Qatar University

College of Engineering

Department of Computer Science and Engineering

Senior Project Report

Autonomous Golf Cart

Project Group Members:
Abdelaziz Shehata (202107370)
Zabin AlDosari (202009034)
Abdulla Musa (202006166)

Supervisor: Uvais Qidwai

2024

This project report is submitted to the Department of Computer Science and Engineering of Qatar University in partial fulfillment of the requirements of the Senior Project course.

Declaration

This report has not been submitted for any other degree at this or any other University. It is solely the work of us except where cited in the text or the Acknowledgements page. It describes work carried out by us for the capstone design project. We are aware of the university's policy on plagiarism and the associated penalties and we declare that this report is the product of our own work.

Student: Abdelaziz Shehata Date: 28/11/2024

Signature: [illegible]

Student: Zabin Al-Dosari Date: 28/11/2024

Signature: [illegible]

Student: Abdulla Musa Date: 28/11/2024

Signature: [illegible]

Abstract

The development of autonomous vehicles has revolutionized the transportation industry, extending beyond personal mobility and logistics to more specialized environments. This project aims to design and implement a Level 4 autonomous golf cart, tailored specifically for controlled environments such as university campuses and golf courses. The primary objective is to create a fully autonomous vehicle capable of navigating without human intervention, providing a safe, efficient, and low-speed transportation solution. By utilizing advanced sensors and powerful computing platforms, this project explores the potential of integrating cost-effective, off-the-shelf components to build a robust, real-world autonomous system.

In recent years, Qatar has emerged as a leader in adopting advanced technologies across several sectors, aligning with its vision to become a global hub for innovation. Autonomous vehicles, such as the proposed golf cart, align perfectly with this trajectory, enhancing the nation's technological evolution. Golf carts have already demonstrated their utility during the FIFA 2022 World Cup, where they played a crucial role in efficiently transporting personnel and materials across large venues [1]. By integrating autonomous golf carts into future mega-events, Qatar could further elevate its reputation for technological excellence. Autonomous golf carts would not only enhance operational efficiency but also provide an eco-friendly solution, aligning with Qatar's sustainability goals and commitment to reducing carbon emissions.

The system architecture of the proposed solution is structured into different layers: at the top, the Jetson Orin NX 16GB RAM AI 100TOPS Development Kit handles high-level tasks such as sensor fusion, object detection, path planning, and obstacle avoidance. At the bottom layer, microcontrollers manage key sensors—LiDAR for environmental mapping, stereo cameras for object identification, and ultrasonic sensors for close-range obstacle detection. The system also includes a Speed Controller connected to the torque motor for controlling the cart's acceleration and movement. This layered architecture ensures seamless communication among components, allowing the cart to perceive its environment, make decisions in real-time, and autonomously navigate through complex terrains.

Designed for thorough testing in predictable environments such as university campuses, the system's integration of Python and ROS2 for LiDAR data preprocessing enhances its ability to create accurate 2D maps, while ROS2 ensures efficient communication and sensor fusion for real-time navigation. The novelty of this design lies in its use of cost-effective hardware, delivering advanced functionality typically reserved for more expensive systems. The successful implementation of this autonomous golf cart will contribute to Qatar's ongoing technological advancements, positioning it as a pioneer in integrating autonomous systems into practical, high-impact applications.

Acknowledgment

We extend our heartfelt gratitude to Dr.Uvais Qidwai for his invaluable guidance and unwavering support throughout this project. His expertise and insightful advice have not only shaped our work but also deepened our understanding of the subject matter.

We are profoundly grateful to our dedicated team, whose relentless efforts, collaborative spirit, and innovative ideas were instrumental in overcoming challenges and achieving our objectives.

Additionally, we sincerely thank our peers, friends, and families for their constant encouragement and unwavering belief in our abilities, which have been a source of motivation and strength.

This project is a testament to the collective effort, dedication, and shared vision of everyone involved. Thank you for being an integral part of this journey.

Table of Contents

List of Figures

List of tables

1. Introduction and Motivation

Initially, our project aimed to develop an autonomous golf cart as a proof of concept for low-speed autonomous transportation in controlled environments. However, due to budget constraints and time limitations, securing a full-sized golf cart was not feasible. Acquiring a golf cart required significant financial resources, which exceeded our available budget, and the time required to seek external funding or sponsorship did not align with our project timeline. To ensure the successful completion of our project within the given timeframe, we adapted our approach by implementing our autonomous system on a mini motorized vehicle instead. This alternative allows us to develop and test the same core functionalities, including autonomous navigation, obstacle detection, and real-time decision-making, in a more cost-effective and manageable manner. While the vehicle scale has changed, the underlying engineering challenges and objectives remain the same, ensuring that the project still demonstrates the feasibility of autonomous mobility in controlled environments.

University campuses, golf courses, and other controlled environments currently rely on human-driven vehicles for short-distance, low-speed transportation. These vehicles are used to shuttle passengers between locations, but their operation comes with a set of challenges, such as high labor costs, human error, and potential safety concerns. Human drivers are prone to fatigue, mistakes, and inconsistencies in their driving behavior, which can lead to inefficiencies and accidents. In environments where transportation needs are constant and predictable, an autonomous solution would provide greater efficiency, reduce labor costs, and minimize the risks of human error. The challenge is to create a system that can safely navigate through these spaces, detect obstacles in real time, and make decisions autonomously without human intervention.

Addressing this problem is vital for improving transportation systems in controlled environments. Autonomous vehicles offer several advantages over human-driven vehicles, including increased safety, reduced labor costs, and improved operational efficiency. By removing the need for human drivers, institutions like universities and resorts can significantly lower their operational costs. Moreover, autonomous vehicles are not subject to fatigue or human error, making them safer for environments with a high pedestrian population, such as campuses or parks. These environments require vehicles to operate at low speeds, avoid pedestrians and obstacles, and adapt to changing conditions. Solving this problem also contributes to the broader field of autonomous vehicles by demonstrating the practicality of deploying autonomous systems in low-speed, well-defined environments, which is an area of research that has been relatively underexplored compared to urban or highway driving.

Despite the predictable nature of controlled environments, developing an autonomous vehicle for these spaces presents several technical and cost-related challenges. First, the vehicle must detect and avoid both static and dynamic obstacles, such as pedestrians, trees, and benches. This requires the integration of multiple sensors, including LiDAR for mapping the environment, stereo cameras for object identification, and ultrasonic sensors for detecting close-range obstacles. The system must process this sensor data in real-time, ensuring that decisions are made quickly and accurately to avoid collisions. Additionally, achieving this level of functionality using cost-effective components is a major challenge. Autonomous systems typically rely on expensive sensors and industrial-grade processing units, but this project aims to demonstrate that similar results can be achieved using commercially available hardware. Finally, sensor fusion combining data from multiple sensors to create an accurate representation of the environment is another technical obstacle. It requires real-time synchronization of sensor data and a robust software platform to handle decision-making. ROS2 (Robot Operating System 2) plays a critical role in managing sensor communication and ensuring that the system can make informed decisions based on sensor inputs.

The problem of autonomous vehicle navigation in controlled environments is particularly interesting because it addresses a significant gap in autonomous vehicle research. Most existing research focuses on high-speed, urban driving scenarios, which present complex challenges like traffic management and unpredictable pedestrian behavior. However, controlled environments like university campuses and golf courses are more predictable and offer an ideal proving ground for autonomous vehicles. Solving this problem has the potential to impact several industries that rely on short-distance transportation, such as education, healthcare, and hospitality. Moreover, the technical challenges of developing a cost-effective, autonomous system using affordable sensors and computing platforms like the Jetson Orin NX make this project a valuable contribution to the field. The ability to deploy autonomous systems in controlled environments could pave the way for similar applications in other areas, such as industrial parks, resorts, and smart cities, where safe and reliable transportation is critical.

This project proposes the development of a Level 4 autonomous golf cart capable of navigating safely without human intervention in controlled environments. The system uses a three-layer architecture, with the Jetson Orin NX 16GB RAM AI 100TOPS Development Kit serving as the central processing unit, responsible for sensor fusion, path planning, and decision-making. The Jetson Orin NX is connected to three Arduino microcontrollers, each dedicated to managing specific sensors: LiDAR for environmental mapping, stereo cameras for object identification, and ultrasonic sensors for close-range obstacle detection. A Speed Controller is integrated with the torque motor, allowing the vehicle to control its speed and acceleration based on sensor data. The system operates on ROS2, which enables real-time communication between sensors and processing units, ensuring that the vehicle can respond quickly to changes in its environment. Python and ROS2 is used for preprocessing LiDAR data, enhancing the system's ability to generate accurate 2D maps of its surroundings. The vehicle will be tested in predictable environments, such as university campuses, where predefined routes and regular traffic conditions make it easier to refine the system's autonomous capabilities. By using cost-effective hardware and focusing on scalability, this project demonstrates that complex autonomous vehicle functionality can be achieved at a fraction of the cost typically associated with such systems, making it accessible for a wide range of applications.

1.1. Problem statement

Transportation within controlled environments such as university campuses, resorts, and golf courses—faces significant challenges related to efficiency, safety, and sustainability. Traditional reliance on human-operated vehicles introduces risks associated with human error, which is a leading cause of traffic incidents. For instance, a comprehensive study found that human errors and deficiencies were definite or probable causes in 90-93% of traffic accidents [2]. These risks are amplified in areas with high pedestrian traffic, where ensuring safety becomes increasingly complex.

Moreover, manual transportation systems often exhibit inefficiencies, such as inconsistent service and elevated operational costs. Environmental concerns also arise from the use of fossil fuel-powered vehicles, contributing to pollution and greenhouse gas emissions. University campuses, for example, experience traffic congestion and parking challenges due to high dependency on private vehicles, leading to environmental degradation [3].

Addressing these issues necessitates the development of scalable, autonomous transportation systems capable of operating in controlled environments with minimal human intervention. Such systems should integrate advanced sensing, decision-making, and control technologies to enhance safety, reduce costs, and align with sustainability goals. The proposed autonomous golf cart system aims to tackle these transportation challenges by utilizing state-of-the-art sensors, artificial intelligence, and electric vehicle technologies.

However, implementing this system involves several technical and non-technical challenges:

Technical Challenges:

- Obstacle Detection and Avoidance: The system must detect and navigate around both static and dynamic obstacles, such as pedestrians, bicycles, and fixed structures, in real time. This requires seamless integration of LiDAR, stereo cameras, and ultrasonic sensors to accurately perceive the environment.
- Real-Time Data Processing: Processing large volumes of data from multiple sensors simultaneously is critical. Utilizing platforms like the Jetson Orin NX for high-level processing and Arduino microcontrollers for specific sensors introduces complexities in managing sensor fusion and ensuring prompt decision-making.
- Sensor Fusion: Combining data from several sensors is essential for creating a comprehensive environmental model. Synchronizing this data in real time to ensure accurate decision-making remains a significant challenge.
- Power and Efficiency: The autonomous golf cart must manage power consumption effectively while operating multiple sensors and maintaining continuous processing for navigation and decision-making.
- System Scalability: Ensuring that the system architecture can adapt to different environments and scale for larger campuses or other low-speed applications is crucial. The solution must balance cost and functionality while remaining adaptable.

Non-Technical Challenges:

- User Acceptance and Trust: Gaining the trust of end-users (e.g., students, faculty, or golfers) is vital for successful implementation. The system must adhere to high safety standards to build confidence among users.
- Regulatory Compliance: The system must comply with local transportation laws and regulations, especially in public or semi-public environments like university campuses. Securing legal approval may be required before deployment.
- Cost Constraints: Developing a cost-effective solution using commercially available components is a key objective. The challenge lies in maintaining low costs while ensuring reliable performance and meeting all technical requirements.
- Maintenance and Support: Post-deployment, the system will require regular maintenance and updates to ensure long-term reliability. Establishing a clear plan for maintenance and system updates is essential for continued operation.

By addressing these challenges, the autonomous golf cart system aims to provide a robust and innovative solution to the transportation issues prevalent in controlled environments.

1.2. Project objectives

- Develop a Fully Autonomous Golf Cart:

  - Design and build a Level 4 autonomous golf cart that can navigate through a controlled environment (e.g., university campus or golf course) without human intervention.
  - Ensure the system operates safely and reliably at low speeds, capable of handling dynamic environments with pedestrians and other vehicles.
- Integrate Advanced Sensor Systems:
  - Incorporate LiDAR for environmental mapping, stereo cameras for object identification, and ultrasonic sensors for close-range obstacle detection.
  - Achieve seamless sensor fusion using ROS2 to combine data from these sensors for real-time decision-making.
- Real-Time Data Processing:
  - Use the Jetson Orin NX 16GB RAM AI Development Kit to handle high-level processing, sensor fusion, and path planning.
  - Ensure real-time data processing for obstacle detection and avoidance, without delay, to maintain safe operation.
- Obstacle Detection and Avoidance:
  - The system must reliably detect both static (e.g., trees, benches) and dynamic (e.g., pedestrians, vehicles) obstacles and adjust its route accordingly.
  - Use Python and ROS2 to preprocess LiDAR data, creating accurate 3D maps of the environment to improve obstacle detection and path planning.
- Navigation and Path Planning:
  - Implement algorithms for global and local path planning that allow the golf cart to follow predefined routes while dynamically adjusting to obstacles.
  - Achieve smooth and efficient navigation of predefined paths while adhering to safety requirements.
- Cost-Effective Solution:
  - Develop the entire system using commercially available components to ensure scalability and affordability.
  - Balance cost and performance to create a solution that can be replicated in other controlled environments without significant expense.
- Testing and Validation:
  - Thoroughly test the autonomous golf cart in controlled environments such as university campuses and golf courses, to validate its ability to navigate, detect obstacles, and avoid collisions.
  - Perform tests under several conditions (e.g., different lighting, weather, pedestrian traffic) to ensure reliability.

1.3. Project significance and expected benefits

The increasing demand for autonomous transportation systems in controlled environments arises from the need to enhance safety, reduce operational costs, and improve transportation efficiency. Currently, transportation in such environments relies heavily on human-operated vehicles, which introduces risks of human error—responsible for over 90% of traffic incidents globally [4]. This dependency not only compromises safety but also results in inefficiencies such as inconsistent service and elevated costs, further increased by the environmental impact of fossil fuel-powered vehicles. Without a viable autonomous alternative, institutions risk continued reliance on outdated systems that increase their carbon footprint and operational expenses.

Our project addresses these challenges by offering a scalable, cost-effective autonomous solution tailored specifically for controlled environments like university campuses, resorts, and golf courses.

By utilizing commercially available components, such as the Jetson Orin NX, LiDAR, stereo cameras, and Arduino microcontrollers, we demonstrate that advanced autonomous functionality can be achieved without the high costs associated with industrial-grade systems. For example, the cost of our core components (see Table 1) reflects a carefully balanced design that ensures affordability while maintaining reliability and functionality.

The autonomous golf cart integrates real-time sensor fusion and decision-making through the ROS2 framework. This provides a replicable and adaptable solution, which can be scaled to other settings, such as larger campuses or other low-speed applications. Furthermore, by employing an electric-powered design, the project aligns with global sustainability goals, offering an eco-friendly alternative to traditional fossil fuel-powered vehicles. This supports the reduction of greenhouse gas emissions and promotes greener transportation practices.

Beyond its practical application, the project makes a significant academic contribution. It provides a framework for developing autonomous vehicles for low-speed, pedestrian-heavy environments, addressing challenges such as obstacle detection, navigation in narrow pathways, and real-time decision-making. Existing autonomous systems, like those designed for urban or highway scenarios, are often tailored to complex, high-speed environments and rely on expensive, high-end sensors. Our project offers an innovative solution by adapting autonomous technologies to controlled environments using affordable, modular components. This approach not only highlights a novel application but also paves the way for broader adoption of autonomous systems in sectors like healthcare, hospitality, and education.

