enhancement of the Deep Deterministic Policy Gradient (DDPG) method, introduces several key innovations designed to improve learning stability and performance. At its core, the TD3 object function integrates a dual Critic mechanism and a policy update delay strategy, addressing the challenges of overestimation bias and excessive policy variance. The TD3 object encapsulates this sophisticated algorithmic structure, embodying the logic required to iteratively refine the decision-making policies of the autonomous robot.

The operational essence of the TD3 object function is encapsulated in its training loop, where it meticulously manages the interplay between exploration and exploitation. By

```
state = torch.Tensor(batch_states).to(device)
next_state = torch.Tensor(batch_next_states).to(device)
action = torch.Tensor(batch_actions).to(device)
reward = torch.Tensor(batch_rewards).to(device)
done = torch.Tensor(batch_dones).to(device)
```

Figure 3.6: Learning function in TD3 class

sampling experiences from a replay buffer, the TD3 object leverages historical interactions with the environment to inform the policy update process. This process is performed in figure (blank), in the figure the trained model is being improved by tau (parameter that is responsible for the amount of learning per iteration). This retrospective analysis allows for a nuanced adjustment of the Actor's action-generating policies, guided by the evaluative insights provided by the dual Critic networks. The introduction of target policy smoothing, achieved through the injection of noise into the action selection process, further enriches the exploration capabilities of the system, enabling the discovery of novel strategies and pathways.

Key to the TD3 object's effectiveness is its implementation of techniques aimed at enhancing learning stability. The policy update delay mechanism, wherein the Actor's policies are updated less frequently than the Critic's value estimations, serves to decouple the dependencies between policy and value updates, mitigating the risk of premature

```
for param, target_param in zip(
    self.actor.parameters(), self.actor_target.parameters()
):
    target_param.data.copy_(
        tau * param.data + (1 - tau) * target_param.data
    )
```

Figure 3.7: TD3 class explore method.

convergence to suboptimal policies. Furthermore, the use of soft target updates for the Actor and Critic target networks ensures a gradual integration of new knowledge, smoothing the learning trajectory and fostering a stable convergence towards optimal navigation behaviors.

The TD3 class is also responsible for training the networks and recording Avg. Q and Max Q data from the Critic networks. This data is used to evaluate the machine learning model's performance, providing key insights into its navigation strategies, and guiding further refinements.

*Section 4: Gazebo*

Lastly is the Gazebo Environment class. This class serves as the interface between the ROS2 framework and the Gazebo simulation, allowing for direct manipulation of the robot and the environment. Through this interface, actions determined by the TD3 algorithm are translated into movement commands, and sensory feedback is relayed back to the learning algorithm.

```
class GazeboEnv(Node):
    ...
    def step(self, action):
        vel_cmd = Twist()
        vel_cmd.linear.x = float(action[0])
        vel_cmd.angular.z = float(action[1])
        self.vel_pub.publish(vel_cmd)
        ...
```

Figure 3.8: Gazebo step function

In the realm of autonomous robot navigation, the GazeboEnv class represents a critical interface, seamlessly bridging the gap between the simulated environment provided

```
# Detect if the goal has been reached and give a large positive reward
if distance < GOAL_REACHED_DIST:
    env.get_logger().info("GOAL is reached!")
    target = True
    done = True

robot_state = [distance, theta, action[0], action[1]]
state = np.append(laser_state, robot_state)
reward = self.get_reward(target, collision, action, min_laser)
return state, reward, done, target
```

Figure 3.9: Determine if goal is reached

by Gazebo and the learning mechanisms powered by deep reinforcement learning algorithms such as TD3. This section explores the pivotal role of the GazeboEnv class in facilitating real-time interaction with the virtual world, enabling the robot to learn and adapt its navigation strategies through iterative experimentation and feedback.

The GazeboEnv class functions as the conduit through which sensory data is collected and actions are executed within the Gazebo simulation environment. By inheriting from the

```
class GazeboEnv(Node):
    def step(self, action):

        # read velodyne laser state
        done, collision, min_laser = self.observe_collision(velodyne_data)
        v_state = []
        v_state[:] = velodyne_data[:]
        laser_state = [v_state]

        # Calculate robot heading from odometry data
        self.odom_x = last_odom.pose.pose.position.x
        self.odom_y = last_odom.pose.pose.position.y
        quaternion = Quaternion(
            last_odom.pose.pose.orientation.w,
            last_odom.pose.pose.orientation.x,
            last_odom.pose.pose.orientation.y,
            last_odom.pose.pose.orientation.z,
        )
        euler = quaternion.to_euler(degrees=False)
        angle = round(euler[2], 4)
```

Figure 3.10: Position Data Collection

ROS2 Node class, it leverages the ROS2 framework to subscribe to sensory inputs like odometry and LiDAR, translating these inputs into states that inform the robot's decision-making process. Conversely, it acts upon the decisions made by the learning algorithm, translating them into movement commands that are executed within the simulation, thus allowing the robot to interact dynamically with its environment. Additionally, the class performs small calculations to determine the robot's position relative to its goal, guiding its navigation strategy as seen in Figure (blank), where the robot's current position with reference to the maze is taken as well as its global coordinates in the world represented as w, x, y, z. Next in Figure (blank), some more calculations are performed to determine the robot's magnitude distance from the goal from the current heading of the robot, to the

One of the key strengths of the GazeboEnv class lies in its deep integration with ROS2 and Gazebo, enabling sophisticated control over the simulation environment. This includes the ability to pause, un-pause, and reset the simulation, facilitating episodic learning where the robot's performance can be evaluated across different runs within a consistent framework. Additionally, the class provides mechanisms for dynamically adjusting the simulation's state, such as repositioning the robot or modifying the environment in response to the learning process, thus enriching the diversity of training scenarios.

The GazeboEnv class significantly enhances the learning process by offering a rich, interactive platform where theoretical navigation strategies can be tested and refined in complex, realistic scenarios. Through continuous interaction with the environment, the robot can gather valuable data on the outcomes of its actions, feeding this data back into the learning algorithm to refine its policy. This iterative cycle of action and feedback is instrumental in enabling the robot to develop robust, adaptive navigation strategies that can handle the unpredictability and complexity of real-world environments.

The GazeboEnv class stands as a testament to the synergistic potential of simulation technologies and deep reinforcement learning in advancing the field of autonomous navigation. By providing a realistic, controllable, and versatile simulation environment, it allows for the rigorous training and evaluation of autonomous robots, pushing the boundaries of what these systems can achieve. Through its contributions, the GazeboEnv class not only facilitates the development of more intelligent and adaptable navigation systems, but also underscores the importance of simulation as a tool for innovation in robotics.

## *Section 5: Prototype*

### A. Interaction of Components in the Hardware Model

In the development of the Autonomous Maze Exploration and Optimal Path Planning Robot, the harmonious interaction between its hardware components is crucial for its successful operation.

This section outlines how the key components—sensors, motors, motor controller and driver, and the Raspberry Pi—coordinate to enable the robot to navigate and solve mazes autonomously. There also must be a selection of the structure of the robot's body. Provided this we investigated multiple models for reference. These models included the Jetbot from Nvidia, and 2 other concepts that were created by us. The components required from each model was that it have 2 wheels, a third dummy/caster wheel, be large enough for the components to rest in or on the chassis. And be compact enough to fit in a maze no larger than 5ft x ft.

Figure 3.11: Jetbot Model

B. Interaction of Hardware Model and Software Model

The fabrication of the virtual model, there were 3 primary issues that were plaguing the model. Issues saving the lidar data. Issues synchronizing up the data from the virtual model to the physical model. These issues are addressed and solved in this chapter, beginning with the syncing up the data from the virtual model to the prototype and vice versa. We will now discuss multiple methods about wireless communication, and whether they are reasonable for this application.

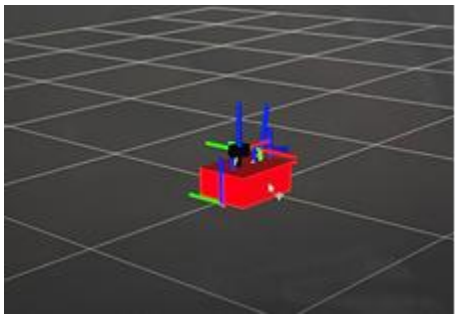


Figure 3.12: Rviz model

With the further development of wireless communication protocols, including SSH (Secure Shell) and other wireless modalities, has markedly advanced the functional capacity of autonomous robots. These wireless protocols are predicated on the principle of enhancing operational agility, allowing robots to traverse and perform within varied environments unencumbered by the physical limitations imposed by wired connections. Secure and reliable data transmission is rendered possible through wireless protocols such as SSH, which employs encryption to safeguard the data against potential security breaches. This aspect is of crucial importance

when robots are deployed in environments where the risk of interception or unauthorized access is non-trivial.

Wireless communication protocols, such as SSH and others, play a pivotal role in augmenting the scalability of autonomous robotic systems. These protocols facilitate the expansion of robotic operations without the need for extensive physical infrastructure, such as wiring, which can be both cost-prohibitive and logistically complex to implement on a larger scale. Currently the most popular protocols are SSH, Wi-Fi connectivity, Connection via Bluetooth, Zigbee and LTE/5G wide-area-connection.

SSH, known for its secure data transmission capabilities, offers a scalable solution for managing a fleet of robots. It allows for remote command execution, software updates, and system monitoring across multiple units simultaneously, all while maintaining high-security standards. With SSH, expanding the network of robots is as simple as adding new units to the SSH configuration, without the need for additional physical setup.