Additionally, this project offers invaluable professional development opportunities for our team. It allows us to explore hands-on applications of robotics, sensor integration, and real-time data processing. The knowledge and skills gained from working on this system will better prepare us for careers in the rapidly growing fields of autonomous systems and robotics.

In summary, our project represents a forward-thinking approach to solving transportation challenges in controlled environments. It offers a safer, more reliable, and cost-effective solution while contributing to sustainability and advancing research in autonomous systems. The practical impact and scalability of our design position it as a meaningful innovation in the field of autonomous vehicles.

Table 1. Components' Prices

Component	Price
Intel RealSense D435i Webcam - 2 Megapixel - 30 fps - USB 3.1	$329.68
Jetson Orin NX 16GB RAM AI 100TOPS Development Kit IMX219 CSI Camera 256GB SSD for Robots ROS2Programming Ubuntu20.04 (Orin NX 16GB Came Adva Kit)	$1,029.99
RPLIDAR A3M1 360 Degree 2D Laser Range Sensor Kit, 15Hz Scan Rate and 25 Meters Distance Ladar Scanner Modulefor Intelligent Obstacle/Robot/Maker Education	$389.99

1.4. Analysis of global, economic, environmental, and social impact

Table 2. Expected benefits and impacts of several contexts

Context	Expected benefits and impacts
Public Health, Safety, and Welfare	The autonomous golf cart can significantly improve safety and overall welfare in controlled environments such as university campuses and golf courses. Autonomous systems reduce the risk of accidents caused by human error, which

	is responsible for approximately 90% of traffic incidents globally [5]. With real-time obstacle detection and avoidance, the golf cart enhances safety for pedestrians and other users in these spaces. Additionally, by using electric power, the system not only reduces air and noise pollution but also promotes public health, contributing to a quieter, cleaner, and more comfortable environment for everyone involved.
Global	The project supports global sustainability efforts by reducing carbon emissions through the use of electric power instead of fossil fuels. The integration of autonomous electric vehicles is aligned with the United Nations Sustainable Development Goals SDG 11: Sustainable Cities and Communities, by enhancing the efficiency and safety of urban transportation [6]. Additionally, the project contributes to SDG 9: Industry, Innovation, and Infrastructure, by fostering advancements in autonomous systems for specialized environments, paving the way for broader global applications in autonomous transportation [6].
Economic	This project offers considerable economic benefits by reducing the reliance on human drivers, thereby lowering operational costs for institutions such as universities, resorts, or golf courses. The use of commercially available components, like the Jetson Orin NX and Arduino microcontrollers, ensures that the system is affordable and scalable for wide deployment. The project also positions itself within the growing autonomous vehicle market, contributing to economic growth and competitiveness as demand for cost-efficient, autonomous solutions continues to rise.
Environmental	The electric-powered autonomous golf cart supports environmental sustainability by contributing to reduced carbon emissions. Studies show that electric vehicles, including autonomous systems, can reduce greenhouse gas emissions by as much as 40% over their lifetimes compared to traditional internal combustion engine (ICE) vehicles [7]. The autonomous system also optimizes routes and avoids unnecessary stops, further improving energy efficiency and reducing environmental impacts such as air pollution and fuel consumption.
Societal	This project has the potential to significantly improve social inclusion by providing reliable and accessible transportation for individuals with limited mobility, particularly in campus or resort settings. By offering an autonomous solution, the project addresses the needs of individuals who may not have easy access to conventional transportation. The success of this project also fosters technological innovation and can inspire further research in the field of autonomous systems, particularly among students and researchers in engineering and robotics.

1.5. Market Research and Business Viability

According to Zion Market Research [8], the global autonomous vehicle market was valued at approximately USD 50.63 billion in 2023 and is expected to reach around USD 251.61 billion by 2032, growing at a Compound Annual Growth Rate (CAGR) of about 19.5% between 2024 and 2032. This significant growth is driven by advancements in technology, increasing investments in research and development, and the rising demand for autonomous solutions across several sectors. Specifically, the market for low-speed autonomous vehicles suitable for controlled environments is projected to expand as organizations recognize the benefits of implementing such systems. The adoption of autonomous golf carts and shuttles in campuses, resorts, and similar settings is expected to contribute significantly to this growth.

The key factors fueling the market growth include:

- Technological Advancements: Improvements in AI, sensor technology, and machine learning algorithms have made autonomous navigation more reliable and efficient.
- Labor Cost Reduction: Autonomous vehicles help organizations reduce expenses associated with hiring and training drivers.
- Safety Improvements: Enhanced safety features reduce the risk of accidents caused by human error, which is crucial in environments with high pedestrian traffic.
- Environmental Sustainability: Electric autonomous vehicles contribute to reduced carbon emissions, helping organizations meet sustainability goals.

The autonomous golf cart system targets a diverse range of customers, particularly institutions and organizations that operate in controlled environments where short-distance, low-speed transportation is essential. The primary customer segments are:

University Campuses

University campuses, such as Qatar University (QU), represent one of the key target markets for the autonomous golf cart system. These campuses often span large areas, with students, faculty, and staff needing frequent transportation between lecture halls, research buildings, dormitories, and other facilities. Autonomous systems offer universities a solution to reduce operational costs by eliminating the need for human drivers, while also improving campus safety and sustainability by utilizing electric-powered vehicles.

Demographics:

- Primary Users: Students (ages 18-30), faculty members, administrative staff, and visitors.
- Key Needs: Efficient, safe, and eco-friendly transportation that reduces walking time across campus while minimizing environmental impact.
- Challenges Addressed: Long walking distances between buildings, transportation bottlenecks during peak hours, and a focus on reducing the carbon footprint of campus operations.

Resorts and Golf Courses

The hospitality sector, including resorts and golf courses, is another key customer base for autonomous golf carts. These environments require smooth, comfortable transportation for guests between several points, such as hotel rooms, recreational areas, golf courses, or dining facilities. Autonomous vehicles provide a convenient, tech-savvy option that can enhance the guest experience while lowering operational costs for resort and golf course operators.

Demographics:

- Primary Users: Resort guests (typically aged 25-60) and golfers who value convenience and novelty in their transportation options.
- Key Needs: Safe, comfortable, and readily available transport across resort areas or golf courses, with an emphasis on luxury, convenience, and eco-friendliness.

    - Challenges Addressed: Long distances between amenities, demand for efficient service without requiring staff intervention, and a growing preference for sustainable solutions in the hospitality industry.

In the rapidly evolving autonomous vehicle market, several companies offer solutions tailored to controlled environments such as university campuses, resorts, and golf courses. Understanding the competitive landscape and differentiating our autonomous golf cart system is crucial for market positioning.

Competitors

- Carteav: Specializes in autonomous low-speed electric vehicles designed for controlled environments, including university campuses, resorts, and golf courses. Carteav offers an end-to-end solution encompassing autonomous vehicles, fleet management, wireless charging, and reservation systems. Their focus on safety, efficiency, and user experience positions them as a significant player in the autonomous golf cart market [9].
- T-Buggy® EV: Developed by Oribay Group, T-Buggy® EV is a next-generation autonomous, connected, and electric golf cart designed for a wide range of uses, from golf courses to resorts. It features advanced driver-assistance systems (ADAS), fleet management capabilities, and cloud infotainment, aiming to provide a safe and efficient ride [10].
- Tesla Model Y: A compact electric SUV equipped with advanced driver-assistance features, including Autopilot and Full Self-Driving capabilities. While the Model Y represents significant advancements in consumer autonomous technology, it is primarily designed for personal use and lacks the full autonomy required for controlled environments like university campuses or golf courses. Additionally, its size and design may not be ideal for the specific transportation needs of such settings [11].
- GEM (Global Electric Motorcars): Produces low-speed electric vehicles suitable for resorts and golf courses. However, GEM vehicles are not fully autonomous, requiring human operation, which may not meet the growing demand for autonomous solutions in these environments [12].

Our autonomous golf cart system distinguishes itself through several key features:

- Cost-Effectiveness: Utilizes commercially available components such as the Jetson Orin NX and Arduino microcontrollers, resulting in a more affordable solution compared to competitors like Carteav and T-Buggy® EV, whose integrated systems may involve higher costs.
- Scalability for Niche Applications: Specifically designed for small-scale, controlled environments, our system offers flexibility and adaptability, making it ideal for resorts, golf courses, and university campuses that require personalized transportation solutions.
- Advanced Sensor Integration: Incorporates LiDAR, stereo cameras, and ultrasonic sensors integrated with ROS2 for seamless sensor fusion and real-time obstacle detection, ensuring safe and efficient navigation in complex environments.

- Modular Design: Features a modular architecture that allows for easy customization and scalability, enabling adaptation to several settings and specific customer requirements—a flexibility not commonly offered by larger competitors.

By focusing on these differentiators, the autonomous golf cart system addresses the unique needs of controlled environments, offering a tailored, efficient, and sustainable transportation solution that stands out in the competitive landscape.

Our autonomous golf cart system introduces several innovative features designed to enhance functionality, safety, and user experience. It employs a combination of LiDAR, stereo cameras, and ultrasonic sensors for comprehensive environmental awareness, enabling precise navigation and real-time obstacle detection. Utilizing the Robot Operating System 2 (ROS2), the system facilitates seamless communication between hardware components and software algorithms, ensuring responsive and reliable autonomous navigation. Its modular design allows for easy customization and scalability, adapting to several settings and specific customer requirements. Powered entirely by electricity, the golf cart promotes environmental sustainability by reducing carbon emissions and ensuring extended operational periods. Additionally, an intuitive user interface offers straightforward control options, including remote operation capabilities, enhancing accessibility and ease of use for operators and passengers alike. By integrating these novel features, our autonomous golf cart system delivers a cutting-edge transportation solution that prioritizes safety, efficiency, and adaptability to meet the evolving needs of controlled environments.

To successfully introduce our autonomous golf cart system to the market and ensure its competitiveness, we will implement a comprehensive strategy based on the four Ps of the marketing mix: Product, Price, Place, and Promotion [13]. The product is an autonomous golf cart designed specifically for controlled environments such as university campuses, resorts, and golf courses. It features advanced sensor integration, including LiDAR, stereo cameras, and ultrasonic sensors, enabling precise navigation and real-time obstacle detection. The system utilizes the Robot Operating System 2 (ROS2) for efficient communication between components, ensuring responsive and reliable autonomous operation. Its modular architecture allows for easy customization and scalability, adapting to several settings and specific customer requirements. Powered entirely by electricity, the golf cart promotes environmental sustainability by reducing carbon emissions and ensuring extended operational periods. An intuitive user interface offers straightforward control options, including remote operation capabilities, enhancing accessibility and ease of use for operators and passengers alike. The pricing strategy aims to balance affordability with value, making the system accessible to a broad range of customers while ensuring profitability. By utilizing commercially available components such as the Jetson Orin NX and Arduino microcontrollers, we can offer a cost-effective solution compared to competitors like Carteav and T-Buggy® EV, whose integrated systems may involve higher costs. We will conduct market research to determine optimal pricing that reflects the system's advanced features and aligns with customer expectations. We will focus on deploying our autonomous golf cart system in environments where there is a clear need for efficient, safe, and eco-friendly transportation solutions. This includes university campuses, resorts, and golf courses that require personalized transportation options. By targeting these specific markets, we can ensure that our product is available where it is most needed and can have the greatest impact. Promotional strategy will leverage several channels to build awareness and generate interest in our autonomous golf cart system. Social media platforms will be utilized to showcase the system's unique features and benefits, engage with potential customers, and share testimonials from pilot deployments. Public relations efforts will highlight the system's contributions to environmental sustainability and technological innovation, positioning it as a forward-thinking solution developed by students at Qatar University to address local transportation challenges. Collaborations with local businesses and community organizations will further enhance the system's visibility and credibility within the Qatari market.

1.6. Justification of the problem as a complex engineering problem

Developing an autonomous golf cart system for controlled environments such as university campuses, resorts, and golf courses presents a diverse engineering challenge. This effort involves balancing conflicting requirements, integrating multiple subsystems, utilizing multidisciplinary expertise, and considering broader technological, economic, environmental, and social impacts.

1.6.1 Conflicting Requirements and Engineering Judgment

The project must harmonize competing demands for safety, cost-effectiveness, and functionality. The system needs to be affordable to encourage widespread adoption while incorporating advanced features like precise navigation and real-time obstacle detection. Achieving this balance requires careful engineering judgment to optimize trade-offs between cost, performance, and user experience. Section 5.3 of this report will explore several design alternatives and the rationale behind the chosen solution, reflecting on related work that presents different approaches to similar problems.

1.6.2 Multiple Components or Subsystems

The autonomous golf cart system comprises several interconnected hardware and software components. Hardware elements include the vehicle platform, sensor suite (LiDAR, stereo cameras, ultrasonic sensors), and control units. The software encompasses navigation algorithms, sensor fusion processes, and user interface applications. Ensuring seamless integration and coordination among these subsystems is crucial for the system's overall performance and reliability.

1.6.3 Multidisciplinary Nature

Developing this system necessitates expertise across several disciplines, including mechanical engineering for designing and optimizing the vehicle's structural components and propulsion systems; electrical engineering for integrating sensors, control units, and power management systems; computer science for developing algorithms for autonomous navigation, sensor fusion, and real-time data processing; and human-computer interaction for designing intuitive user interfaces for operators and passengers. Collaboration among these fields is essential to create a cohesive and effective solution.

1.6.4 Technological Solution and Its Impacts

Implementing an autonomous golf cart system introduces advanced technology into everyday environments, offering potential benefits and challenges. Socially, the system aims to enhance mobility and convenience for users, including students, faculty, resort guests, and golfers, and must be user-friendly and accessible to diverse populations. Economically, by reducing the need for human drivers, the system can lower operational costs for institutions and businesses; however, it may also affect employment in transportation roles. Environmentally, as an electric vehicle, the system contributes to reduced emissions and aligns with sustainability goals, though the environmental impact of manufacturing and energy consumption must be assessed. Globally, success in this project could serve as a model for similar applications worldwide, influencing the adoption of autonomous transportation solutions in several sectors.

1.6.5 Innovation and Real-Life Application

This project embodies real-world engineering challenges where innovation is crucial to meet intricate design constraints and stakeholder requirements. It demands not only technological advancements but also a deep understanding of operational contexts to ensure the solution is practical and deployable in real-world scenarios.

In summary, developing an autonomous golf cart system for controlled environments is a complex engineering problem that requires a multidisciplinary approach, careful consideration of conflicting requirements, and an understanding of its broader impacts. The project's success depends on effectively navigating these complexities to deliver a solution that is technologically sound, economically viable, environmentally responsible, and socially beneficial.

2. Background and related work

2.1. Background

Autonomous vehicles are transforming transportation by allowing cars to operate without direct human interaction. These vehicles rely on a combination of sensors, artificial intelligence, and control systems to perceive their environment, make decisions, and navigate safely. According to the SAE (Society of Automotive Engineers), autonomous driving is classified into six levels, from Level 0 (no automation) to Level 5 (full autonomy) [14]. Our project targets Level 4, where the vehicle can handle all driving tasks in specific environments without human input.

While most autonomous vehicle development has focused on road-going cars, controlled environments like university campuses, golf course, offers a unique opportunity to implement and test these technologies. Autonomous golf carts are ideal for such settings, providing low-speed, safe, and efficient transport in predictable environments.

In this project, we aim to develop an autonomous golf cart for campus use. The vehicle will integrate key technologies like LiDAR, stereo cameras, ultrasonic sensors, and GPS for navigation and obstacle detection, aiming to operate independently in predefined campus areas.

The development of an autonomous golf cart requires several key components that work together to enable the vehicle to perceive its environment, make decisions, and navigate safely. The primary systems include sensors, control systems, and communication devices, all of which are essential for achieving Level 4 autonomy.

- Distance LiDAR: LiDAR (Light Detection and Ranging) is used to detect objects around the golf cart and measure their distance accurately. By emitting laser pulses and calculating the time it takes for them to return, the LiDAR creates a detailed point cloud of the surrounding environment. This allows the cart to detect obstacles, determine their distance, and avoid collisions.

- Intel® RealSense™ D435i depth camera: The D435i camera is responsible for identifying objects detected by the LiDAR. It provides 3D imagery by capturing input from two cameras, allowing the system to differentiate between objects such as pedestrians, other vehicles, and static obstacles. This combination of LiDAR and camera enables both object detection and recognition, improving the cart's ability to respond to different situations.
- Ultrasonic Sensors: These sensors are used for short-range obstacle detection, complementing the LiDAR and camera systems. Ultrasonic sensors provide accurate distance measurements in close-range scenarios, such as when the cart is parking or maneuvering in tight spaces.
- Arduino UNO: The Arduino UNO microcontroller plays a crucial role in handling data from the sensors and controlling the golf cart's subsystems. It processes sensor input and manages tasks like motor control and communication between the cart's components.
- High Torque Stepper Motor: This motor is used for precise control of the cart's steering and braking systems. With its high torque, it ensures accurate maneuverability, which is critical for navigating tight or crowded areas within the campus.
- GPS: The GPS module provides real-time location data, allowing the cart to follow predefined routes across the campus. Accurate GPS positioning is essential for ensuring that the cart stays on track and avoids restricted areas.

2.2. Related work

[15] outlines how ROS2 (Robot Operating System 2) provides a flexible and robust framework for developing autonomous vehicles. The architecture leverages ROS2's real-time communication and distributed processing capabilities to manage sensor inputs, decision-making, and control. ROS2's decentralized nature and improved middleware allow for reliable data exchange between components like LiDAR, cameras, and control systems, ensuring that critical tasks like obstacle detection and path planning occur in real-time. In our project, we adopt a similar approach by using ROS2 to integrate and synchronize data from multiple sensors (LiDAR for distance measurement and stereo cameras for object identification). This architecture enables modular development, where individual nodes handle specific tasks such as sensor fusion, navigation, and motor control, ensuring real-time performance and scalability for the autonomous golf cart operating in a campus environment.

[16] highlights that one of the key barriers to the widespread adoption of autonomous vehicles is the issue of public trust. Despite advancements in technology, many people remain skeptical about the safety and reliability of autonomous systems. Concerns around system malfunctions, the ability to handle unexpected situations, and the potential for accidents contribute to a general reluctance among users to fully embrace these vehicles. Additionally, factors such as data privacy and the security of the systems further erode trust, as individuals fear unauthorized access or cyber-attacks. In our project, we aim to address these concerns by developing the autonomous golf cart for a controlled campus environment, where the lower complexity and well-defined routes minimize risks. By ensuring high levels of safety and transparency in operation, our project can serve as a step toward building public confidence in autonomous systems.

[17] provides a comprehensive overview of the technologies and methods used to equip autonomous vehicles with the ability to perceive their environment. It highlights key sensors such as LiDAR, cameras, and ultrasonic sensors, which are crucial for object detection, localization, and environmental mapping. These technologies align closely with our project, where LiDAR is used for detecting objects and measuring distance, while stereo cameras handle object identification. Additionally, the paper discusses the integration of these sensors through sensor fusion techniques, which is a core component of our system's real-time decision-making. The review also addresses challenges such as sensor performance in adverse conditions, an important consideration for ensuring the reliability of our autonomous golf cart in several campus environments. By utilizing insights from this systematic review, we aim to build a robust perception system tailored to the specific needs of a controlled environment setting.

[18] offers valuable insights into the design and implementation of an autonomous golf cart, which aligns closely with the goals of our project. The Florida Polytechnic system integrates sensors such as LiDAR, stereo cameras, and GPS to enable the vehicle's perception and navigation capabilities. This setup, combined with low-level control via microcontrollers for steering and braking, mirrors the architecture we are adopting for our golf cart. Moreover, the paper outlines a cooperative multitasking approach to software design, which ensures efficient communication between the vehicle's microcontroller and higher-level systems. By building on the lessons learned from this project, particularly in sensor integration and control algorithms, our design will focus on similar strategies but adapted for our specific environment and hardware choices.

[19] provides a comprehensive overview of the process involved in converting a conventional electric golf cart into a fully autonomous vehicle, using cost-effective components and accessible technologies. The authors detail the use of vision-based camera systems, ultrasonic sensors, and infrared sensors for lane detection and obstacle avoidance, which align closely with our approach of using LiDAR for object detection and cameras for object identification. The paper also explores the integration of microcontrollers for managing sensor data and controlling the cart's movement, a concept mirrored in our project where we use Arduino for real-time control of the golf cart's steering, acceleration, and braking systems. By following a low-cost design approach, this work serves as a valuable reference for developing the control algorithms and sensor fusion techniques necessary for achieving autonomy in our golf cart project.

[20] developed at Ohio Northern University provides insights into the design and implementation of an autonomous golf cart for campus transport. The system employs a National Instruments (NI) CompactRIO real-time controller to manage several components, including a LIDAR sensor for obstacle detection, GPS/INS for navigation, and servo motors for steering control. Similar to our approach, this project uses LIDAR for detecting objects and avoiding collisions while navigating predefined paths. The integration of sensors and real-time control through the CompactRIO closely mirrors our use of industrial computer for communication between components. This paper serves as a valuable reference, particularly in relation to the path-planning algorithms and obstacle avoidance strategies, which align with our focus on operating in a controlled environment like a campus.

[21] explores the integration of IoT devices to enhance the functionality and connectivity of autonomous golf carts. The authors discuss how the use of IoT improves communication between the vehicle and cloud-based systems, enabling real-time data transmission and remote monitoring. This is closely aligned with our project, where we plan to use a GSM Router for similar purposes. Additionally, the paper outlines the use of GPS and ultrasonic sensors for navigation and obstacle detection, which mirrors our use of LiDAR for object detection and stereo cameras for object identification. The incorporation of a mobile application to control and track the golf cart provides a novel interface for user interaction, which could inspire future enhancements to our project. Overall, this work serves as a valuable reference for integrating IoT technology and real-time control into our autonomous golf cart design.

[22] provides a comprehensive review of the challenges and future directions in autonomous vehicle development, highlighting the critical importance of real-time obstacle detection and tracking in ensuring safe navigation. The article emphasizes the need for robust perception systems capable of handling dynamic environments where objects, such as pedestrians or small vehicles, may appear unexpectedly. In our project, these insights are particularly relevant as the autonomous golf cart will operate in open spaces like campuses and golf courses. By employing LiDAR to detect objects around the cart and stereo cameras to classify those objects, the system addresses these challenges through sensor fusion techniques. utilizing ROS2 to synchronize data from the LiDAR and cameras ensures real-time path planning and obstacle avoidance, aligning with the principles and methodologies discussed in the paper.

[23] explores the ethical dilemmas, social challenges, and technological opportunities associated with autonomous vehicles, emphasizing the importance of building public trust for widespread adoption. Ethical concerns, such as how autonomous systems prioritize decisions in critical situations, and social challenges, including fears over job displacement and the reliability of these systems, have significantly impacted public perception. These issues highlight the need for transparent and accountable system design to address user concerns about safety and reliability. In our project, the autonomous golf cart is designed for controlled environments, such as university campuses, where the lower complexity and well-defined routes minimize risks and ethical dilemmas. By demonstrating safe and reliable operation in such settings, our project aims to address social skepticism and contribute to building public confidence in autonomous technologies.

[24] introduces a hierarchical framework for autonomous navigation in smart environments, emphasizing modular and layered architectures for managing complex tasks. This approach aligns with our implementation of ROS2 (Robot Operating System 2), which facilitates real-time communication and distributed processing across system components. ROS2's decentralized structure and robust middleware ensure seamless integration of LiDAR, stereo cameras, and control systems, enabling critical functions such as obstacle detection, path planning, and decision-making. By adopting a similar hierarchical design, our autonomous golf cart project optimizes sensor fusion, navigation efficiency, and scalability, ensuring reliable operation in controlled environments like university campuses.

Table 3 Comparison

Citation	Year	Key Focus	Sensors and Technological Methods	Challenges Addressed	Level of Economy	Level of Autonomy
[15]	2020	ROS2 framework for modular integration	LiDAR, Stereo Cameras Real-time communication ROS2 Middleware	Real-time performance and scalability	Real-time performance and scalability	Level 4
[16]	2022	Public trust and safety in autonomous vehicles	N/A, Risk reduction strategies	Public skepticism and data privacy	Focused on trust rather than economic factors	N/A
[17]	2019	Sensor fusion technologies and reliability	LiDAR, Stereo Cameras, Ultrasonic Sensors and Sensor fusion	Reliability under adverse conditions	Moderate cost due to advanced sensor fusion	N/A

			techniques			
[18]	2021	Design of an autonomous golf cart at Florida Polytechnic	LiDAR, Stereo Cameras, GPS and Cooperative multitasking	Sensor integration and control	High, focused on academic projects	Level 3
[19]	2022	Cost-effective conversion of conventional golf cart	Vision Cameras, Ultrasonic Sensors, Infrared Sensors and Sensor management with microcontrollers	Achieving autonomy cost-effectively	Very high, cost-effective methods highlighted	Level 3
[20]	2013	Real-time control using CompactRIO	LiDAR, GPS/INS, Servo Motors and Path-planning algorithms	Obstacle detection, predefined path navigation	Moderate, dependent on CompactRIO system	Level 2
[21]	2021	IoT integration for real-time monitoring	LiDAR, Stereo Cameras, GPS, IoT Devices and IoT and cloud-based system integration	Enhanced connectivity and remote monitoring	High, leveraging IoT for cost reduction	Level 2
[22]	2024	Challenges in dynamic environments	N/A and Obstacle detection and tracking	Dynamic obstacle handling	Low, advanced sensor fusion is costly	N/A
[23]	2023	Ethical and social aspects of autonomous vehicles	N/A and Transparent system design	Ethical concerns and public trust	N/A	N/A
[24]	2023	Hierarchical frameworks for navigation	LiDAR, Stereo Cameras, Control Systems and Modular, layered architectures	Scalability, navigation efficiency	Moderate, balance between cost and scalability	Level 4
Our work	2024	Cost-effective and scalable autonomous golf cart for controlled environments	LiDAR, Stereo Cameras, Ultrasonic Sensors, GPS and Real-time ROS2 communication, sensor fusion, path planning	Cost-efficient sensor integration, scalability, user trust	Very high, emphasizes cost-efficiency with modular design	Level 4

Based on the comparison table, our project stands out as one of the best solutions among the related works. It combines cost-effectiveness, high autonomy, and advanced technology to address the specific needs of controlled environments. Unlike many other projects, which either focus on general applications or involve high-cost components, our project utilizes readily available, affordable hardware such as LiDAR, stereo cameras, and ultrasonic sensors. These are seamlessly integrated using ROS2 middleware, ensuring real-time performance and robust decision-making.

Our project's Level 4 autonomy is another distinguishing factor. While some related works only achieve partial or conditional autonomy, our golf cart operates fully independently within defined settings, providing a practical and reliable solution for environments like campuses and golf courses. The system's modular design further enhances its adaptability, making it easy to maintain, upgrade, and scale for future applications.

Additionally, our focus on sustainability and public trust sets our project apart. The electric-powered design aligns with global sustainability goals, reducing carbon emissions while offering an eco-friendly transportation option. By operating in a controlled and safe environment, our system minimizes risks and addresses concerns about the reliability and safety of autonomous vehicles, fostering greater public acceptance.

In comparison to the related works, our project excels in balancing affordability, technical capability, and practical application. It represents a forward-thinking approach to autonomous transportation, offering a scalable, sustainable, and innovative solution tailored to real-world needs.

3. Requirements analysis

3.1. Functional requirements

The functional requirements define what the autonomous golf cart system must accomplish to operate effectively and safely within its environment. This includes key data inputs, core processes, workflows, outputs, and primary users of the system.

3.1.1 Description of Data/Signals

- LiDAR Data: Provides 2D spatial information to help the system detect and measure distances to surrounding objects, essential for collision avoidance and path planning.
- Stereo Camera Data: Captures visual and depth information, allowing the cart to classify obstacles as either 3D objects (like pedestrians or other vehicles) or 2D objects (such as painted lines or signage).
- Ultrasonic Sensor Data: Offers close-range detection for immediate obstacles, helping to prevent collisions at close distances.
- GPS Data: Supplies real-time location information, enabling accurate navigation and localization within defined areas.

3.1.2 Operations/Processes

- Object Detection and Classification: The system processes data from LiDAR and the stereo camera to detect and categorize objects, distinguishing between stationary and moving obstacles.
- Path Planning and Navigation: Using sensor data, the system calculates and continuously updates an optimal path based on its destination and environmental conditions.
- Obstacle Avoidance: When an obstacle is detected, the system decides whether to stop, slow down, or navigate around it based on the type, location, and distance of the obstacle.

- Speed and Steering Control: The motors control acceleration and turning, adjusting based on real-time sensor inputs to maintain safety and stability during navigation.

3.1.3 Workflows

- Startup Workflow: The system powers on, initializes all sensors, and conducts self-checks to confirm that each component is operational.
- Navigation Workflow: Once a destination is set, the cart continuously gathers data, calculates an efficient route, and sends commands to control its speed and direction, adjusting dynamically as the environment changes.
- Obstacle Handling Workflow: If an obstacle is detected, the system evaluates its characteristics and determines the best course of action (e.g., stop, slow down, navigate around).
- Shutdown Workflow: When the destination is reached, the cart powers down non-essential components to conserve battery and concludes operations.

3.1.4 System Outputs

- Movement Commands: Signals are sent to the motors to control speed and steering based on the calculated path and obstacle data.
- Status Reports: Provides updates on the cart's position, detected objects, and system health for real-time monitoring and diagnostics.
- Data Logs: Records sensor data, actions, and system decisions for diagnostics, performance analysis, and system improvements.

3.1.5 Main Users

- End Users: Responsible for starting, monitoring, and managing the cart's operations. They can set destinations, monitor the cart's progress, and stop or pause it as needed.
- Maintenance Technicians: Handle troubleshooting and maintenance, accessing data logs and performing diagnostics to ensure the system functions correctly. They oversee system health without the need for real-time remote monitoring.

These functional requirements focus on providing a safe, efficient, and user-friendly autonomous cart system suited for controlled environments like campuses and golf courses.

3.2. Design constraints

In developing the autonomous golf cart, several constraints were identified to ensure that the project meets technical, economic, environmental, social, ethical, health, and safety standards. These constraints act as boundaries that define what the system must adhere to for successful implementation and operation. Each constraint was determined with the guidance of the design standards and serves as a guideline for both development and testing.

Table 4. Design constraints

Constraint	Type (Technical/ Socio-Economical)	Relevance to the Project	Evaluation Plan	Success Criteria
Real-Time Responsiveness	Technical	Real-time responsiveness is critical for safe navigation and effective obstacle avoidance. The system must be capable of processing sensor data and responding swiftly to any changes in the environment.	Measure the latency from data input (sensor) to action output (motor control) across several scenarios.	The system should respond within 100 [25] to 220 [26] milliseconds to ensure prompt and safe reactions.
Power Efficiency and Battery Life	Technical/Economic	The system must be power-efficient to operate continuously for several hours without interruption, maximizing its usability in real-world settings.	Conduct endurance tests to evaluate battery life under typical load conditions.	The cart operates for a minimum of 4 hours on a single charge in normal operational settings. [27]
Environmental Impact	Socio-Economic (Environmental)	Since the cart operates in environmentally sensitive areas like campuses and golf courses, it should have minimal ecological impact in terms of noise and emissions.	Select components that are environmentally friendly and test the cart to ensure it meets emission and noise standards.	The cart complies with environmental standards, producing low noise and emissions, making it suitable for campus environments.
Safety and Collision Avoidance	Technical/Safety	Ensuring pedestrian safety is paramount as the cart will operate in public areas. It must reliably detect and avoid obstacles, adjusting speed or stopping as necessary.	Conduct controlled tests to evaluate the cart’s ability to detect and avoid several obstacles at different distances.	The system accurately detects and avoids obstacles within a 3-meter range 95% of the time, reducing the risk of collisions.
Data Security and Privacy	Ethical	If the system collects	Implement encryption for	Data is encrypted and

		operational data, it must be protected to ensure privacy. Securing data is important, especially if monitoring data is stored or transmitted.	data storage and apply access controls.	protected, with no unauthorized access during the testing and deployment phases.