Wi-Fi connectivity, with its high data transfer rates, enables robots to communicate with central servers and each other over local networks. This is crucial when scaling operations within a facility or across multiple locations. Wi-Fi's robust infrastructure and widespread availability make it a scalable choice for indoor robotic systems that require real-time data exchange and coordination.

Figure 3.13: Virtual Model

Bluetooth technology, particularly suitable for smaller or less complex robotic systems, supports scalability in scenarios where low power consumption and short-range communication suffice. It is ideal for swarm robotics, where multiple small robots operate in proximity and coordinate their tasks collectively.

Zigbee shines in its application to sensor networks and low-data rate requirements. When scaling a system that relies on a multitude of sensors, Zigbee offers a network

solution that conserves power and reduces operational costs. It is especially useful in large-scale deployments where the sensor data load is moderate and consistent.

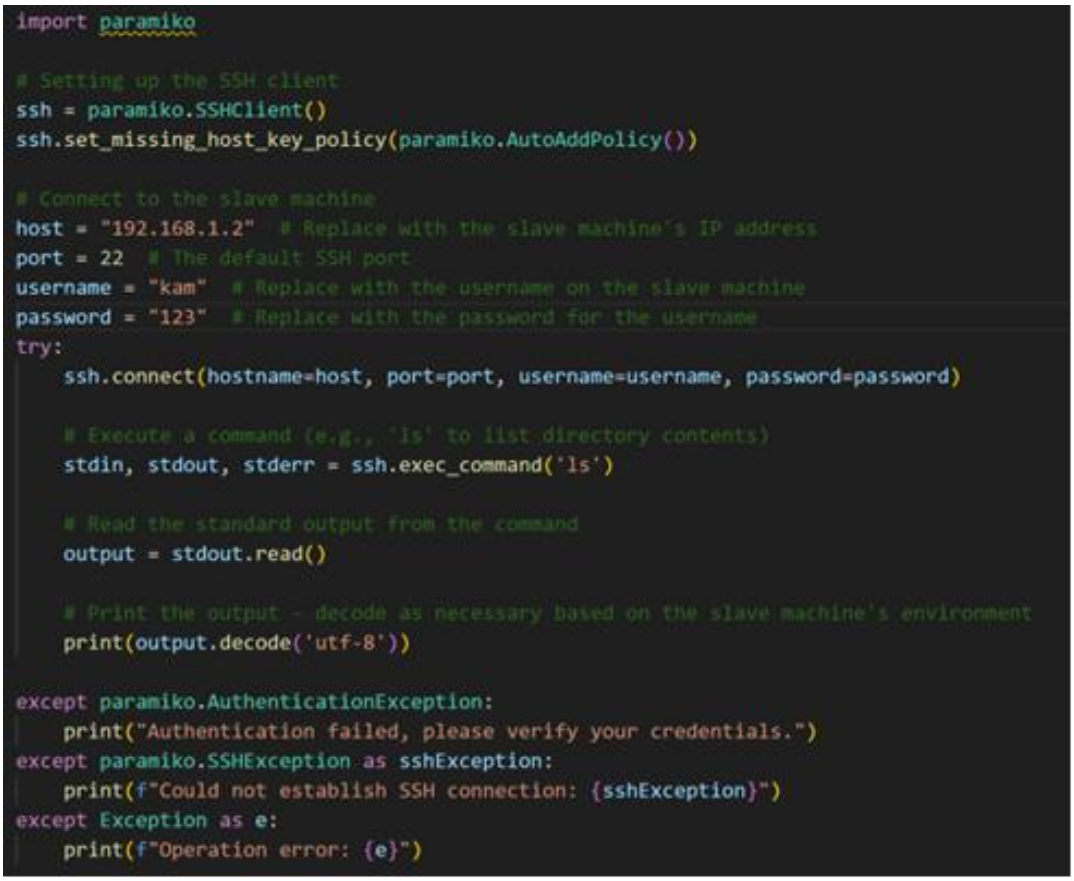
The advent of LTE and 5G technologies has revolutionized scalability for outdoor or geographically dispersed robotic systems. These cellular networks enable robots to operate over vast distances while maintaining constant communication with control centers. The high bandwidth of 5G supports the deployment of robots with advanced data needs, such as high-definition video streaming for surveillance or inspection tasks.

Each wireless communication protocol operates based on specific principles:

- SSH functions through an encrypted channel, using public-key cryptography for secure remote login, command execution, and file transfer.
- Wi-Fi operates on IEEE 802.11 standards, using a router as a central hub to connect devices over radio waves.
- Bluetooth uses short-wavelength UHF radio waves and a small network called a piconet to connect devices within a close range.
- Zigbee creates personal area networks with low-powered digital radios, relying on a mesh network topology for data transmission.
- LTE/5G utilizes cellular towers and a complex network infrastructure to provide wide-area networking.

The choice of wireless communication protocol is integral to the scalability of autonomous robotic systems. Each protocol offers different operational functions that cater to specific scalability needs, whether it's the secure management of multiple robots with SSH, the local network expansion with Wi-Fi, the energy-efficient scaling with Bluetooth and Zigbee, or the wide-area network capabilities with LTE/5G. As robotic systems grow in complexity and number, the ability to maintain secure, reliable, and efficient communication becomes increasingly crucial, positioning wireless communication as a cornerstone of scalable robotic system design. Given the current need of the system the more useful mode of wireless communication is SSH protocol.

The Paramiko library is imported to facilitate SSH communication within the Python script. An instance of the SSHClient class is created from the Paramiko library, representing an SSH client capable of connecting to remote servers and executing commands. The policy for handling missing host keys is set to automatically add unknown



```
import paramiko

# Setting up the SSH client
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# Connect to the slave machine
host = "192.168.1.2"  # Replace with the slave machine's IP address
port = 22  # The default SSH port
username = "kam"  # Replace with the username on the slave machine
password = "123"  # Replace with the password for the username
try:
    ssh.connect(hostname=host, port=port, username=username, password=password)

    # Execute a command (e.g., 'ls' to list directory contents)
    stdin, stdout, stderr = ssh.exec_command('ls')

    # Read the standard output from the command
    output = stdout.read()

    # Print the output - decode as necessary based on the slave machine's environment
    print(output.decode('utf-8'))

except paramiko.AuthenticationException:
    print("Authentication failed, please verify your credentials.")
except paramiko.SSHException as sshException:
    print(f"Could not establish SSH connection: {sshException}")
except Exception as e:
    print(f"Operation error: {e}")
```

Figure 3.14: SSH network method

host keys to the client's known hosts file, ensuring seamless connections without manual confirmation. The IP address, port number, username, and password for authentication on the remote machine are defined. Upon attempting to establish an SSH connection using the specified credentials, any exceptions encountered during the process are caught and handled accordingly. Following a successful connection, a command is executed on the remote machine, with the standard output stream decoded and printed to the console. Now that a secure connection has been established it is now possible to sync the movements between the prototype model and the virtual model. With the secure connection made,

instead of the lidar data being generated virtually, the data can then be directly transferred from the physical lidar on the robot given this data the host computer can decide based on the trained virtual model. Then to have the prototype model function as the virtual model does you also must publish the cmd_vel/ topic data to the prototype. The cmd_vel/ topic data is simply the direction that the wheels should be moving at any given point. This topic is continuously updated to keep the robot moving in the correct direction while also avoiding obstacles.

*Section 6: Prototype model*

| Parts List | |
|---|---|
| A | Mounting Bracket for Lidar |
| B | Motor Placement & Wheel Location |
| C | Robot Chasis(Body) |
| D | Caster Ball |
| E | Detachable Cover |

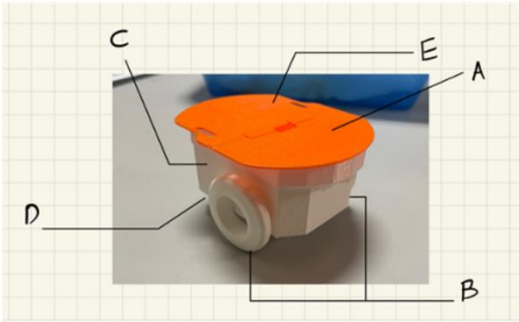



Figure 3.15: Parts List and Model

Above is a 3d printed figure of the robot model. The model consists of 6 sections consisting of a lidar mount, wheels, chassis body, caster ball and the detachable cover to access the components inside of the chassis body. The components that will be inside of this model are the battery, motor driver, motor controller, 2 serial encoded motors, and the RPLidar A1.

A. Sensor Integration and Data Acquisition

Figure 3.16: Lidar

The centerpiece of the robot's sensory system is the LiDAR sensor, which is responsible for environmental perception and obstacle detection. The LiDAR continuously scans the surroundings, generating a point cloud that represents the maze's layout and any obstacles within it. This data is relayed to the Raspberry Pi, where it is processed to create a map of the environment and identify viable paths. The effectiveness of the robot's navigation and path planning is directly tied to the accuracy and reliability of the data captured by the LiDAR sensor.

*I. Saving Sensor Data*

```
def save(self, filename, directory):
    torch.save(self.actor.state_dict(), "%s/%s_actor.pth" % (directory, filename))
    torch.save(self.critic.state_dict(), "%s/%s_critic.pth" % (directory, filename))
```

Figure 3.17: Save trained model.

Here is the method in which the lidar data is saved to the system. The current implementation method that we are using to view and analyze the map data is to first pass the data through first Rviz so that the data can be viewed, then that data is passed to Gazebo. However, before which the data is saved to a .pcd file that is used by the td3 algorithm and the actor and critic policies. Below is a picture of the lidar data.

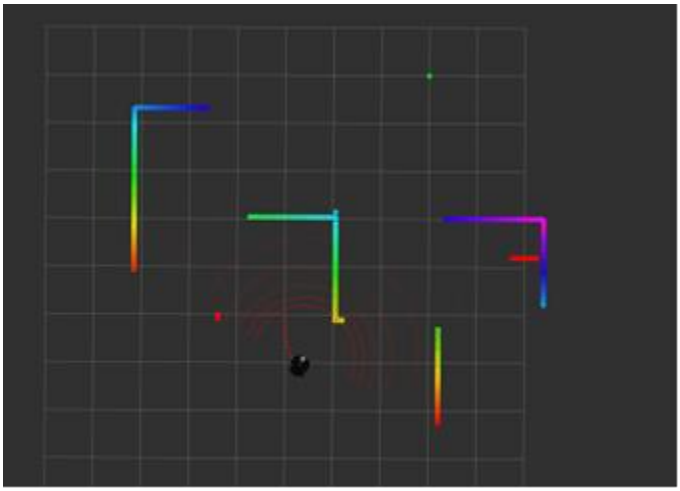


Figure 3.18

B. Raspberry Pi: The Core

The Raspberry Pi serves as the brain of the robot, processing input from the LiDAR sensor to make real-time navigation decisions. It runs the path planning algorithm, which calculates the optimal route through the maze based on the map generated from the LiDAR data. The Raspberry Pi also orchestrates the overall operation of the robot, ensuring seamless communication between the sensors, motors, and other components. Through its GPIO (General Purpose Input Output) pins, the Raspberry Pi sends control signals to the motor controller, dictating the speed and direction of the motors based on the chosen path.

Figure 3.19: Raspberry pi

Figure 3.20: Backup motor driver

C. Motor Controller and Motor Driver: Executing Movement Commands

The motor controller receives commands from the Raspberry Pi and translates these into actionable signals for the motor driver. It determines the appropriate responses required to navigate the robot along the calculated path, adjusting for speed, direction, and obstacle avoidance maneuvers. The motor driver, in turn, supplies the necessary power to the motors, enabling them to execute the movement commands. This component is critical for

Figure 3.21: Motor controller/driver

modulating the motor's responses to achieve smooth and precise navigation through the maze.

D. Motors: The Actuation System

The motors are the final output stage of the robot's navigation system. Upon receiving power and control signals from the motor driver, they propel the robot forward, steer it, and adjust its speed as necessary to follow the optimal path through the maze. The motors must be responsive and accurately calibrated to ensure that the robot can navigate the complex terrain of the maze effectively.

E. Complete Component Interaction

To analyze and determine the component interaction between the selected hardware. Automation Studio serves as a comprehensive tool that allows for the simulation and validation of our system's hardware components. By modeling the system's architecture and individual components, we can confirm the suitability of our selected parts, ensuring they meet the project's requirements and specifications. This validation process includes simulating real-world scenarios, allowing for a thorough evaluation of each component's performance in diverse operating conditions, ensuring the overall robustness of our system.

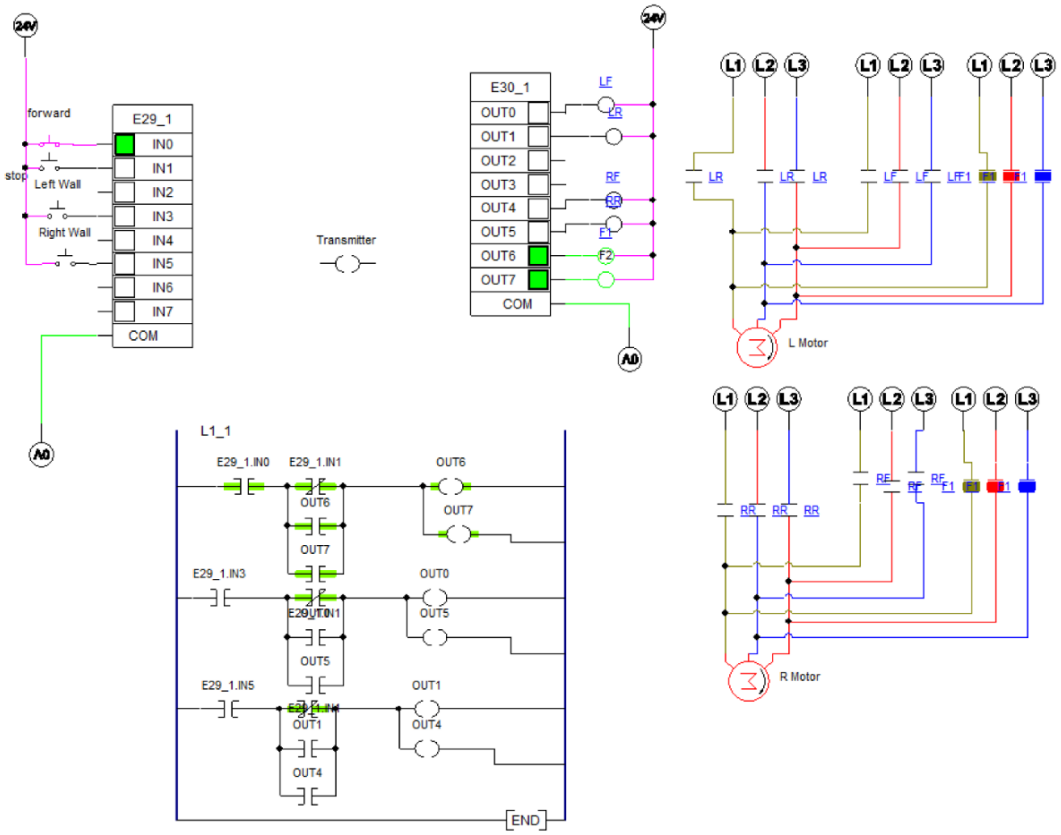



Figure 3.22: Autonomation studio PLC

In addition to confirming component selection, Automation Studio provides invaluable insights into the interplay between hardware components. By constructing a digital twin of our system, we can simulate the interactions between the various components, identifying potential bottlenecks, inefficiencies, or conflicts. This holistic view enables us to optimize the configuration, ensuring smooth operations and seamless communication between components, which is essential for developing a functional and reliable system.

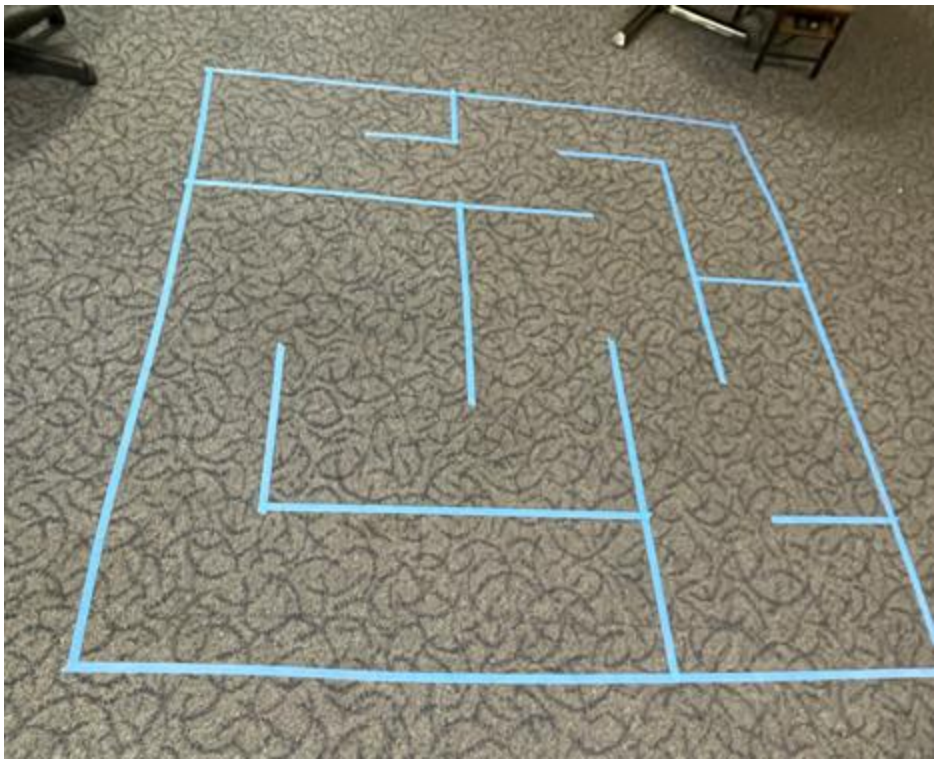


Figure 3.23: Maze outline for prototype

Automation Studio also offers a platform for preemptive troubleshooting, helping to identify potential issues before they manifest in the real world. This proactive approach reduces the risk of costly delays and failures during the implementation phase. By modeling the system's operation, we can refine the design, adjust configurations, and test new strategies in a virtual environment, ensuring that the final implementation performs as expected.

In summary, Automation Studio is a critical tool for validating and optimizing our hardware components. By providing a detailed simulation environment, it ensures that each component meets project requirements, interacts seamlessly with other parts, and operates efficiently in various scenarios. This contributes to the overall success of our system, enabling a smooth implementation and a reliable final product.

The successful operation of the hardware model depends on the seamless interaction between these components. The LiDAR sensor must accurately detect and map the

environment, the Raspberry Pi must efficiently process this data and generate effective navigation commands, and the motor controller and driver must precisely execute these commands to guide the motors. Each component plays a vital role, and their interactions are orchestrated through carefully designed software running on the Raspberry Pi. This software enables real-time data processing, decision-making, and control, ensuring that the robot can autonomously explore and navigate mazes with optimal efficiency.