Table 5 Logistical & Organizational Constraints

Constraint	**Type**	**Relevance to the Project**	**Evaluation Plan**	**Success Criteria**
Limited Time Availability	Project Management	Narrow timeframe during second semester affected testing, debugging, and system integration.	Compare actual vs. planned timeline using Gantt chart.	Prioritized critical tasks; worked in parallel where possible.
No Initial Golf Cart Provided	Resource/Logistical	Lack of a physical golf cart early on delayed hardware integration and real-world testing.	Document time lost and adaptation strategies (e.g., simulation).	Focused on module testing; started integration as soon as cart was available.
Lack of Team Members	Human Resource	Team size was small relative to project scope, increasing individual workload.	Task tracking and member contribution logs.	Delegated tasks efficiently based on skillsets.
No Initial Workspace	Logistical	No place to work on hardware integration in the early stages.	Measure impact on testing/setup start dates.	Supervisor provided a workspace mid-semester.
No Financial Sponsorship	Economic	Budget limitations impacted hardware quality, speed of procurement, and available tools.	Cost vs. requirement analysis.	Used off-the-shelf, affordable components and leveraged university resources.

3.3. Design standards

The autonomous golf cart project adheres to several engineering standards and communication protocols to ensure safety, reliability, and efficiency. These standards govern several aspects of the design and implementation, providing a structured approach to the system's development.

Table 6. Engineering Standards used in the project

Standard	**Relevance to the project**
ISO 26262 – Functional Safety	Ensures the functional safety of electrical and electronic systems, which is essential for autonomous operation in public spaces.
ISO 13849 – Safety of Machinery	Sets safety requirements for control systems, especially for critical functions like braking and steering.
ISO 12100 – Safety of Machinery	Covers general safety principles for machinery design, including risk assessment and hazard mitigation.
ISO 212 – Intelligent Transport Systems	Provides guidelines for automated and connected vehicles, supporting interoperability and safe operation.
IEC 61508 – Functional Safety of Electrical/Electronic Systems	Ensures safe design practices for electronic systems, with a focus on handling unexpected faults.
ISO 16750 – Environmental Conditions	Establishes environmental testing standards to verify that the system can operate effectively in varying conditions.
ISO/IEC 27001 – Information Security Management	Provides a framework for securing data, essential for protecting operational data and ensuring user privacy.
ISO 6405 – Symbols for Operator Controls	Standardizes symbols for controls and displays, enhancing user experience and safety.
ISO 7010 – Safety Signs	Sets guidelines for safety signage to improve awareness and promote safe operation within shared spaces.

Table 7. Communication Protocols Used in the Project

Standard	Relevance to the project
CAN Bus (Controller Area Network)	Enables reliable communication between sensors, controllers, and actuators within the vehicle.
ROS 2 (Robot Operating System)	Manages data exchange and communication between sensors, controllers, and actuators in real time, enabling autonomous navigation.
UART (Universal Asynchronous Receiver/Transmitter)	Supports serial communication between components, allowing for efficient wired data transmission within the system.
Ethernet	Provides high-speed, wired communication between critical system components, ensuring low-latency data transfer and stability.

3.4. Professional Code of Ethics

In developing an autonomous golf cart, we must consider a range of ethical, legal, social, and security responsibilities, with attention to the perspectives of diverse stakeholders. Our goal is to ensure that our decisions prioritize safety, privacy, transparency, and social responsibility, in alignment with recognized professional codes of ethics from IEEE and ACM.

Table 8. Engineering Code of Ethics and Professional Practice

Sec. No	Code	Usage and practice addressing an identified issue
IEEE 1.1	Public Safety - Engineers shall hold paramount the safety, health, and welfare of the public.	Safety mechanisms, such as obstacle detection and avoidance, will be prioritized to protect pedestrians and ensure safe operation of the autonomous golf cart.
IEEE 1.3	Honesty - Engineers shall be honest and realistic in stating claims or estimates.	Inform users of the cart's capabilities and limitations, providing clear guidelines on safe and responsible usage in campus environments.
IEEE 3.5	Privacy - Engineers shall respect the privacy of others.	Implement encryption and access controls for data transmitted via the GSM router to protect user privacy and prevent unauthorized access.
ACM 1.2	Avoid Harm - Avoid harm to others.	Safety protocols and thorough testing ensure the cart operates safely without endangering users or bystanders.
ACM 1.6	Respect Privacy - Respect the privacy of others and ensure proper handling of personal data.	Secure data collected during remote monitoring, maintaining confidentiality and restricting access only to authorized personnel.
ACM 2.1	Contribute to Society and Human Well-Being.	The project is designed to contribute to safer transportation within the campus, reducing potential harm to pedestrians through effective safety mechanisms.
ACM 2.9	Respect and Protect the Environment.	Use energy-efficient components and follow sustainable practices to minimize the project's environmental impact, especially in natural or landscaped areas.

3.5. Assumptions

In developing the autonomous golf cart project, several assumptions have been made to guide planning, design, and implementation. These assumptions represent conditions that are expected to hold true throughout the project, though they are not guaranteed. Monitoring these assumptions is critical, as they could affect the project's outcome if proven false.

- Controlled Environment: The autonomous golf cart is assumed to operate in a controlled environment, such as a campus or golf course, where traffic and obstacles are predictable and limited.
- Data Availability and Accuracy: It is assumed that GPS and environmental data will be available and accurate to support effective navigation and localization within defined areas.
- Power Reliability: The power supply, primarily the battery, is assumed to be reliable, providing sufficient energy for at least four hours of continuous operation under typical conditions.
- Favorable Weather Conditions: It is assumed that the system will operate primarily in favorable weather, as adverse weather conditions (such as heavy rain or fog) could impact sensor accuracy and overall system performance [28].
- Stakeholder Cooperation: The project assumes that key stakeholders, including campus administration and maintenance personnel, will provide necessary support for system deployment, maintenance, and troubleshooting.

- User Compliance: It is assumed that users will follow the provided instructions and safety guidelines when interacting with the golf cart, minimizing the risk of unintended misuse or unsafe behavior.

These assumptions serve as foundational elements in the project plan, influencing design decisions and risk management strategies. Each assumption will be monitored to ensure it remains valid, as changes in these conditions may require adjustments to the project scope, timeline, or design.

4. Project Plan

4.1. Project milestones

Table 9. Milestones of the project

Milestone	Description	Deliverables
Initial Project Setup and Research	Define the project scope and conduct an initial literature review on autonomous navigation, focusing on existing technologies and sensor options.	- Project scope and objectives document
- Literature review report		
Background and Related Work	Conduct a detailed study on related work and foundational technologies in autonomous vehicles. Complete these sections in the project report.	- Background and Related Work sections
System Architecture and Component Selection	Design the system architecture and select key components, such as LiDAR, RealSense, and processing units, ensuring compatibility with project objectives.	- System architecture diagrams
- Component list		
Initial Prototype Development	Develop initial prototype by integrating LiDAR and RealSense camera for object detection and classification. Begin sensor fusion techniques.	- Prototype perception module
- Sensor fusion results		
Golf Cart Instrumentation	Mount and integrate all selected hardware onto the golf cart. Conduct tests to validate connectivity and power management.	- Fully instrumented golf cart
- Installation and connectivity report		

4.2. Project timeline

Table 10. Project timeline

Milestone	Task	Description	Assigned Member(s)	Start Date	End Date
1. Initial Project Setup and Research	Define Project Scope	Establish clear objectives and scope for the project.	All Members	27-08-2024	03-09-2024
	Complete Grant Application	Prepare and submit grant proposal for project funding.	All Members	03-09-2024	10-09-2024
2. Research and Documentation	Literature Review	Conduct research on autonomous systems and relevant sensor technologies.	Abdelaziz Shehata	10-09-2024	17-09-2024
	Draft Background and Related Work	Complete the background and related work sections in the report.	Abdulla Tariq, Zabin Al-Dosari	17-09-2024	01-10-2024
3. System Architecture and Component Selection	Develop System Design	Outline system architecture, sensor layout, and integration plan.	All Members	01-10-2024	08-10-2024
	Select and Order Components	Finalize and order necessary components (Jetson, LiDAR, Camera).	All Members	08-10-2024	15-10-2024
4. Initial Prototype Development	Integrate LiDAR	Implement and test LiDAR setup for obstacle detection with Python.	Abdelaziz Shehata	15-10-2024	29-10-2024
5. Sensor Integration and Testing	Sensor Fusion Testing	Develop and test sensor fusion algorithms in ROS2 for combining LiDAR and camera data.	Zabin Al-Dosari	29-10-2024	05-11-2024
	Set Up RealSense Camera	Integrate and test RealSense camera for 3D classification.	Abdulla Tariq	15-10-2024	29-10-2024
	Power and Connectivity Testing	Verify power distribution and connectivity for all hardware components.	Zabin Al-Dosari	12-11-2024	19-11-2024
6. Navigation and Control Algorithm	Path Planning	Implement path planning algorithms for navigating	Abdelaziz Shehata	19-11-2024	03-12-

Development		predefined paths.			2024
	Obstacle Detection Integration	Integrate obstacle detection into the path planning system.	Abdulla Tariq	03-12-2024	10-12-2024
	Emergency Stop Mechanism	Implement emergency stop functionality for safety protocols.	Abdulla Tariq	10-12-2024	17-12-2024
7. Golf Cart Instrumentation	Hardware Mounting	Install all sensors and hardware components onto the golf cart.	All Members	05-11-2024	12-11-2024
8. Final Testing and Optimization	Optimization of Algorithms	Fine-tune navigation, control, and safety algorithms based on testing results.	Abdulla Tariq, Abdelaziz Shehata	14-01-2025	28-01-2025
	Final Field Testing	Conduct final field tests under realistic conditions on campus and golf course paths.	All Members	28-01-2025	18-02-2025
9. Documentation and Presentation Preparation	Final Report Compilation	Complete all project documentation and results.	All Members	18-02-2025	08-03-2025
	Presentation Preparation	Develop and practice final project presentation for stakeholders.	All Members	08-03-2025	15-03-2025

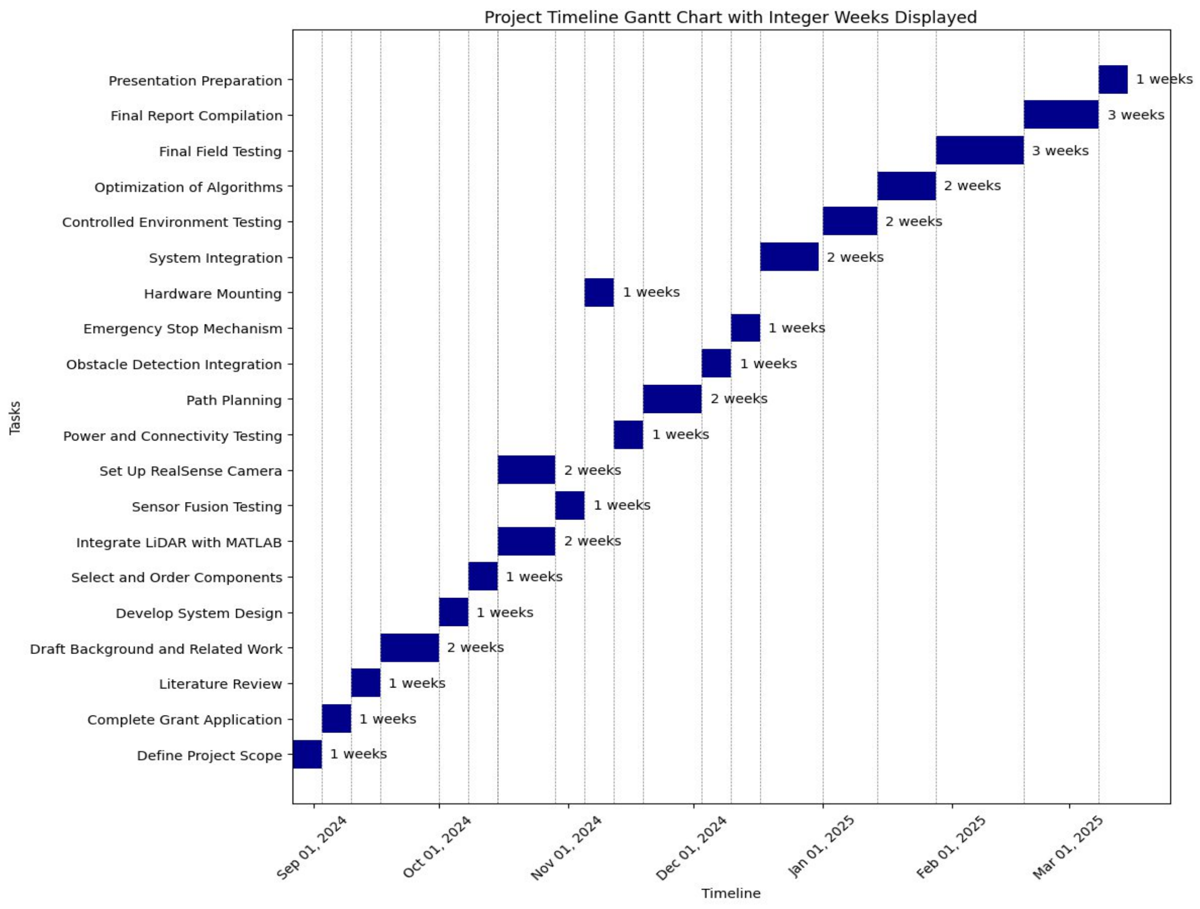


Figure 1 Gant Chart

4.3. Anticipated risks

Table 11. Risks

Risk event	Approach to minimizing the effect on project success
Hardware Failure	Conduct regular maintenance, ensure proper ventilation and power distribution, and keep spare parts available for critical components.
Software Bugs	Implement iterative testing and debugging phases, conduct code reviews, and perform frequent software updates.
Safety Protocol Failure	Test safety protocols rigorously and simulate emergency stop and manual override scenarios in a controlled environment.
Environmental Conditions	Equip the golf cart with multi-sensor fusion and test in varied conditions to ensure reliability.
Component Delays	Order components early in the project timeline and maintain communication with suppliers.
Battery and Power Management Issues	Use high-capacity batteries, monitor power usage, and establish a reliable power management system.
Risk of Accidents during Testing	Ensure controlled testing environments, monitor vehicle closely, and test at low speeds initially.
Communication Failure between Components	Use robust communication protocols, perform regular diagnostics, and implement fail-safe mechanisms.

Public Acceptance and Trust	Demonstrate safety protocols, gather feedback, and show reliable results to build public confidence.
Team Stress and Burnout	Distribute workload evenly, encourage regular breaks, and maintain open communication to address stress issues.
Time Management and Task Overlap	Use project management tools, set clear deadlines, and hold weekly check-ins to track progress and adjust as needed.
Skill Gaps or Learning Curves	Allocate time for training, encourage collaborative learning, and assign tasks according to each member's strengths

5. Solution design

5.1. Solution overview

Our project presents an innovative solution for autonomous transportation within controlled environments such as university campuses, resorts, and golf courses. The proposed system is an autonomous golf cart equipped with advanced sensors, enabling it to navigate predefined routes while detecting and avoiding obstacles in real-time. This system is designed to operate without human intervention, ensuring safe, efficient, and eco-friendly mobility for passengers.