This hardware model's design emphasizes modularity, allowing for easy adjustments and upgrades to each component. By ensuring each part communicates effectively with the others, the model lays a solid foundation for the development of advanced autonomous navigation systems capable of tackling more complex environments and tasks in the future.

## *Section 7: Conclusion*

Autonomous systems are the driving force behind engineering. Through our research with this capstone project we have explored many different facets including Machine Learning, the history of autonomous systems and different algorithms used to program these robots. We explored the notions needed to improve above optimal path plans and determined the best ways to improve upon these robotic systems.

When looking into machine learning algorithms we have several efforts that already exist including SLAM, Dijkstra, A*, D* and RRT. We used a combination of the Dijkstra and RRT to create our own method to traverse the maze. Combining the sensor technologies and our combined algorithms we can improve upon the methods used to autonomously explore mazes.

# CHAPTER 4

## Design Procedure and Implementation

1. Physical Maze
2. Training process
3. Training Models
4. Prototype Testing
    A. Fabrication
    B. Procedure
5. Results
6. Conclusions
7. Recommendations

## Chapter 4
## Design Procedure and Implementation

The design procedure and implementation strategies for fabricating and training a physical maze-navigating robot are critical to its success in autonomous navigation. By carefully integrating selected hardware, conducting thorough simulation-based training, and adapting the robot to real-world conditions, the maze-navigating robot is equipped to handle the challenges presented by complex environments. This comprehensive approach ensures the development of a robust and capable autonomous system, poised to advance the field of robotics through innovative navigation solutions.

### *Section 1: Physical Maze*

This physical maze for this robot served as a guideline for the viewer to understand where the robot was in the maze. As for the initial goal of this project was to connect the virtual robot and the prototype model wirelessly. Once this done and a virtual action was made, the physical robot would just be sharing sensor data with the virtual model, while the virtual model will be sharing the individual wheel velocities with the prototype model. The advantage of this method of communication between the prototype model and the virtual robot is scalability. Given you want to have several maze solving robots, the trained neural network model can be located on a server and the robots can connect to the server via SSH protocol or some other means of connection (WIFI or Bluetooth) the server can perform all the calculation while the robot is only responsible for moving according to the transmitted velocities. If each robot in this scenario was to have an its own neural network on each system, this allows for the prototype robots to be more independent, meaning they no longer require a connection some way with a server or computer, but the robot will have to have more computational power to handle the installed neural network.

*Section 2: Training Process*

The TD3 algorithm's training regimen commences with the initialization of the actor and critic networks, which embody the policy and value functions, respectively. Each network is mirrored by a corresponding target network, initialized with identical parameters, as per the conventional practice in deep reinforcement learning to stabilize training. Upon this foundational setup, the agent engages with the simulated environment, executing actions and perceiving the consequent state transitions and rewards. Each interaction is meticulously recorded in a replay buffer, a repository designed to decouple the sequence of experiences, thereby enabling the agent to benefit from historical transitions, distilling insights from past explorations.

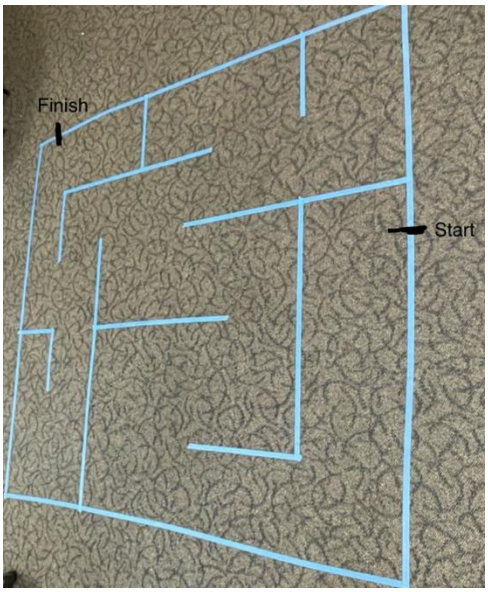



Figure 4.1: Maze outline for prototype (defined)

At the core of the training methodology is the iterative sampling of experience batches from the replay buffer. This reservoir of past interactions provides the substrate for the continuous refinement of the agent's decision-making apparatus. For each sampled batch, the algorithm applies noise to the policy's output—implementing a policy smoothing mechanism that deters premature convergence to suboptimal policies.

The algorithm subsequently employs the target critic networks to generate the value targets, integral to the temporal-difference learning framework. These targets provide the benchmarks for updating the critic networks, fostering their ability to appraise the selected actions' potential returns accurately.

Akin to a symphony's rhythm, updates to the actor network follow a deliberate cadence, occurring less frequently to circumvent the perils of overfitting. This actor update

adjusts the policy in the direction of actions that amplify the expected returns, as gauged by the critic networks.

Completing the cycle, the training process incorporates a measured updating of the target networks, with the tau parameter guiding the incremental assimilation of the primary networks' learned behaviors. This measured assimilation ensures the target networks remain steadfastly proximate to their evolving counterparts, yet sufficiently aloof to confer stability across training epochs.

Through an iterative confluence of experience replay, policy smoothing, value target generation, and network updates, the TD3 training process meticulously curates the policy's evolution. This progression is characterized by a gradual but relentless pursuit of an optimal policy capable of navigating the agent through the maze's challenges to the preordained goal.

As discussed in Chapters 2 and 3, the TD3 model's training process leverages a series of mechanisms to refine the actor and critic networks effectively. However, due to hardware limitations, we were limited to training the model for only 1.227 days, amounting to roughly 4,000 iterations. This constrained training duration led to lower convergence, affecting the model's performance, and limiting its ability to achieve a truly optimal policy.

### *Section 3: Trained Models*

The ideal model of the TD3 algorithm's best trained models resides in their duality—comprising both the actor and critic networks, each encapsulating distinct yet complementary aspects of the learned policy. The actor network embodies the policy model, also known as the decision-making entity, which maps environmental states to a specific action space. Through its intricacies, the network proffers the policy that guides the robotic agent's actions, calibrated to optimize navigational efficacy within the maze's confines.

Conversely, the critic networks serve as evaluators, providing an estimation of the expected cumulative rewards for state-action pairs. In essence, these networks encapsulate the value function that critiques the actions proposed by the actor. The TD3 algorithm

enhances the critic's precision by incorporating a pair of networks, which operate in tandem to mitigate the overestimation of future rewards—a common pitfall in value-based approaches.

The training process imbues these networks with a distilled synthesis of experiences accrued during the agent's interaction with the environment. During this process the td3 also adds a noise gradient to a copy of the trained network, the reason for this is part of the nature of the TD3 algorithm. It compares the trained model with that of the one with randomly generated noise values that fall within the preset parameters. Then the model with compare the random noise model with that of the trained model, this encourages exploring throughout the maze.

```
noise = torch.Tensor(batch_actions).data.normal_(0, policy_noise).to(device)
noise = noise.clamp(-noise_clip, noise_clip)
next_action = (next_action + noise).clamp(-self.max_action, self.max_action)
```

Figure 4.2: Noise policy implementation

The actor network's parameters are tuned to elicit a trajectory of actions leading to maximal rewards. Through backpropagation and policy gradient methods, the network's weights are adjusted to incrementally steer the policy towards more profitable actions, as adjudicated by the critic's evaluations.

The critic networks, through their learning process, assimilate a representation of the Q-function. They are trained on a temporal difference error signal that captures the discrepancy between predicted and actual rewards, subsequently refining the network's parameters to reduce this divergence. The robustness of the TD3 approach is augmented by using a twin network setup, where the lower of the two Q-value estimates is employed to temper updates, thereby fostering a more conservative and stable learning process.

```
current_Q1, current_Q2 = self.critic(state, action)

# Calculate the loss between the current Q value and the target Q value
loss = F.mse_loss(current_Q1, target_Q) + F.mse_loss(current_Q2, target_Q)
```

Figure 4.3: Loss determination method

Contained within the trained models are the weight matrices and bias vectors that constitute the parameters of the neural networks. These parameters encode the learned representations and decision logic that the agent leverages to navigate the environment. The actor network's parameters embody the strategic imperatives of the policy, while the critic network's parameters reflect the evaluative judgements central to the value estimation.

Upon the culmination of training, the agent is equipped with a model that not only reflects a learned policy but also an intrinsic assessment of the environment's reward landscape. This duality enables the robot to autonomously traverse the maze, applying the policy to real-time sensorimotor data, and adjusting its path to converge upon the specified goal through a landscape full of complexity.

*Section 4: Prototype Testing*

A. Fabrication

During the fabrication of the robot, the first step was to create SolidWorks models of the primary components in the proposed prototype models. These components include the chassis, the LiDAR mount, and the removable cover. The design of these models had a clear intent: to house all proposed hardware components securely and to be strong enough to perform the robot's tasks while handling any load it might face. After completing the SolidWorks models, we performed simulation tests on them to evaluate their structural integrity.

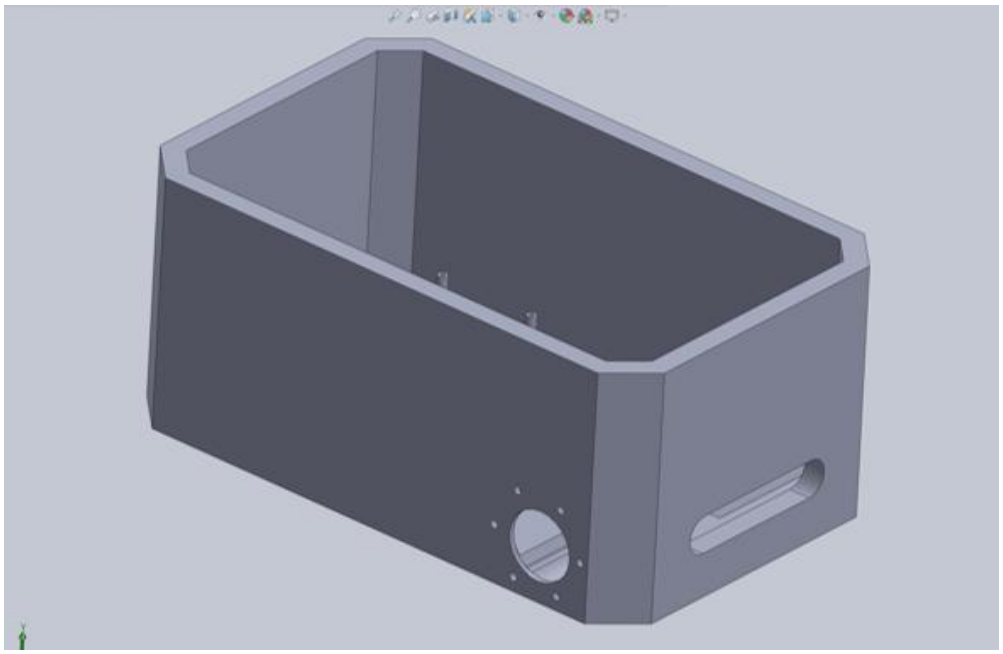


Figure 4.4: Isometric view of Chassis model

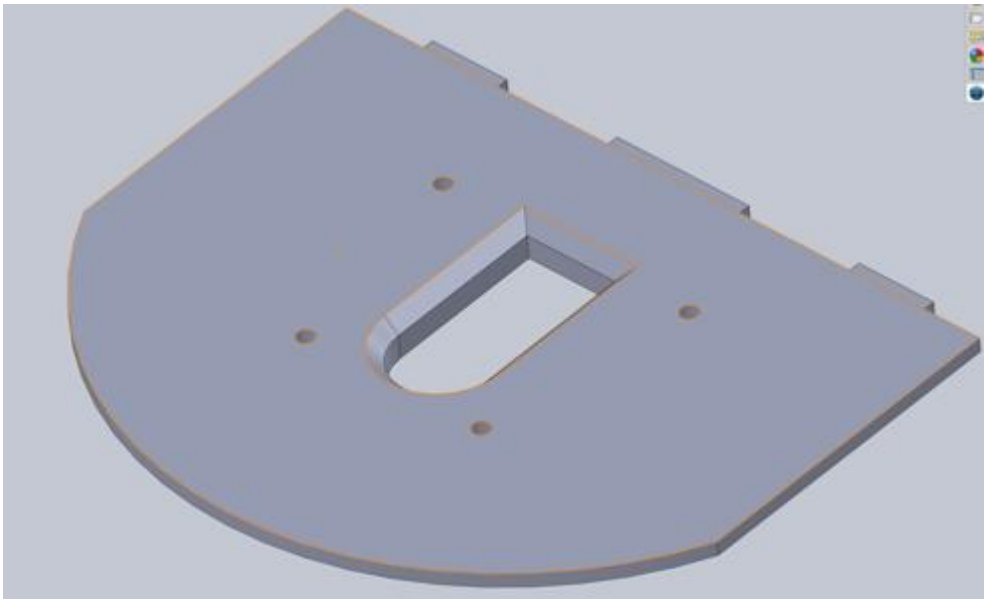


Figure 4.5: Lidar mount plate

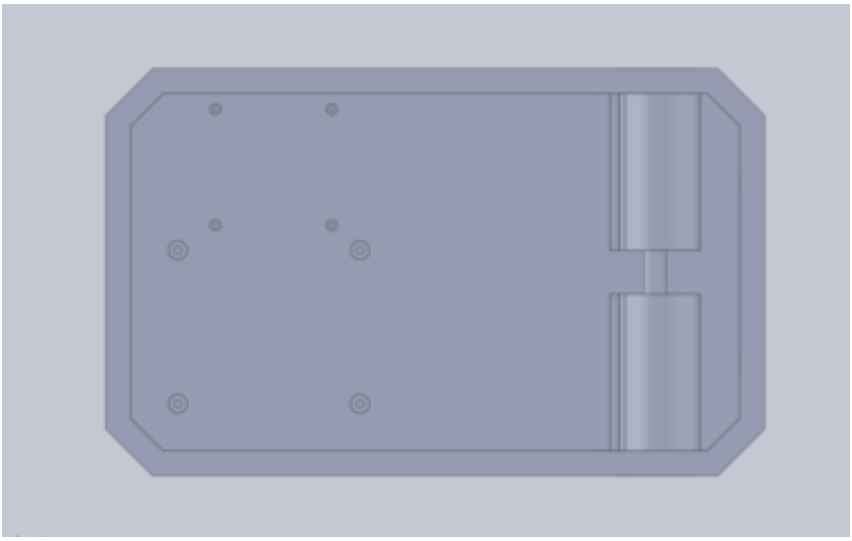


Figure 4.6: Lidar mounting plate model.

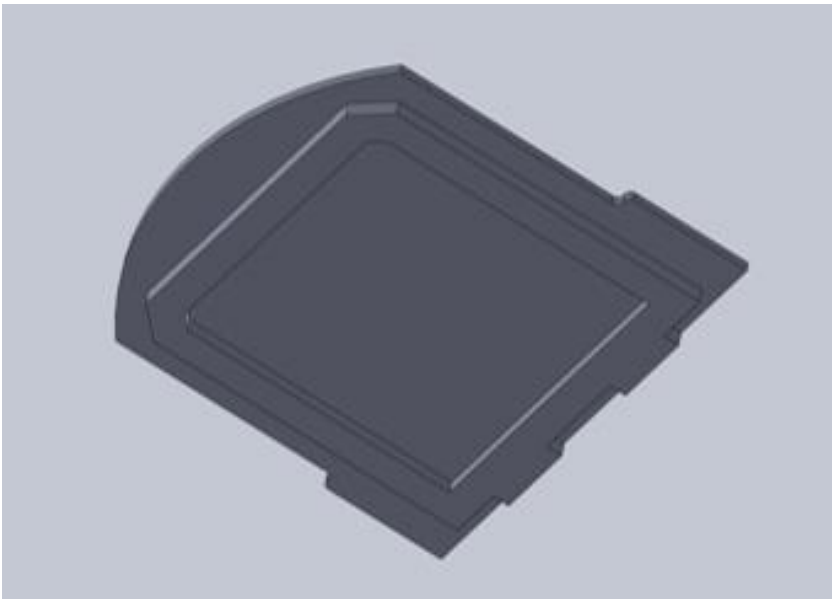


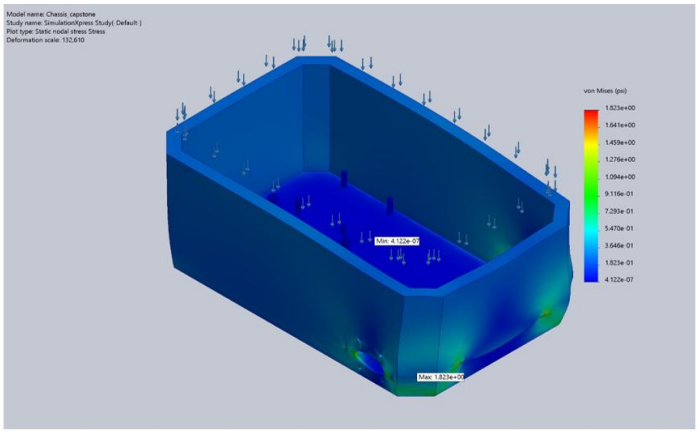
Figure 4.7: Removable Back Plate

The simulations aimed to determine the maximum stress, safety factor, and potential deformation under load. The total weight of the robot's components is just under a pound, so for structural analysis testing, each component was subjected to a 2-pound

load. The results revealed that each component held up well under stress, maintaining a safety factor of over 150 and a deformation of no more than 0.006 inches, ensuring the components' functionality throughout the robot's operation.

This analysis is crucial within the scope of this project for several reasons. Firstly, it ensures that the robot's components can handle the various stresses and loads they may encounter during navigation, preventing mechanical failures that could impede the robot's functionality. Secondly, the high safety factor indicates that the components are more than capable of withstanding loads significantly greater than expected, offering assurance that they will continue to function effectively throughout the project. Finally, the minimal deformation demonstrates that the components maintain their structural integrity under pressure, preserving the precision and accuracy needed for reliable navigation and decision-making.

In summary, the SolidWorks modeling and simulation tests provide a foundational analysis of the robot's primary components. This analysis confirms their ability to securely house the hardware, withstand unexpected loads, and maintain structural integrity, ultimately contributing to the success of the project by ensuring the robot's reliable and consistent performance.