The golf cart integrates state-of-the-art processing technology to manage sensor data, execute navigation algorithms, and ensure smooth coordination among its components. Passengers will interact with the system through a user-friendly interface, simplifying ride requests and destination selection. By automating transportation, the solution addresses the growing need for modern, sustainable, and cost-effective mobility solutions. Its electric-powered design minimizes environmental impact while reducing labor dependency, making it a practical choice for institutions and businesses seeking innovative transportation alternatives.

5.2. High level architecture

Hardware high-level Architecture

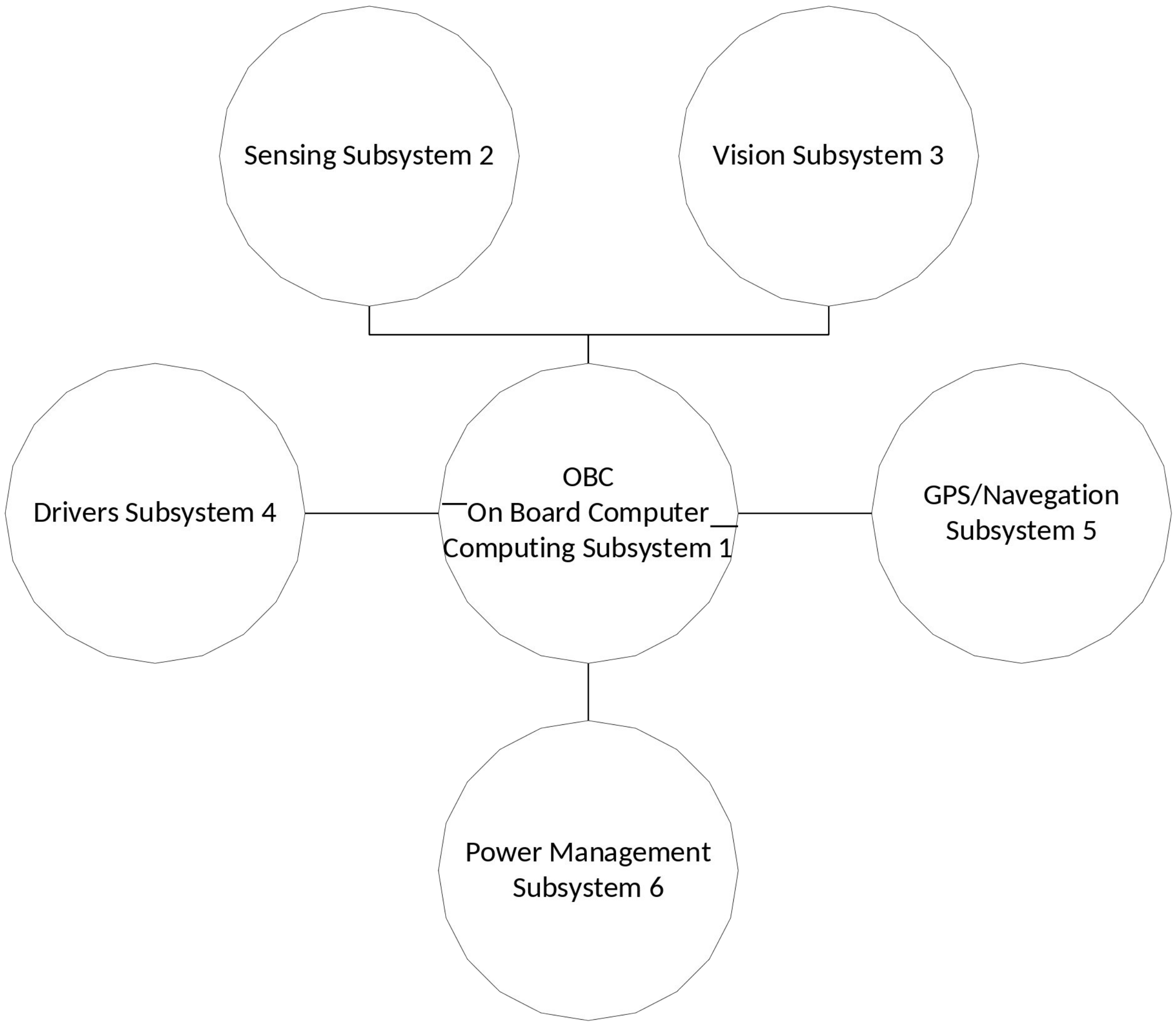


Figure 2 Hardware high-level Architecture

The hardware architecture of the autonomous vehicle system is designed around a modular approach that enhances clarity, flexibility, and future scalability. At the core of the system is the On-Board Computer (OBC), which acts as the central processing unit responsible for fusing sensor data, executing navigation algorithms, and making real-time decisions. The OBC orchestrates the communication and coordination among all subsystems to enable smooth and intelligent autonomous operation.

Surrounding the OBC are five main subsystems, each tasked with a specific role in the overall functionality of the vehicle. The Sensing Subsystem is responsible for collecting proximity and environmental data through various sensors, such as ultrasonic and RP LiDAR, which assist in long/close-range obstacle detection and collision avoidance. Complementing this is the Vision Subsystem, which employs depth cameras and computer vision techniques to enable object detection, classification, and depth estimation. These two subsystems work together to build a comprehensive understanding of the surrounding environment.

The Drivers Subsystem handles the physical actuation of the vehicle. It receives movement and steering commands from the OBC and translates them into electrical signals to control motors, wheels, and servos. Simultaneously, the GPS/Navigation Subsystem provides the system with global positioning and orientation data. This subsystem supports localization, route planning, and waypoint navigation, especially when operating in open environments.

To ensure continuous and stable operation, the Power Management Subsystem distributes electrical power to all hardware components, including the sensors, OBC, and motor drivers. It also ensures that voltage and current levels are well-regulated to prevent system failures. At the heart of the architecture, the Computing Subsystem (OBC) integrates all sensor inputs and executes the software stack, including Robot Operating System 2 (ROS2), perception algorithms, and control logic.

This modular architecture allows for a more abstract and hardware-independent system design, facilitating ease of testing, component upgrades, and reusability in various applications. It also supports the project's shift to a smaller-scale platform while preserving the original functional goals of autonomous navigation, obstacle detection, and intelligent path planning.

Software high-level Architecture

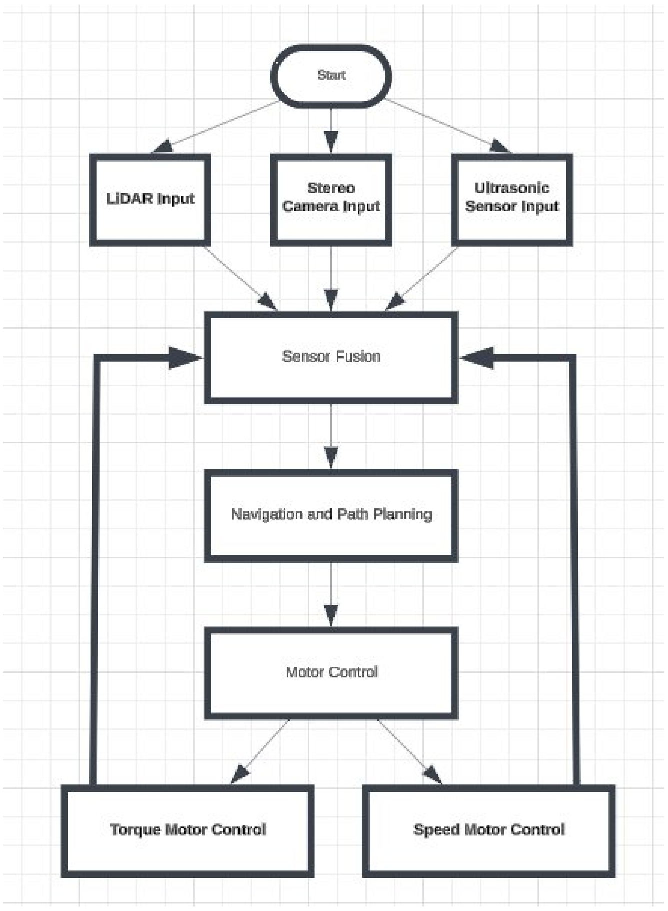


Figure 3 Software high-level Architecture

The software architecture for the autonomous golf cart system is meticulously designed to manage data flow, process sensor inputs, and control navigation in real-time. This modular and layered design ensures smooth integration of hardware and software components, utilizing ROS2 for efficient communication and synchronization among the modules. The architecture begins with sensor input modules, which capture raw environmental data and publish it as ROS2 topics for subsequent processing. The three primary sensors, LiDAR, stereo camera, and ultrasonic sensors, operate independently to provide continuous streams of data. The LiDAR module captures detailed point cloud data for long-range obstacle detection, the stereo camera module identifies objects and estimates distances for precise recognition, and the ultrasonic sensor module provides close-range detection to enhance safety during navigation.

At the core of the system lies the Sensor Fusion Module, which integrates data from all input modules into a comprehensive environmental map. This module subscribes to ROS2 topics, synchronizing data streams and merging them into a unified representation of the surrounding environment. The sensor fusion process eliminates redundancy, reduces noise, and ensures high accuracy for obstacle detection, enabling the system to have a reliable and cohesive understanding of its surroundings.

The Navigation and Path Planning Module builds on the environmental map created by the sensor fusion process. It uses advanced algorithms to calculate the optimal route for the golf cart, dynamically adjusting the path to avoid both static and dynamic obstacles. By continuously updating its decisions based on real-time sensor data, this module ensures that the cart navigates safely and efficiently even in complex and changing environments.

The final layer of the software architecture is the Motor Control Module, which translates navigation commands into actionable instructions for the propulsion system. This module sends control signals to the torque motor and speed motor controllers, regulating the cart’s movement and ensuring precise speed and direction adjustments. A feedback loop, maintained using ROS2, allows the system to monitor performance and refine motor commands dynamically, ensuring smooth operation.

Each module in the architecture interacts seamlessly with the others through ROS2's real-time messaging capabilities. Sensor data flows from the input modules to the Sensor Fusion Module, which then informs the Navigation and Path Planning Module to generate movement commands. These commands are executed by the Motor Control Module, completing the cycle of autonomous operation. This software architecture is designed to be modular and scalable, supporting future enhancements or the integration of additional functionalities. Its real-time capabilities and robust design ensure that the golf cart performs reliably in diverse controlled environments, offering a safe and efficient autonomous transportation solution.

The autonomous vehicle system is built on a tightly integrated hardware and software architecture. The hardware architecture follows a modular design centered around the On-Board Computer (OBC), which communicates with dedicated subsystems for sensing, vision, navigation, actuation, and power management. This modular setup enhances system flexibility and simplifies integration. On the other hand, the software architecture, built using ROS2, mirrors this modularity by organizing the software into layers that handle sensor input, data fusion, navigation, and motor control. Data from the hardware sensors is processed, allowing the system to generate real-time navigation commands that are translated into precise motor actions. Together, the hardware and software architectures form a cohesive system capable of real-time autonomous operation in dynamic environments.

5.3.Design alternatives

This document explores four design alternatives for an autonomous golf cart. Each design integrates a level of autonomy tailored to specific features and use cases. The designs are compared across key parameters to support decision-making.

5.3.1 Design Alternatives

1. Enclosed with AC Autonomous Golf Cart
   a. Designed to enhance comfort in extreme temperatures with an enclosed cabin and air conditioning. The autonomous system controls climate and provides safe navigation, making it suitable for hot climates.
2. Solar Panel Roof Autonomous Golf Cart
   a. Features solar panels to supplement battery power, extending operational time in sunny conditions. The autonomy includes energy management to prioritize solar charging, enhancing eco-friendliness.
3. Special Needs Autonomous Golf Cart
   a. Customized for accessibility with features like low-floor access and ramps, tailored for passengers with mobility challenges. Autonomous features focus on safety and smooth navigation.
4. Standard Autonomous Cart
   a. A baseline autonomous design focused solely on navigation and obstacle avoidance. Cost-effective and suitable for general use, without additional comfort or accessibility features.

Table 12. Comparative Analysis of Design Alternatives

Parameter	Enclosed with AC	Solar Panel Roof	Accessible for Special Needs	Standard Autonomous Cart
Cost	High	Moderate	High	Low
Ease of Use	High (automated climate control)	Moderate (energy management autonomy)	High (autonomous accessibility features)	High (simple and intuitive)
Marketability	High (comfort-focused automation)	High (eco-friendly, self-sustaining)	Niche (targeted at accessibility market)	Moderate (general-use appeal)
Ease of Manufacturing	Moderate	High	Moderate	High (minimal features)
Development Time	Moderate	Low	High	Low
Autonomous Functionality	Climate control, navigation, safety	Energy management, navigation, safety	Accessibility features, safety, navigation	Basic navigation and obstacle avoidance

5.3.2 Evaluation and Selection of Final Design

Based on the comparative table and project goals, each design has its strengths and weaknesses:

- The Enclosed with AC option is ideal for environments with extreme temperatures, providing comfort but at a higher cost.
- The Solar Panel Roof design is an eco-friendly option that offers extended operational time in sunny conditions.
- The Special Needs design addresses accessibility, making it suitable for users with mobility challenges.
- The Standard Autonomous Cart is a basic, cost-effective design for general use. After evaluating each design, the selected design should balance the project's specific needs, budget, and environmental requirements.

Based on a comprehensive analysis of the project requirements, available resources, and the parameters highlighted in the table, the Standard Autonomous Cart emerged as the optimal choice for our autonomous vehicle system. This decision aligns with the project's focus on simplicity, cost-effectiveness, and practical implementation in controlled environments such as university campuses and golf courses.

5.4. Hardware/software used

The autonomous golf cart system relies on a carefully selected combination of hardware and software components to achieve its functional requirements, including real-time navigation, obstacle detection, and autonomous operation. Each component was chosen based on its specific role in the system, compatibility with other components, and suitability for supporting autonomous functionalities.

Table 13. Needed HW/SW

HW/SW details	Platform	Justifications
Jetson Orin Development Kit	Linux	Serves as the primary processing unit for AI tasks, object recognition, and real-time sensor data analysis, providing high computational power for autonomous operations.
RPLIDAR A3M1	Python	Offers accurate LiDAR sensing capabilities to map surroundings and detect obstacles, crucial for safe navigation.
Intel RealSense D435i Stereo Camera	Intel RealSense SDK	Provides depth perception and object classification abilities, which are essential for distinguishing between 3D objects and 2D surfaces.
Arduino UNO r3/r4	Arduino IDE	Controls the motors for acceleration and rotation, receiving commands from the Jetson Orin to execute movement decisions.
High-Torque Motor	Arduino IDE	Ensures precise rotational control of the

		golf cart, enabling it to navigate turns smoothly and reliably.
Accelerator Motor (both drivers)	Arduino IDE	Regulates the speed of the cart, responding dynamically to sensor inputs for safe speed control.
Ultrasonic Sensors	Arduino IDE	Detects close-range obstacles, acting as an additional safety measure for immediate obstacle detection.
GPS Module	Arduino IDE	Provides location data for path tracking and navigation, enabling the system to maintain a defined route.
ROS 2 (Robot Operating System)	Framework	Manages real-time communication between components, allowing for synchronized data flow and command execution.
Python	Software	Used to configure and test the LiDAR sensor, providing a flexible environment for calibration and data visualization.
Intel RealSense SDK	Software	Integrates the stereo camera with the Jetson Orin, enabling depth data processing and object classification.
Python	Software	Primary language for writing ROS 2 nodes and processing scripts, providing flexibility for data handling and model integration.
C++	Software	Used for performance-intensive tasks in ROS 2, such as motor control and real-time sensor data processing.
Ubuntu (Linux OS)	Software	The operating system on the Jetson Orin, providing a stable and compatible platform for ROS 2 and other software.
GitHub	Software	Facilitates version control and collaborative project management, ensuring efficient code tracking and team coordination.
Arduino IDE	Software	Used to program the Arduino microcontrollers for managing sensor inputs and motor control. It provides a user-friendly interface, supports C/C++/python development, and offers real-time debugging tools, making it ideal for embedded systems integration.
Processing GUI	Software	Used to create a simple graphical user interface (GUI) for visualizing sensor data and monitoring the system in real time. It supports serial communication with the Arduino, allowing for easy data interpretation and debugging during development and testing phases.