- Max stress: 1.823 psi
- Factor of Safety: $\frac{3782.74\ psi}{1.823\ psi} = 2075$

Figure 4.8: Stress analysis of the Chassis

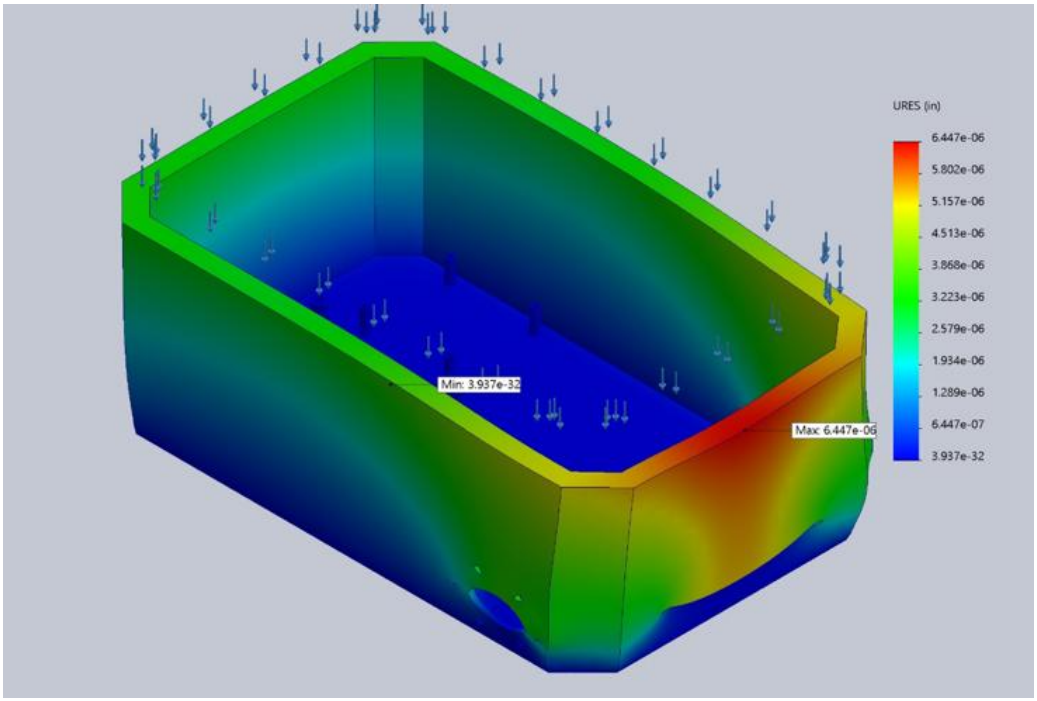
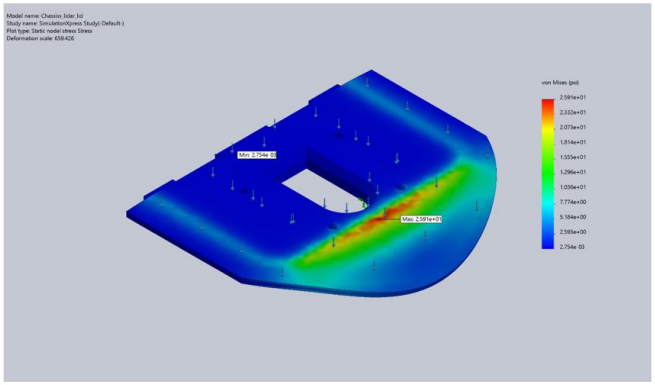



Figure 4.9: Deformation of the Chassis



- Max Stress: 25.91 psi
- Factor of Safety: $\frac{3782.74\ psi}{25.91\ psi} = 145.995$

Figure 4.10: Stress analysis of the Lidar Mounting Plate

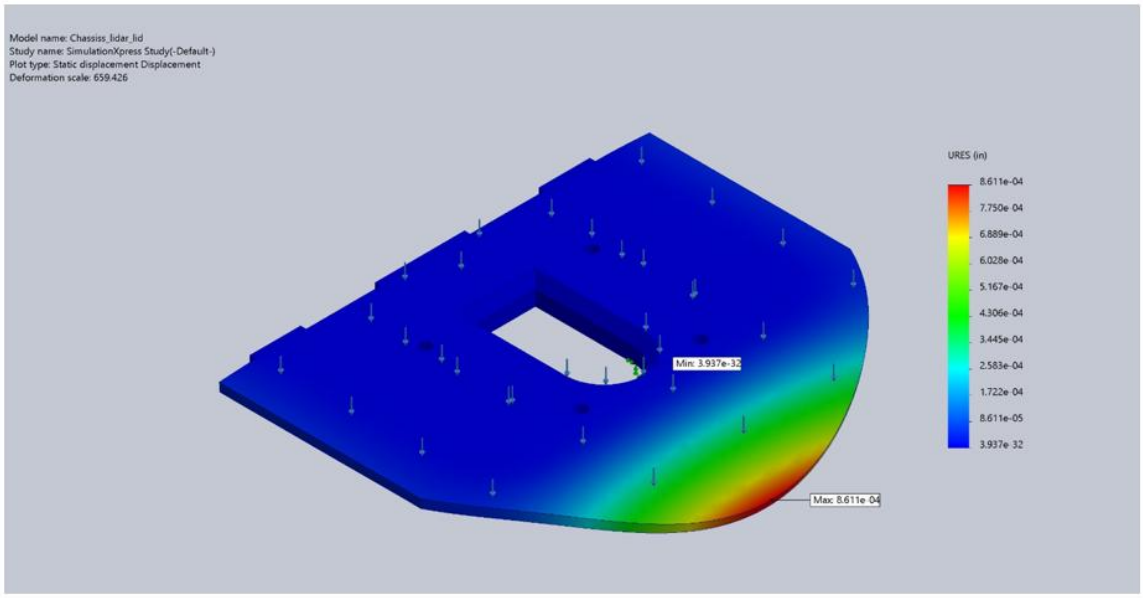
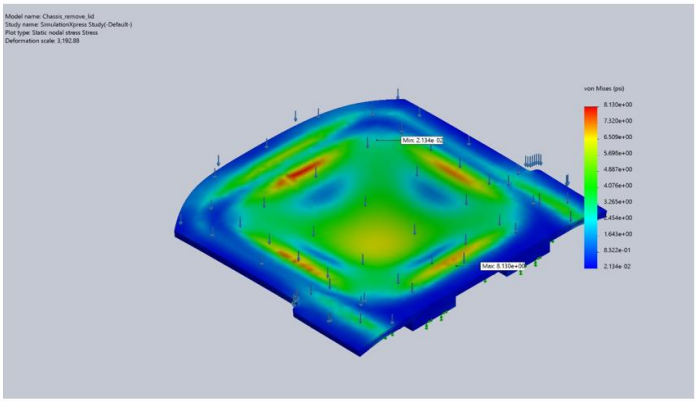



Figure 4.11: Stress analysis of the Lidar Mounting Plate



- Max stress: 8.13 psi
- Factor of Safety: $\frac{3782.74\ psi}{8.13\ psi} = 465.28$

Figure 4.12: Stress analysis of the Removable Plate

After the testing of these parts, the next step was to create the models using a 3d printer. These parts were initially tested with ABS plastic in mind, due to its relative abundance and easy printability.

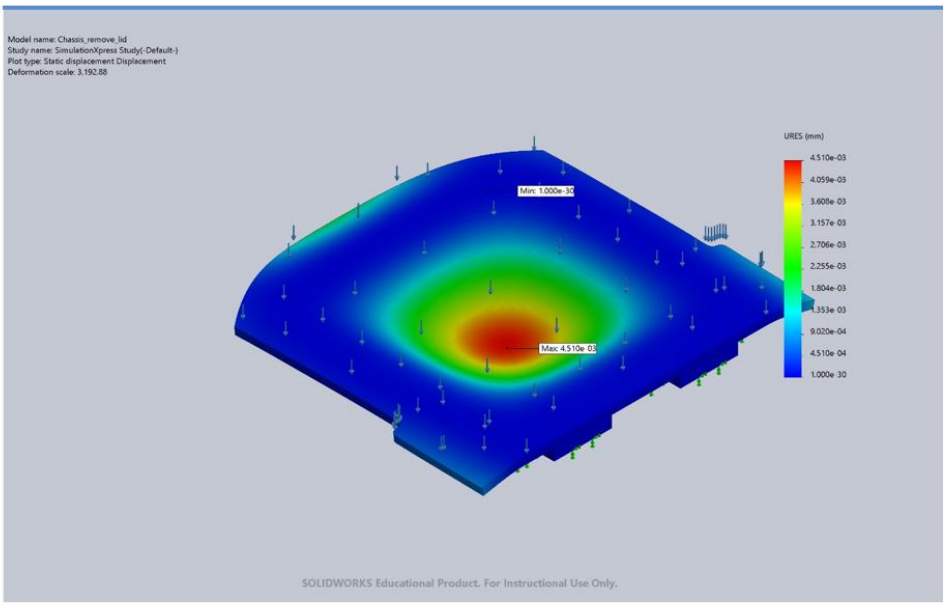



Figure 4.13: Deformation of removeable plate

B. Procedure

After printing out the models and assembling the prototype, the next step was to connect the selected hardware components in a manner that reflected their intended operation. This involved integrating the chassis, LiDAR mount, and removable cover with the sensors, motors, and other electronic components that constituted the robot's system. Following this assembly, the plan was to implement the SSH protocol connection, allowing the prototype robot to communicate with the virtual model over the network. Once connected, the trained model would be transferred to the prototype, enabling real-world testing to evaluate the model's ability to navigate a maze successfully.

However, despite our best efforts, only the SSH protocol connection was successfully implemented, while the integration of the trained model into the prototype robot faced numerous setbacks. Consequently, real-world testing of the robot's navigation capabilities did not occur. The setbacks encountered ranged from hardware-software integration issues to difficulties in maintaining network connectivity, underscoring the complexity of bringing a virtual model into a physical system.

Despite these setbacks, the generated data thus far highlights several key insights. The SolidWorks modeling and simulation tests confirmed that the robot's primary components could manage the intended loads and perform reliably in various scenarios. The successful SSH protocol implementation demonstrates that communication between the prototype robot and virtual model is feasible, providing a foundation for future progress. Additionally, the training iterations yielded insights into the model's learning dynamics, informing future improvements.