5.5.Hardware design

5.5.1 Processing Unit

Theory Behind Technological Approaches

The NVIDIA Jetson Orin NX is a compact, high-performance AI computing module designed for autonomous machines. It delivers up to 100 TOPS (Tera Operations Per Second) of AI performance, enabling the execution of multiple concurrent AI inference pipelines. This capability is crucial for real-time processing of data from several sensors, ensuring responsive and intelligent system behavior. Equipped with a 1024-core NVIDIA Ampere architecture GPU and 32 Tensor Cores, the Jetson Orin NX excels in parallel processing tasks, making it ideal for complex computations required in sensor fusion and AI inference. The module features an 8-core Arm Cortex-A78AE v8.2 64-bit CPU, providing robust processing power for managing system operations and executing control algorithms efficiently. With up to 16GB of 128-bit LPDDR5 memory and a bandwidth of 102.4 GB/s, it ensures rapid data access and processing, essential for handling high-throughput sensor data and maintaining system responsiveness. The module's power consumption is configurable between 10W and 25W, allowing for optimization based on performance needs and energy availability, which is vital for battery-powered applications [29].

Trade-offs Associated with Each Approach

While the Jetson Orin NX offers significant advantages, certain trade-offs were considered: the advanced features and high performance come at a higher cost compared to less powerful alternatives; although power-efficient, the module's configurable power consumption between 10W and 25W necessitates careful power management, especially in battery-operated systems; utilizing the full capabilities of the Jetson Orin NX may require a steeper learning curve and more complex software development compared to simpler microcontrollers; and the high processing power can generate significant heat, necessitating effective thermal management solutions, which adds to the design complexity and may require additional components, impacting the overall system design.

Justification for Selecting the Jetson Orin NX

The choice of the Jetson Orin NX is justified by its superior AI processing capabilities, compact form factor, and energy efficiency. Its ability to handle complex computations and multiple sensor inputs in real-time makes it particularly well-suited for autonomous applications requiring high performance within constrained power budgets. The module supports several interfaces, including multiple USB 3.2 Gen2 and USB 2.0 ports, facilitating high-speed data transfer and peripheral connectivity. It also offers display outputs such as DisplayPort 1.4a and HDMI 2.1, enabling integration with visual output devices. Additional I/O options include UART, SPI, I2S, I2C, and CAN interfaces, providing flexibility for connecting a wide range of sensors and actuators [29].

5.5.2 Sensing and Perception

In autonomous systems, effective sensing and perception are crucial for understanding and interacting with the environment. Our system integrates multiple sensors, each selected for its unique capabilities and contributions to overall functionality.

RPLIDAR A3M1

Theory Behind Technological Approaches: LiDAR (Light Detection and Ranging) systems emit laser pulses and measure the time it takes for the reflections to return, calculating distances to objects. By rotating the laser emitter, LiDAR provides 360-degree environmental mapping, creating detailed 2D representations. The RPLIDAR A3M1 utilizes a high-speed laser ranging core, achieving a ranging radius up to 25 meters and a sampling rate of 16,000 times per second, enabling precise mapping and localization [30].

Trade-offs Associated with Each Approach: While LiDAR offers high-resolution mapping and accurate distance measurements, it comes with higher costs compared to other sensors. Additionally, LiDAR performance can be affected by environmental factors such as rain, fog, or dust, which may scatter the laser beams and reduce accuracy [31].

Justification for Selecting the RPLIDAR A3M1: The RPLIDAR A3M1 was chosen for its balance between performance and cost. Its extended range and high sampling rate provide detailed environmental data essential for navigation and obstacle avoidance. Moreover, its ability to operate in both indoor and outdoor environments enhances system versatility.

Stereo Camera: Intel® RealSense™ D435i

Theory Behind Technological Approaches: Stereo cameras capture images from two slightly different perspectives, mimicking human binocular vision. By analyzing disparities between these images, the system computes depth information, facilitating object detection and environmental understanding. The Intel® RealSense™ D435i combines stereo vision with an inertial measurement unit (IMU), enhancing depth perception and motion tracking [32].

Trade-offs Associated with Each Approach: Integrating stereo cameras can be complex, requiring calibration and synchronization. They may also struggle in low-light conditions or with texture less surfaces, leading to less accurate depth measurements. However, they provide rich color information alongside depth data, beneficial for object recognition tasks.

Justification for Selecting the Intel® RealSense™ D435i: The D435i offers a compact solution with integrated depth sensing and motion tracking, reducing the need for multiple separate sensors. Its wide field of view and high frame rate support dynamic environments, and the inclusion of an IMU aids in stabilizing depth data during movement.

Ultrasonic Sensors

Theory Behind Technological Approaches: Ultrasonic sensors emit high-frequency sound waves and measure the time it takes for the echoes to return after reflecting off objects. This time-of-flight measurement allows for calculating distances to nearby obstacles, making them effective for close-range detection [33].

Trade-offs Associated with Each Approach: Ultrasonic sensors are cost-effective and simple to implement but have limited range and resolution. They may also have difficulty detecting soft or angled surfaces that poorly reflect sound waves. Environmental noise at ultrasonic frequencies can interfere with their operation [34].

Justification for Selecting Ultrasonic Sensors: Despite their limitations, ultrasonic sensors provide reliable short-range obstacle detection, serving as a supplementary safety measure. Their low cost and straightforward integration make them a practical addition to the sensor suite, enhancing redundancy and robustness in obstacle detection.

GPS

Theory Behind Technological Approaches: The Global Positioning System (GPS) utilizes signals from satellites to determine the precise location of a receiver on Earth. By calculating the time delay of signals from multiple satellites, the system triangulates the receiver's position, providing latitude, longitude, and altitude data [35].

Trade-offs Associated with Each Approach: While GPS offers global coverage and accurate positioning outdoors, its performance degrades in environments with obstructions like tall buildings, dense foliage, or indoors, where satellite signals are weak or unavailable. Additionally, standard GPS lacks the precision required for fine-grained navigation tasks without augmentation systems [36].

Justification for Selecting GPS: Incorporating GPS provides essential global positioning data, aiding in large-scale navigation and route planning. When combined with other sensors, GPS enhances overall system accuracy and reliability, especially in open outdoor environments.

5.5.3 Microcontrollers

In our system architecture, microcontrollers play a pivotal role in managing sensor inputs and interfacing with the central processing unit. We have selected Arduino microcontrollers for their versatility and ease of integration.

Theory Behind Technological Approaches

Microcontrollers are compact computing devices designed to execute specific control tasks within embedded systems. They process input signals from several sensors and generate appropriate output signals to actuators or higher-level processing units. Arduino microcontrollers, such as the Arduino Uno, are based on the ATmega328P microcontroller, featuring 14 digital input/output pins and 6 analog inputs. They operate at a clock speed of 16 MHz and are programmable via the Arduino Integrated Development Environment (IDE), which simplifies code development and deployment [37].

Trade-offs Associated with Each Approach

While Arduino microcontrollers offer user-friendly programming and a vast ecosystem of libraries and community support, they come with certain limitations. Their processing power and memory capacity are modest compared to more advanced microcontrollers or microprocessors, which may restrict their ability to handle complex computations or manage multiple high-speed peripherals simultaneously. Additionally, the abstraction layers provided by the Arduino platform, while beneficial for rapid development, can introduce overheads that affect performance [38].

Justification for Selecting Arduino Microcontrollers

Despite these trade-offs, Arduino microcontrollers are well-suited for our application due to their ease of use, extensive community resources, and sufficient capabilities for managing sensor inputs and interfacing tasks. Their open-source nature allows for customization and scalability, facilitating integration with several sensors and communication protocols. Moreover, the availability of numerous shields and modules compatible with Arduino boards enhances their adaptability to different project requirements [39].

5.5.4 Power Management

Effective power management is crucial for the optimal performance and longevity of autonomous systems. Selecting an appropriate battery system involves balancing capacity, weight, and cost to meet the system's energy demands efficiently.

Theory Behind Technological Approaches

Li-ion batteries operate on the principle of lithium-ion movement between the anode and cathode through an electrolyte. During discharge, lithium ions move from the anode to the cathode, releasing energy. During charging, an external power source drives the ions back to the anode, storing energy for future use. This reversible process allows for multiple charge and discharge cycles, contributing to the battery's longevity and efficiency [40].

Trade-offs Associated with Each Approach

When selecting a battery system, several trade-offs must be considered. Higher capacity batteries provide longer operational times but often come with increased weight, which can impact the system's mobility and efficiency. Advanced battery technologies like Li-ion offer superior performance but at a higher cost, necessitating a compromise between budget constraints and desired battery performance. Additionally, batteries with higher energy densities can pose safety risks, such as thermal runaway, if not managed properly, making the implementation of robust safety mechanisms essential to mitigate these risks [41].

Justification for Selecting Lithium-Ion Batteries

After evaluating the available options, lithium-ion batteries have been selected for the following reasons. Li-ion batteries provide a high energy-to-weight ratio, enabling longer operational periods without significantly increasing the system's weight. They retain their charge over extended periods, reducing the need for frequent recharging and ensuring readiness for use. Their reduced weight and size contribute to the system's overall efficiency and portability. While the initial cost of lithium-ion batteries is higher compared to some alternatives, their performance benefits and suitability for weight-sensitive applications justify the investment. Additionally, implementing appropriate safety measures will address potential risks associated with their use [41].

5.5.5 Motors

In autonomous systems, the selection of motors is pivotal, as it directly influences the vehicle's performance, efficiency, and adaptability to several tasks. Understanding the relationship between torque and speed is essential for optimizing motor selection and achieving desired operational outcomes.

Theory Behind Technological Approaches

Electric motors operate based on the interaction between magnetic fields and current-carrying conductors, producing rotational motion. The fundamental relationship governing motor performance is expressed as: Power (P) = Torque (T) × Speed (ω). This equation indicates that, for a given power output, an increase in speed results in a decrease in torque, and vice versa. Different motor types exhibit varying torque-speed characteristics:

- Brushed DC Motors: These motors provide a linear relationship between voltage and speed, and between current and torque. They are simple to control but may require maintenance due to brush wear.

Trade-offs Associated with Each Approach

- Torque vs. Speed: High-speed motors typically deliver lower torque, while high-torque motors operate at lower speeds. Achieving both high torque and high speed often requires complex gearing systems or advanced motor designs.
- Efficiency vs. Control Complexity: Motors like BLDC offer high efficiency but require sophisticated control systems. Conversely, brushed DC motors are easier to control but may have lower efficiency and require more maintenance.

- Cost vs. Performance: High-performance motors with advanced features come at a higher cost. Budget constraints may necessitate compromises between performance requirements and financial considerations.
- Size and Weight: High-torque motors are generally larger and heavier, which can impact the design and mobility of the system. Space constraints and weight limitations must be considered in the selection process.

Justification for Selecting Specific Motors

In our autonomous system, the choice of motor is guided by the need for a balance between torque and speed to ensure efficient navigation and task execution. Brushless DC motors (BLDC) have been selected due to their high efficiency, reliability, and favorable torque-speed characteristics. BLDC motors provide consistent torque across a wide speed range, facilitating smooth acceleration and precise control. Their maintenance-free operation and longer lifespan align with the system's requirements for durability and reduced downtime. While BLDC motors necessitate more complex control systems, the benefits in performance and efficiency justify the investment in advanced control strategies. Additionally, the compact size and lower weight of BLDC motors contribute to the overall agility and energy efficiency of the autonomous system.

5.6 Software design

Software Development and Frameworks in Robotics: ROS 2, Python, and C++

In modern robotic systems, software frameworks and programming languages play a critical role in enabling efficient development, hardware integration, and seamless communication between several system components. Frameworks like ROS 2 provide middleware for communication, while languages like Python and C++ offer different strengths depending on the nature of the tasks at hand. We will analyze the role of each of these tools and their tradeoffs, along with justifications for selecting them in different parts of a robotic system.

1. ROS 2: Middleware for Modular Robotics Development

Overview:
ROS 2 (Robot Operating System 2) is a flexible middleware framework that enables modular robot development [42]. It provides a set of tools, libraries, and conventions to help developers build complex robotic systems. Unlike its predecessor ROS 1, ROS 2 is designed to support real-time communication, better scalability, and cross-component integration [43].

Key Features:

1. Modular Structure: ROS 2 promotes modular development, allowing components of a robot (such as sensors, actuators, or algorithms) to interact through well-defined interfaces. For example, autonomous vehicles camera system can publish data on a topic, which is then consumed by an object detection node.
2. Real-Time Communication: ROS 2 uses the DDS (Data Distribution Service) protocol for efficient communication, which is essential for systems requiring real-time feedback, such as robotic control loops or sensor fusion.
3. Cross-Component Integration: ROS 2 excels at integrating different hardware and algorithms. For example, integrating a LAIDR, IMU, and motor controllers in a single robot platform is straightforward due to ROS 2's standardized interfaces and messaging system.

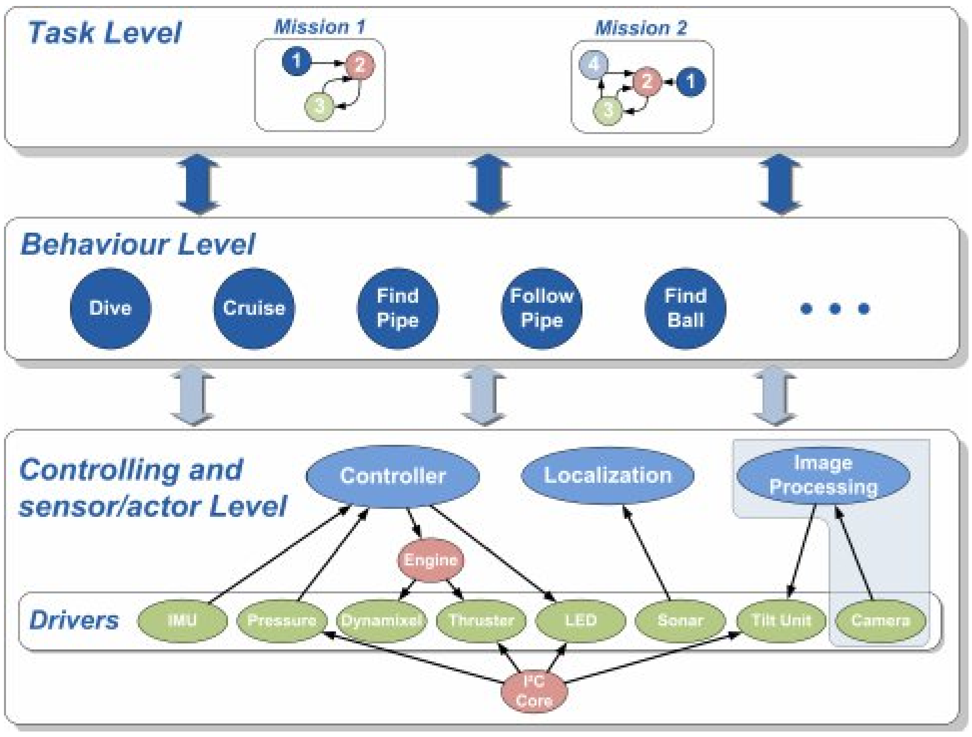


Figure 4 Hierarchical Architecture of a Robotic System: Task, Behavior, and Control Levels

Tradeoffs:

- Learning Curve: ROS 2 can be overwhelming for beginners, as it introduces concepts like nodes, topics, and services, which might be new to developers unfamiliar with middleware architectures. The complexity increases with the need for understanding network communication, message passing, and the design of distributed systems.
- Overhead for Simple Systems: with a few simple tasks, ROS 2 might be overkill. The setup and configuration can become unnecessarily complex if the robot doesn't need the full suite of features ROS 2 provides, such as real-time communication or high scalability.