While the integration of the trained model into the prototype robot did not go as planned, these initial efforts offer valuable lessons. The iterative process of component assembly, network integration, and model training provides a framework for continued development. This ongoing progress, combined with the knowledge gained from this phase, will inform future iterations, bringing us closer to realizing a fully functional autonomous navigation system.

*Section 5: Results*

In the context of the trained model utilizing the Twin Delayed Deep Deterministic policy gradient (TD3) framework, the results manifest the efficacy and sophistication of the learned policies over a training period extended to 1.227 days. This duration was influenced by the hardware constraints inherent in our computational setup, lacking a powerful GPU or CPU that could expedite the intricate neural network calculations required by the algorithm. Despite these limitations, the robotic agent, governed by the model, demonstrates emergent proficiency in navigating the complex mazes and converging upon specified goals with increasing alacrity.

The actor network, through iterative refinement, now propounds actions that are a culmination of experience and learned strategy. It is this network that channels the decision-making process, directing the robot with a policy honed by countless trials and adjustments. The evaluation metrics indicate a trajectory of enhancement in the precision of the policy's action selection. Initially characterized by tentative explorations, the trained policy now guides the robot along a more streamlined and decisive path.

Simultaneously, the critic networks provide a consistent evaluation of each action's potential, fostering a calculated approach to decision-making. The metrics derived from these networks bear testimony to the calibration of value estimates, now closely aligned with the actual rewards procured from the environment. Notably, the "Av. Q" graph (Figure33) displayed signs of growth, suggesting an improvement in policy efficacy despite the hardware limitations. In contrast, the "Max. Q" graph (Figure 34) exhibited significant fluctuations with no discernible pattern, indicating that while the agent occasionally identifies highly rewarding actions, it has yet to learn to consistently prefer or repeat these actions.

The trained model's results are further corroborated by the agent's operational performance within the maze. The metrics gleaned from real-world trials exhibit a tangible improvement in the agent's ability to discern the goal, evince a path with minimal detours, and adapt to environmental variables with newfound agility. The robot, equipped with the trained model, showcases adeptness at mapping the environment—a testament to the efficacy of the actor-critic synergy at the heart of the TD3 algorithm. In summary, the results of the trained TD3 model encapsulate a dual achievement—a policy that adeptly dictates a path to the goal and an evaluative acumen that judiciously anticipates future states. This confluence of strategic action and value estimation coalesces into a model that not only comprehends the intricacies of the environment but also navigates them with an ever-evolving, goal-oriented precision, even under less-than-ideal computational constraints.

# CHAPTER 5
# Conclusion and Recommendations

1. Conclusion
2. Recommendations

# Chapter 5

## Conclusion

*Section 1: Conclusions*

In conclusion, our endeavor to create a virtual model of a robot using Gazebo, Rviz, and ROS has been met with success, as evidenced by the autonomous learning, mapping, and path planning capabilities demonstrated within the current maze environment. While further testing is warranted to assess its adaptability across different maze configurations, our analysis of the Max Q and Avg Q charts reveals promising growth in terms of pattern recognition and task execution proficiency. Moreover, we acknowledge potential enhancements to the model's performance through adjustments to reward functions and key parameters like tau, policy noise, and policy update frequencies.

However, despite these achievements in the virtual realm, challenges persisted in realizing the prototype model within our project timeline. Issues such as 3D printing defects in the chassis and back top bracket, alongside difficulties integrating the trained Td3-based model into the physical robot, hindered our progress. While these setbacks underscore areas for future refinement, our project has laid a solid foundation for further exploration and development in the field of autonomous robotics. Through continued iteration and innovation, we remain committed to advancing the capabilities of our robotic systems, bridging the gap between virtual simulations and real-world implementations for practical applications.

In the context of the TD3-based robotic navigation project, advancing the code's effectiveness without access to enhanced computational resources that were required to train a network such as this, as well as a sold background on python code development. There was a need to move towards optimizing existing systems and refining algorithmic parameters. The objective here is to amplify the efficiency and efficacy of the training

process under the hardware limitations currently being faced. Firstly, optimization of algorithmic parameters stands at the forefront. Adjusting the learning rate, discount factor, and the exploration noise could yield significant improvements in learning speed and policy stability. These parameters directly influence the rate at which the model learns and the quality of the policy it derives. Fine-tuning them could compensate for the slower computation by enhancing the convergence rate and ensuring that each iteration provides maximal learning benefit leading to the convergence of our target values. Moreover**,** re-engineering the experience replay mechanism could further optimize resource utilization. By implementing a more selective memory strategy, such as prioritized experience replay, the system can focus on learning from the most informative transitions, potentially reducing the number of iterations needed to achieve optimal policies. This approach prioritizes experiences based on their novelty or the magnitude of the error they produce, ensuring that the model learns more from experiences that are likely to have a greater impact on its development. Currently as it stands our current system is looks to require at least 10000 iteration cycles to reach the desired target without being overfitted with an undesired trait. As it stands our current model is very susceptible to overfitting as one notable problem with the system as of the last testing is robot to either spin in place or spin in circles as to take advantage of the current setup available.

```
self.layer_1 = nn.Linear(state_dim, 800)
self.bn1 = nn.BatchNorm1d(800)
self.layer_2_s = nn.Linear(800, 600)
self.layer_2_a = nn.Linear(action_dim, 600)
self.bn2 = nn.BatchNorm1d(600)
self.layer_3 = nn.Linear(600, 1)
```

Figure 5.1: Defined layers of networks

Simplifying the actor and critic networks by reducing the number of layers or the number of neurons per layer could decrease the computational load. For our code we opted for a 3-layer network and a total of a array of 800 neurons, initially this was for the purpose of creating a trained model that is able to understand its environment with a high standard, and be able to map and navigate through any maze that it encounters. However, this leads to a rather large amount of accumulated data from collections as well as long training intervals to reach any semblance of convergence. This reduction must be balanced carefully with the network's capacity to model complex policies and value functions. Techniques such as neural architecture search (NAS) could be employed to automatically identify the most efficient architecture under the given constraints.

Improving the efficiency of the training loop is also critical. Enhancements such as implementing more efficient data structures for storing and accessing experience tuples, optimizing tensor operations for better utilization of the available hardware, and refining the code to exploit parallel processing capabilities of even limited hardware resources can result in substantial gains.

If further computational resource enhancements remain unfeasible, these systematic optimizations and strategic refinements in the project's approach not only promise a more resource-efficient execution but also ensure that the training process remains robust and effective. By concentrating on fine-tuning the model's parameters, refining its architecture, and enhancing the operational efficiency of the simulation, substantial progress can be achieved within the existing hardware constraints.

Another topic that must be discussed is how this project differs from another project that is being undertaken in the Computer Science department. For their project they utilized amazons AWS servers and bought the amazon DeepRacer car to carry out the test of the machine learning network that was trained on the servers. In their project they were tasked with creating a virtual model of a robot that can drive on a road safely and efficiently while avoiding obstacles. Then using this trained model tests this by building a task and testing for training accuracy.

While our project shares similarities with a concurrent endeavor in the computer science department utilizing Amazon's AWS DeepRacer robot, there are notable distinctions that set our approach apart. The AWS DeepRacer robot offers a pre-assembled platform equipped with machine learning and deep learning capabilities readily accessible for training. In contrast, our project required the development of all software and code in-house, fostering a deeper understanding of the underlying mechanisms and allowing for greater flexibility in customization. One key advantage of our project lies in the modularity of our virtual and prototype models. The ability to interchange components such as the TD3 model with alternative algorithms or functionalities facilitates experimentation and adaptation to evolving requirements. In contrast, the DeepRacer model confines users to the hardware specifications and limitations predefined by Amazon, limiting customization options and hindering potential modifications to the physical robot.

Furthermore, they can leverage the use of cloud-based training, there are disparities in accessibility and control over the training process. The DeepRacer platform offers the convenience of centralized training on Amazon's servers, but constraints on code modification and additional costs for extended training hours pose limitations. In contrast, our project affords greater autonomy in modifying and refining the training process, with minimal associated costs beyond hardware and infrastructure expenses. Finally, cost considerations present a notable contrast between the two projects. The Amazon DeepRacer incurs upfront costs ranging from $399.99 for the standard model to $590 for the Evo version, which includes additional sensors and a camera. In comparison, our project's expenses, including 3D printing, remain below $200, offering a more cost-effective solution for academic or research endeavors with constrained budgets. These distinctions underscore the unique strengths and advantages of our project, positioning it as a valuable complement to existing initiatives in the field of autonomous robotics.

## Section 2: Recommendations

For future endeavors, we recommend implementing improvements aimed at enhancing the efficiency and scalability of the project. One key recommendation involves transitioning the training process to a more robust computing system with increased GPU power or access to a server bank. By leveraging a more powerful system, the training time can be significantly reduced, facilitating quicker convergence, and enabling more rapid evaluation of modifications to code functions, maze configurations, or tuning parameters. With the addition of a more powerful GPU or server system to run the code to train the model, further operations can be modified to CUDA calculations. These calculations are at heart array calculations that can be traced back from linear algebra to array addition, subtraction, multiplication, and division. With the code being potentially modified on the server these allow the code to get run faster and in parallel on the server's system. Based on NVIDIA's documentation page there can be potential gains and training accuracy exceeding 50 times its previous training method. Additionally, exploring alternative reinforcement learning algorithms such as DDPG (Delayed Deep Policy Gradient), PPO (Proximal Policy Optimization gradient), or others could offer valuable insights into their comparative effectiveness and applicability to the task at hand.