Justification for Selection: ROS 2 is ideal for more complex robotic systems where modularity, real-time communication, and the ability to integrate different sensors, actuators, and algorithms are crucial. It is well-suited for large systems or multi-robot coordination, where communication between distributed components must be reliable and efficient.

2. Python: High-Level Programming for Rapid Development

Overview:
Python is a high-level, interpreted programming language known for its simplicity and readability [44]. It is commonly used in robotics for higher-level logic, prototyping, and system orchestration. Python’s ecosystem includes powerful libraries for data analysis (NumPy, Pandas), machine learning (TensorFlow, PyTorch), and computer vision (OpenCV), making it an excellent choice for integrating complex algorithms and data processing tasks into an autonomous system [45].

Key Features:

1. Rapid Prototyping: Python allows developers to quickly test ideas, write scripts, and prototype algorithms. It is especially useful in the early stages of development, when speed of iteration is important.
2. Integration with ROS 2: Python has excellent support for ROS 2, with libraries like rclpy (ROS Client Library for Python) enabling easy communication with ROS 2 nodes. For instance, Python can be used to write a high-level controller that handles decisions like path planning or obstacle avoidance, based on data received from several sensor nodes.
3. Extensive Libraries: Python’s ecosystem includes powerful libraries for data analysis (NumPy, Pandas), machine learning (TensorFlow, PyTorch), and computer vision (OpenCV), making it an excellent choice for integrating complex algorithms and data processing tasks into an autonomous system.

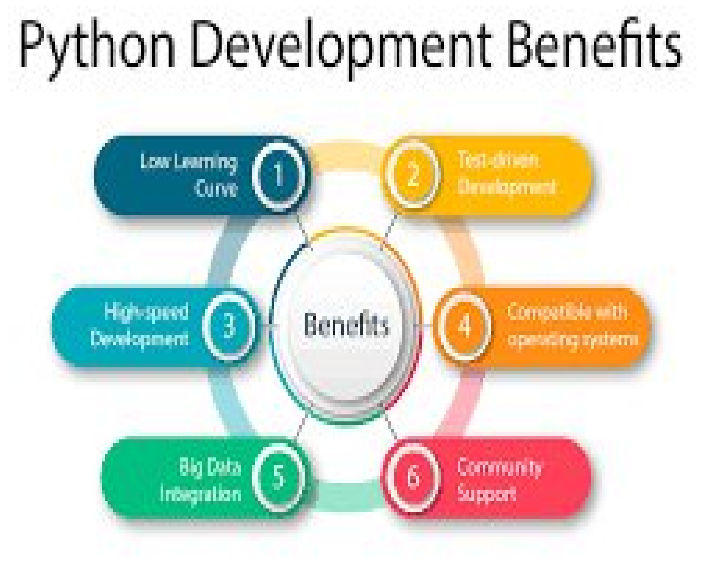


Figure 5 Python Development Benefits

Tradeoffs:

- Slower Performance: While Python is great for developing algorithms quickly, its performance is slower than compiled languages like C++. This makes it less suitable for tasks that require real-time processing, such as low-level motor control or handling large streams of sensor data.
- Not Ideal for Time-Critical Tasks: In robotics, real-time tasks like controlling a motor or processing data from high-frequency sensors need to be handled efficiently. Python's performance limitations can hinder real-time responsiveness, especially when precision and low latency are crucial.

Justification for Selection: Python is perfect for high-level logic, algorithm development, and tasks like sensor fusion, data analysis, and machine learning. It is often used in conjunction with ROS 2 to handle tasks that don't demand real-time performance but benefit from Python's rapid development capabilities.

3. C++: High-Performance Control and Hardware Interfacing

Overview:
C++ is a lower-level, compiled programming language known for its high performance and fine-grained control over system resources [46]. It is extensively used in robotics for tasks that require real-time control, such as motor actuation, sensor data processing, and efficient hardware interfacing [47].

Key Features:

1. Real-Time Control: C++ provides the speed and precision necessary for real-time control tasks. For example, controlling motors with precise timing or processing sensor data at high frequencies (e.g., lidar data) often requires the performance capabilities of C++.

2. Memory Management: Unlike higher-level languages like Python, C++ gives developers control over memory allocation and management, which is crucial in resource-constrained environments, such as embedded systems.
3. Efficient Hardware Interfacing: C++ allows direct access to hardware interfaces and is commonly used to write drivers for sensors, actuators, and other low-level hardware components. It is often used in ROS 2 to develop nodes that handle real-time data processing.

Tradeoffs:

- Complexity: C++ is more complex than Python, requiring a deeper understanding of topics like memory management, pointers, and the intricacies of multi-threading. This complexity can make development slower and more error-prone.
- Longer Development Time: Writing and debugging C++ code tends to take longer than Python due to its verbosity and the need for manual memory management. It can also be more challenging for rapid prototyping compared to Python's simplicity.

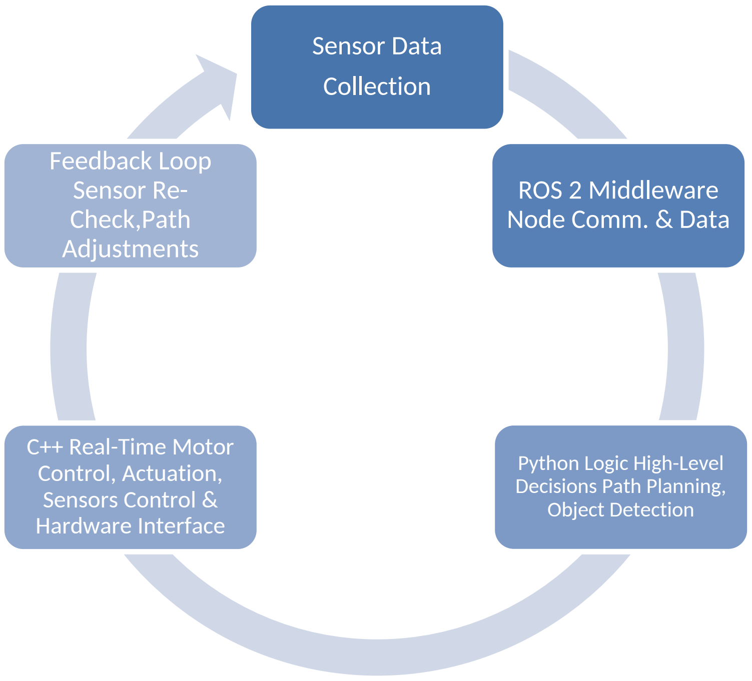
Justification for Selection: C++ is essential for low-level, real-time control tasks that demand high performance and memory efficiency. It is ideal for situations where precise control over hardware and fast processing of large amounts of data (such as sensor inputs) are critical. In robotics, C++ is often used for tasks like motor control, real-time sensor fusion, and processing data from high-frequency sensors.

Table 14 Summary of Tradeoffs and Justifications

Aspect	ROS 2	Python	C++
Purpose	Middleware for modular integration and communication	High-level control, rapid prototyping, system orchestration	Real-time control, low-level hardware interfacing
Learning Curve	Moderate to hard	Easy to moderate	Hard
Performance	Real-time capabilities (via DDS)	Slower, not suitable for real-time tasks	High-performance, real-time control
Development Speed	Moderate (requires infrastructure setup)	Fast prototyping, easy to develop and iterate	Slower, more complex development
Use Cases	Complex robot systems with multiple components	Sensor fusion, data analysis, high-level decision making	Motor control, sensor data processing, real-time systems

Each of these ROS 2, Python, and C++—plays a vital role in the development of a robotic system. ROS 2 provides a powerful middleware framework for modular development and real-time communication between components, making it perfect for complex robot systems. Python is ideal for higher-level tasks, rapid prototyping, and algorithm development, particularly when working with data analysis or machine learning. C++, on the other hand, excels in performance-critical, real-time control tasks, such as motor actuation and sensor data processing.

By combining these tools, developers can leverage the strengths of each to build efficient, scalable, and highly functional robotic systems, with Python handling the high-level orchestration, C++ managing low-level performance, and ROS 2 facilitating seamless communication across all components.



Sensor Data Collection
ROS 2 Middleware Node Comm. & Data
Python Logic High-Level Decisions Path Planning, Object Detection
C++ Real-Time Motor Control, Actuation, Sensors Control & Hardware Interface
Feedback Loop Sensor Re-Check,Path Adjustments

Figure 6 Software Life cycle

Integration Explanation

In the autonomous golf cart project, ROS 2, Python, and C++ work in synergy to ensure seamless operation across several components. ROS 2 acts as the middleware framework, enabling modular communication between nodes, whether written in Python or C++. For example, a Python node responsible for high-level path planning. This design allows each component to specialize in specific tasks while maintaining real-time communication using DDS (Data Distribution Service), the core protocol in ROS 2.

Python nodes directly interact with ROS topics, handling higher-level tasks like decision-making and sensor fusion. They subscribe to stereo camera and LiDAR data, process the information, and publish navigation commands for motor controllers. On the other hand, C++ nodes handle real-time, low-level tasks such as processing LiDAR data for obstacle detection or directly controlling motor actuators. These C++ nodes ensure the system meets real-time constraints by providing fast and efficient execution. Together, the ROS 2 middleware facilitates reliable communication between Python and C++ nodes, allowing the system to function cohesively.

Examples Specific to the Project

In this project, Python and C++ have distinct roles to optimize performance and development speed. Python is used for tasks that require high-level logic and flexibility, such as path planning, where algorithms are implemented. It also handles sensor fusion by combining data from the LiDAR, GPS, and stereo camera to create a comprehensive environmental model. Python nodes classify obstacles based on stereo camera data, determining whether to avoid or navigate through them.

C++, on the other hand, is utilized for performance-critical tasks. A C++ node translates navigation commands from Python into precise motor control signals, allowing the cart to adjust its speed and direction dynamically. ROS 2 nodes and topics tie these tasks together, with specific nodes publishing and subscribing to relevant data. For instance, the stereo camera node publishes depth and RGB data, the LiDAR node publishes obstacle distances, and the path planning node publishes navigation commands, which are consumed by the motor control node.

Error Handling

The system incorporates robust error-handling mechanisms to ensure continuous operation. If the LiDAR fails to provide data, a fallback mechanism coded in C++ switches to using ultrasonic sensors for basic obstacle detection. This fallback system ensures that navigation continues safely. In case a Python node crashes, ROS 2's lifecycle management automatically restarts the node, minimizing downtime. Furthermore, redundant nodes can take over critical functions in the event of failures.

To address communication failures, ROS 2's DDS protocol ensures reliable message delivery with retries for lost messages. Any errors in node communication or computation are logged using ROS 2's logging framework, enabling the development team to quickly diagnose and fix issues.

Workflow of Tasks

The system's workflow is divided among Python, C++, and ROS 2 middleware to optimize efficiency. Python nodes handle high-level tasks such as decision-making, path planning, and sensor fusion. These nodes process data from sensors like LiDAR and stereo cameras and publish navigation commands to ROS topics. C++ nodes focus on low-level tasks like processing real-time LiDAR data and controlling motors with precise actuation commands. The ROS 2 middleware facilitates seamless communication between these components, ensuring that each node operates in sync with the overall system.

5.7 New-gained knowledge

Throughout the design and implementation of the autonomous golf cart project, the team acquired several skills and insights that are instrumental in solving real-world engineering challenges. These skills span across technical, analytical, and collaborative domains, contributing to the team's overall growth and preparedness for future projects.

Advanced Software Tools and Development Frameworks

One of the most notable areas of learning was the extensive use of ROS2 for sensor communication and real-time data processing. Exploring ROS2 enabled the team to develop a robust communication architecture, synchronizing inputs from LiDAR, stereo cameras, ultrasonic sensors, and GPS modules. Additionally, while Lucidchart proved invaluable for visually conceptualizing the system's hardware and software architectures. The team also gained proficiency in using Chart.io to present complex data insights interactively, enhancing stakeholder understanding of performance metrics and system evaluation results.

Integration of Advanced Hardware Components

The team developed expertise in integrating and optimizing advanced hardware components such as the Jetson Orin NX, RPLIDAR A3M1, Intel® RealSense™ D435i camera, and ultrasonic sensors. Understanding the nuances of these components, including their interfaces, trade-offs, and operational constraints, enabled the team to design a cohesive system that balances performance and cost. Specific challenges, such as thermal management of high-performance processors and synchronizing multiple sensors, provided a deeper understanding of embedded systems and hardware design principles.

Problem-Solving and Decision-Making

Through iterative design and testing, the team addressed key challenges like sensor fusion, obstacle detection, and path planning in real-time. They learned how to evaluate design alternatives based on trade-offs, such as cost versus accuracy, and adopted a data-driven approach to justify decisions. For instance, selecting lithium-ion batteries for power management required analyzing energy density, weight, and cost to meet operational requirements.

Multidisciplinary Collaboration

This project demanded collaboration across several disciplines, including mechanical engineering for motor and vehicle dynamics, computer science for AI algorithms, and electrical engineering for sensor integration and power management. By working together, team members gained insights into disciplines beyond their own areas of expertise, fostering a holistic understanding of engineering solutions.

Real-World Application of Research and Standards

The project emphasized the importance of aligning design decisions with global standards and practical constraints. Standards such as ISO 26262 for functional safety and ISO 212 for intelligent transport systems guided the development of a safe and reliable system. Moreover, integrating sustainability goals aligned with Qatar’s vision for technological innovation further underscored the impact of engineering solutions in a societal and environmental context.

Innovative Use of Visualization and Planning Tools

Lucidchart and Chart.io were extensively used to conceptualize system interconnections and showcase data trends during testing phases. These tools enhanced the team's ability to communicate complex ideas to diverse audiences, including non-technical stakeholders, making it easier to demonstrate the project's feasibility and performance.

In conclusion, the development of the autonomous golf cart not only resulted in a functional and innovative solution but also equipped the team with valuable technical knowledge and problem-solving abilities. These skills will serve as a foundation for tackling more complex engineering problems in future efforts.

6 Implementation

6.1 System Design Overview

The implementation of the autonomous vehicle system follows a modular design, structured to ensure clear subsystem separation and efficient integration. At the center of the architecture is the Jetson Orin NX, acting as the On-Board Computer (OBC). This unit handles the primary computational tasks, including perception, sensor fusion, navigation, and real-time motor control.

The Jetson Orin NX is directly connected via USB interfaces to both the RPLIDAR A3M1 and the Intel® RealSense™ D435i depth camera. These sensors are essential for environmental perception: the LiDAR performs 360-degree long-range mapping, while the RealSense camera provides stereo vision and depth estimation. Their direct USB connection ensures high-bandwidth and low-latency data communication for real-time processing.

Three Arduino Uno microcontrollers interface with the Jetson via USB. Each Arduino is assigned to a specific role: one manages the motor drivers responsible for controlling steering and propulsion; the second processes data from three front-facing ultrasonic sensors; and the third handles three rear-facing ultrasonic sensors. The ultrasonic sensors are arranged to provide a coverage range of 1.55 meters forward/backward and 1.76 meters forward left/right backward left/right, enhancing close-range obstacle detection on both ends of the vehicle.

The power system is driven by a 24V battery, housed inside the chassis, and controlled via a physical on/off switch. A buck converter steps down the voltage to supply 5V regulated power to the Jetson, while the Arduinos are powered through the Jetson itself. The motor drivers receive power directly from the 24V battery but are controlled through the Arduinos, which receive high-level commands from the Jetson. Both the LiDAR and camera are powered via USB directly from the Jetson, ensuring a compact and streamlined power setup.

Currently, the system remains modular to allow for flexible testing and debugging, though component mounting on the final chassis is in progress. The motors operate in open-loop mode, receiving movement and steering commands without encoder feedback. This simplified design supports early-stage prototyping and will be refined based on further testing outcomes.