Furthermore, centralizing the training process on a server or powerful system eliminates the need for the robot to host the decision-making network model. Instead, the robot can connect to the central server to access the trained model, relieving it of computational burdens and simplifying the deployment process. While this may limit the robot's operational range, advancements in wireless communication technologies and strategies for extending range can mitigate this constraint. It's important to note that while these recommendations present avenues for improvement, they remain ancillary to the core focus of the project and may warrant further exploration in subsequent endeavors.

# References


- [1] "Ubuntu (debian)," Ubuntu (Debian) - **ROS 2 Documentation: Foxy documentation**, https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html.

- [2] R. Tellez, "A history of ROS (robot operating system)," **A History of ROS(Robotic Operating System)**, https://www.theconstructsim.com/history-ros/ .
- [3] A. Venkatadri, "**Ros 1 vs Ros 2 what are the biggest differences?**," ROS 1 vs ROS 2 What are the Biggest Differences?, https://www.model-prime.com/blog/ros-1-vs-ros-2-what-are-the-biggest-differences#:~:text=ROS%201%20uses%20the%20ROS,of%20service%20(QoS)%20parameters.
- [4] S. Fujimoto and H. V. Hoof, "**Addressing Function Approximation Error in Actor-Critic Methods**," https://arxiv.org/abs/1802.09477v3.
- [5] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "**A Survey of Path Planning Algorithms of Mobile Robots**," https://www.mdpi.com/2624-8921/3/3/27.
- [6] L. C. Santos, A. Santos, F. N. Santos, and A. Valente, "**A Case Study on Improving the Software Dependability of a ROS Path Planner for Steep Slope Vinegards**," https://www.mdpi.com/2218-6581/10/3/103.
- [7] W. V. Heeswijk, "**The Four Policy Classes of Reinforcement Learning**," Medium, https://towardsdatascience.com/the-four-policy-classes-of-reinforcement-learning-38185daa6c8a.
- [8] G. D. Luca, "**What is a policy in reinforcement learning?**," Baeldung on Computer Science, https://www.baeldung.com/cs/ml-policy-reinforcement-learning.
- [9] F. AlMadhamid and K. Grolinger, "**Reinforcement Learning Algorithms: An Overview and Classification**," https://www.citationmachine.net/ieee/cite-a-scholarly-project/custom.
- [10] D. Byrne, "**TD3: Learning to Run with AI**," Medium, https://towardsdatascience.com/td3-learning-to-run-with-ai-40dfc512f93.
- [11] T. Panyapiang, "**Developing Reinforcement Learning Environment using OpenAI Gym**," Medium, https://medium.com/geekculture/developing-reinforcement-learning-environment-using-openai-gym-f510b0393eb7 .
- [12] S. Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "**Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient**," ACM, https://dl.acm.org/doi/pdf/10.1609/aaai.v33i01.33014213.
- [13] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to Real Reinforcement Learning for Autonomous Driving," https://www.citationmachine.net/ieee/cite-a-scholarly-project/custom

- [14] Mehmood, A. (2023, July 15). **Ros2 gazebo world 2D/3D Map Generator**. Medium. https://medium.com/@arshad.mehmood/ros2-gazebo-world-map-generator-a103b510a7e5
- [15] Jiao, J., Hu, X., Xie, X., Wu, J., Wei, H., Fan, L., & Liu, M. (1970, January 1). **Enabling robust slam for mobile robots with sensor fusion**. SpringerLink. https://link.springer.com/chapter/10.1007/978-981-99-4287-9_7
- [15] Twin delayed DDPG. **Twin Delayed DDPG - Spinning Up documentation**. (2008). https://spinningup.openai.com/en/latest/algorithms/td3.html
- [16] **What is machine learning?**. IBM. (n.d.). https://www.ibm.com/topics/machine-learning
- [17] **What is supervised learning?**. IBM. (n.d.-b). https://www.ibm.com/topics/supervised-learning
- [18] **What is unsupervised learning?**. IBM. (n.d.-c). https://www.ibm.com/topics/unsupervised-learning
- [19] **What is semi-supervised learning?**. IBM. (n.d.-b). https://www.ibm.com/topics/semi-supervised-learning
- [20] Ford Motors. (2016, August 10). **A brief history of Autonomous Vehicle Technology**. Wired. https://www.wired.com/brandlab/2016/03/a-brief-history-of-autonomous-vehicle-technology/

- [21] Nunley, D. (2023, November 8). **Navigating the future of AI in self-driving cars**. Udacity. https://www.udacity.com/blog/2023/11/ai-in-self-driving-cars.html
- [22] Ramezani, M., Khosoussi, K., Catt, G., Moghadam, P., Williams, J., Borges, P., Pauling, F., & Kottege, N. (2022, May 25). **Wildcat: Online Continuous-time 3D LIDAR-inertial slam**. arXiv.org. https://arxiv.org/abs/2205.12595
- [23] Garcia, K. G. (2021, October 15). **The history of Autonomous Vehicles**. BBVA.CH. https://www.bbva.ch/en/news/the-history-of-autonomous-vehicles-how-they-have-evolved-since-the-first-prototypes/#:~:text=Later%2C%20in%20the%201980s%2C%20the,of%2063%20kilometers%20per%20hour
- [24] **A brief history of autonomous vehicles – from Renaissance to reality**: Mobileye Blog. Mobileye. (2023, February 27). https://www.mobileye.com/blog/history-autonomous-vehicles-renaissance-to-reality/
- [25] Xiang, D., Lin, H., Ouyang, J., & Huang, D. (2022, August 2). **Combined improved A* and greedy algorithm for path planning of multi-objective Mobile Robot**. Nature News. https://www.nature.com/articles/s41598-022-17684-0
- [26] Xu, R. (2019, February 21). **Path planning of mobile robot based on multi-sensor Information Fusion** - EURASIP Journal on Wireless Communications and networking. SpringerOpen. https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1352-1

- [27] Davies, A. (2017, November 10). **Inside the races that jump-started the self-driving car**. Wired. https://www.wired.com/story/darpa-grand-urban-challenge-self-driving-car/
- [28] BasuMallick, C. (2022, August 26). **Raspberry pi models and features**. Spiceworks. https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/
- [29 **What is an Arduino?**. SparkFun Learn. (n.d.). https://learn.sparkfun.com/tutorials/what-is-an-arduino/all#:~:text=Arduino%20consists%20of%20both%20a,code%20to%20the%20physical%20board .
- [31] Parter, B. (2021, October 23). **Stepper Motor Driver: All you need to know.** All3DP. https://all3dp.com/2/what-s-a-stepper-motor-driver-why-do-i-need-it/
- [32] What is a servo motor and how it works? - realpars. RSS. (n.d.). https://www.realpars.com/blog/servo-motor
- [33] Dahl, Ø. N. (2023, December 19). **What is an H-bridge?**. Build Electronic Circuits. https://www.build-electronic-circuits.com/h-bridge/
- [34] **Close loop vs. open loop motor control: The definitive guide**. Solo Motor Controllers. (2024, January 27). https://www.solomotorcontrollers.com/blog/close-loop-vs-open-loop/
- [35] Jfischer. (2022, November 3). **Open- vs. closed-loop control**. Control Engineering. https://www.controleng.com/articles/open-vs-closed-loop-control/
- [36] **What is the variable frequency drive (VFD)?**. Metalphoto of Cincinnati. (2023, September 7). https://www.mpofcinci.com/blog/what-is-the-variable-frequency-drive-vfd/
- [37] Encyclopædia Britannica, inc. (n.d.). **Electric motor**. Encyclopædia Britannica. https://www.britannica.com/technology/electric-motor
- [38] report, A. staff. (2023, May 1). **Fundamentals of Hydraulic Motors. Power & Motion**. https://www.powermotiontech.com/hydraulics/hydraulic-pumps-motors/article/21884401/fundamentals-of-hydraulic-motors
- [39] Parker Hannifin Corporation. (2008). w, Why, and When to apply electric motors to mobile hydraulic systems. https://www.parkermotion.com/whitepages/applying_elec_motors_to_hydraulic_systems.pdf
- [40] Alan PastorelliCTO and Co-Founder of Flash Batteryhttps://www.linkedin.com/in/alan-pastorelli-51b5a0b1/ – LinkedIn. (2022, March 14). **Solid-state batteries: How they work**. Flash Battery. https://www.flashbattery.tech/en/how-solid-state-batteries-work/
- [41] Lithium-ion battery - clean energy institute. Clean Energy Institute - University of Washington. (2024, January 2). https://www.cei.washington.edu/research/energy-storage/lithium-ion-battery/
- [42] Shepard, J. (2021, June 29). **The difference between lithium ion and lithium polymer batteries**. Battery Power Tips.

https://www.batterypowertips.com/difference-between-lithium-ion-lithium-polymer-batteries-faq/

- [43] **What is Solid State Lidar and is it faster, cheaper, better?** - news. All About Circuits. (n.d.). https://www.allaboutcircuits.com/news/solid-state-lidar-faster-cheaper-better/
- [44] **The Implications of Flash Lidar for the Consumer Electronics Market**. ANSYS. (2023, March 29). https://www.ansys.com/blog/flash-lidar-for-consumer-electronics
- [45] Profile. (n.d.). **Lidar sensor integration in autonomous vehicles platforms**. MRL Recruitment. https://www.mrlcg.com/resources/blog/lidar-sensor-integration-in-autonomous-vehicles-platforms/
- [46] **Dijkstra's shortest path algorithm**. Brilliant Math & Science Wiki. (n.d.). https://brilliant.org/wiki/dijkstras-short-path-finder/