The figure below illustrates the actual wiring and interconnection of the implemented modules:

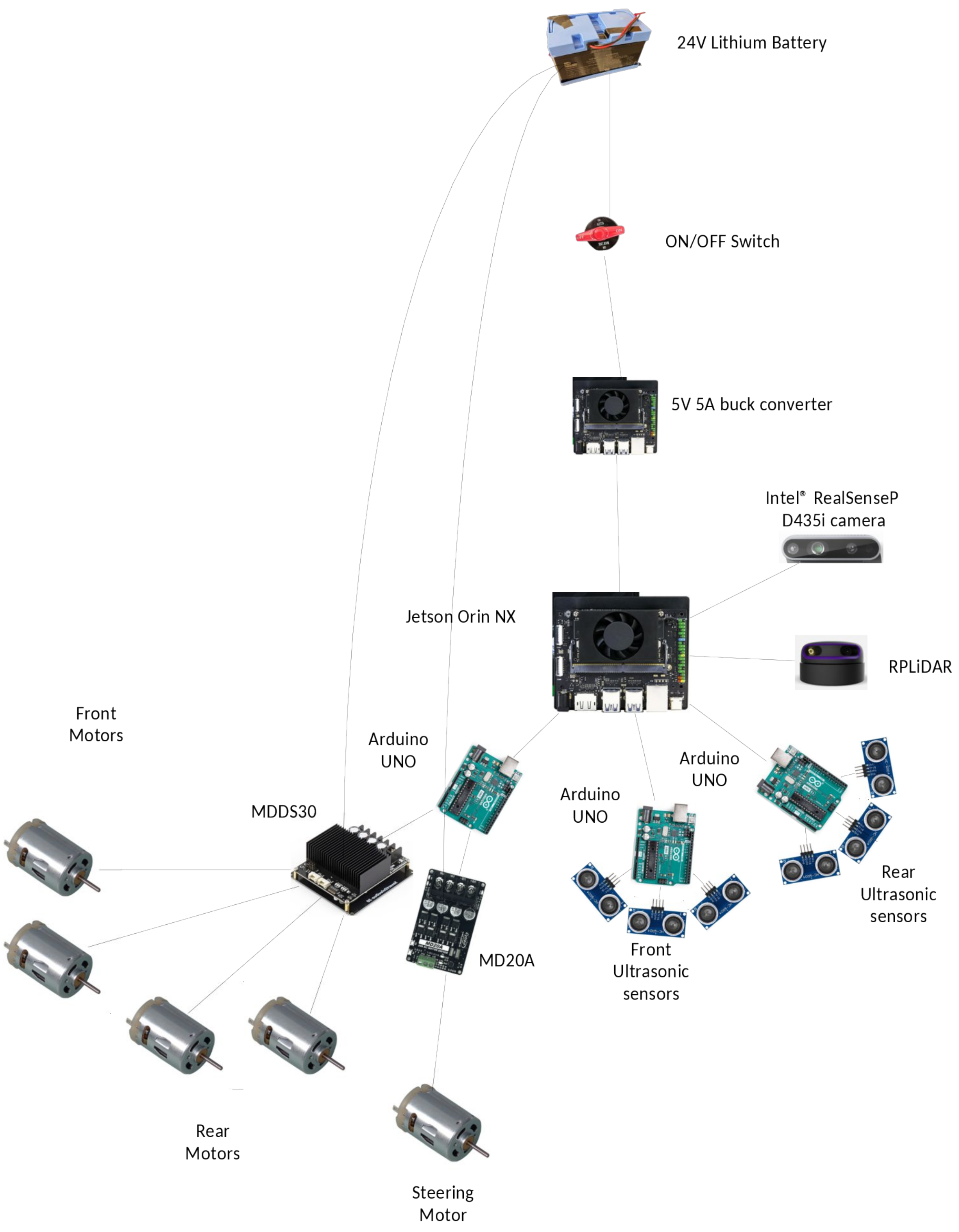


Figure 7 System connectivity Implementation

This implementation strategy provides a well-organized hardware layout, simplifies debugging and maintenance, and supports the modular development of individual subsystems while maintaining a fully integrated autonomous solution.

6.2 Hardware Modules and Connections

The implementation of the autonomous golf cart system began by assembling the core hardware components around the Jetson Orin NX development kit, which serves as the main processing unit. The Jetson is enclosed in its official protective case and includes a built-in cooling fan to ensure thermal stability during extended operation.

For environmental sensing, the system integrates two key components: the RPLIDAR A3M1, used for 360-degree long-range obstacle detection, and the Intel® RealSense™ D435i stereo depth camera, responsible for front-facing object recognition and depth estimation. Both sensors are connected directly to the Jetson Orin NX via USB and are mounted on custom-designed brackets to provide stable and accurate field-of-view coverage.

To manage lower-level hardware interactions, the system uses three Arduino Uno boards. One Arduino controls the motor drivers, including the MDDS30, which drives four brushed DC motors — two located at the front and two at the rear wheels — responsible for forward and backward motion. Another motor, dedicated to steering, is controlled by a separate MD20A driver. These motor drivers receive PWM control signals from the Arduino to execute navigation commands issued by the Jetson.

The remaining two Arduinos are connected to a total of six ultrasonic sensors three positioned at the front and three at the rear of the vehicle. These sensors provide close-range obstacle detection with a directional coverage of approximately 1.55 meters directly forward or backward (north/south) and up to 1.76 meters diagonally to the front-left/front-right and rear-left/rear-right (northeast/northwest and southeast/southwest). This spatial arrangement enhances the vehicle's awareness in tight environments and during maneuvers. All Arduino boards remain exposed for ease of access and modification during testing.

The system is powered by a 24V lithium battery installed inside the chassis. A 5V 5A buck converter is used to step down voltage to power the Jetson Orin NX, while the Arduinos receive power through USB connections from the Jetson itself. The motor drivers are powered directly from the battery, receiving logic signals from the Arduino. A manual power switch is used to turn the entire system on or off. Wiring across the setup uses a combination of jumper wires and XT60 connectors, providing both flexibility and stability during development. Final wire management and component mounting are planned once the testing phase is complete.

The following images illustrate key aspects of the hardware implementation:

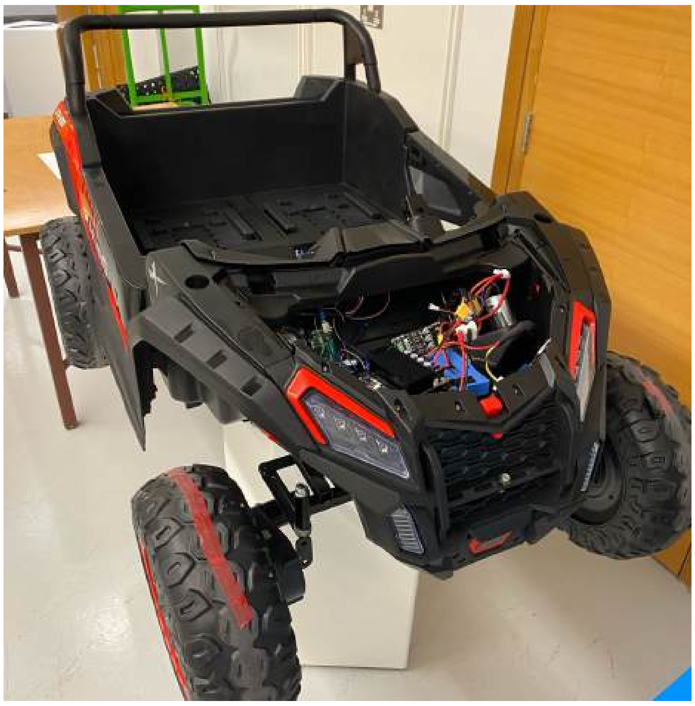


Figure 9 Overall Chassis with initial Component equipped.

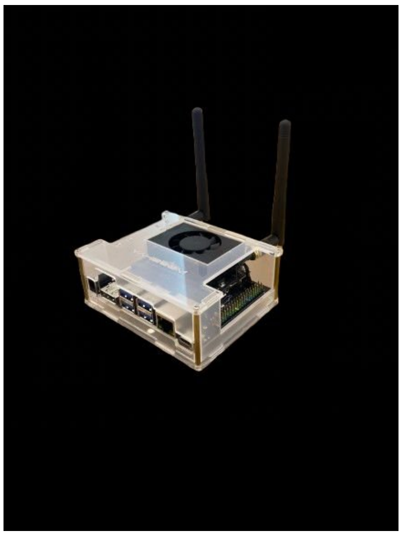


Figure 8 Jetson Orin NX Setup with Cooling

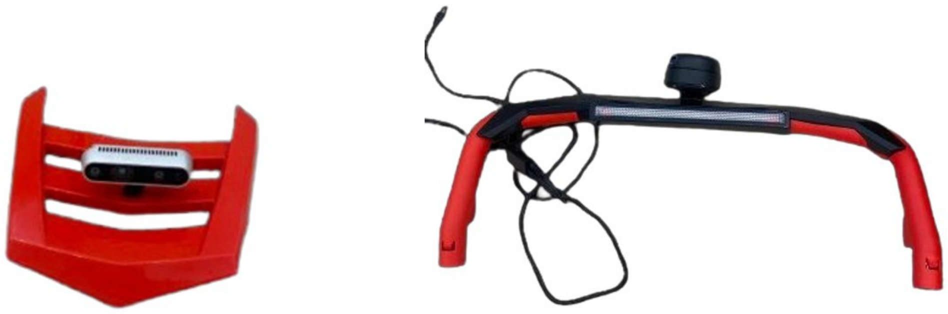


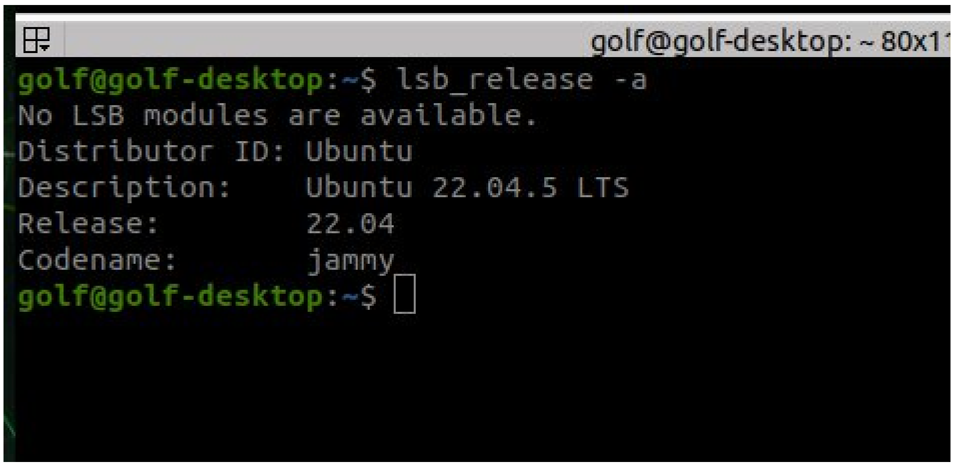
Figure 10 Mounted LiDAR and RealSense Camera

This hardware configuration forms a solid foundation for autonomous operation, ensuring reliable perception, control, and actuation in controlled environments.

6.3 Software Stack and Tools Used

The software stack for the autonomous golf cart system was carefully selected to support modular development, real-time processing, and seamless integration of sensors, actuators, and control logic. The system runs on the Jetson Orin NX Development Kit, a powerful AI computing module offering up to 100 TOPS (Tera Operations Per Second), ideal for autonomous applications requiring parallel execution of perception and control algorithms.

The Jetson was set up with Ubuntu 22.04 LTS, chosen for its compatibility with ROS 2 Humble, NVIDIA Jetson SDKs, and its strong community support. The team used the default Ubuntu desktop environment with no additional customizations, ensuring a stable and well-supported base.



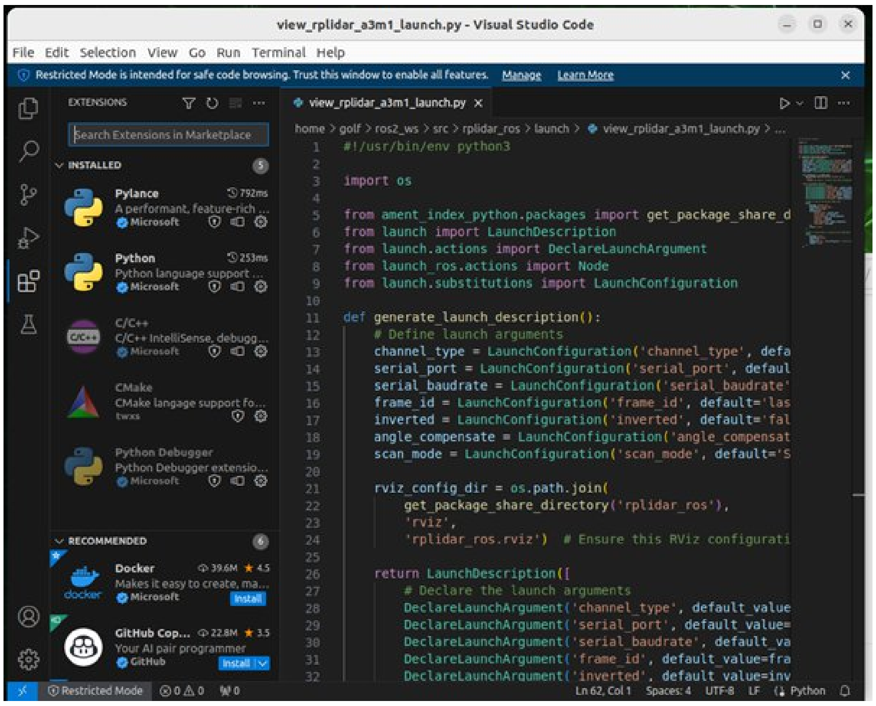
```
golf@golf-desktop: ~ 80x1
golf@golf-desktop:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.5 LTS
Release:        22.04
Codename:       jammy
golf@golf-desktop:~$ 
```

Figure 11 Screenshots of the Jetson Orin running Ubuntu 22.04

At the core of the system lies ROS 2 Humble, an open-source middleware framework designed for modular robotic development. ROS 2 enabled inter-process communication through its publish/subscribe model using DDS (Data Distribution Service), allowing each software node (e.g., sensor input, processing, motor control) to operate independently yet collaboratively. Packages were organized by function within a single workspace (ros2_ws), and colcon was used as the primary build system.

To support autonomous navigation and environment mapping, the team integrated the Navigation2 (Nav2) stack, a ROS 2-based navigation framework designed for mobile robots. Nav2 will be used for pre-mapping the environment, enabling the system to generate an occupancy grid map before running autonomous path planning. The stack provides essential navigation capabilities such as localization, SLAM, path planning, and controller integration. While the full navigation pipeline is still under development, the current implementation focuses on the mapping phase, with data gathered from sensors being used to generate usable maps. This integration will form the basis for autonomous waypoint-based navigation in future phases.

For software development, the team used Visual Studio Code (VS Code) as the main IDE, enhanced with extensions for Python, C++, CMake, and ROS 2 support. Multi-terminal operations, such as launching multiple ROS nodes concurrently, were streamlined using Terminator, a lightweight and customizable terminal manager.


view_rplidar_a3m1_launch.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
EXTENSIONS
Search Extensions in Marketplace
INSTALLED
Pylance
A performant, feature-rich ...
Microsoft
Python
Python language support ...
Microsoft
C/C++
C/C++ IntelliSense, debugg...
Microsoft
CMake
CMake langage support fo...
twxs
Python Debugger
Python Debugger extensio...
Microsoft
RECOMMENDED
Docker
Makes it easy to create, ma...
Microsoft
Install
GitHub Cop...
Your AI pair programmer
GitHub
Install
home > golf > ros2_ws > src > rplidar_ros > launch > view_rplidar_a3m1_launch.py > ...
Restricted Mode
Ln 62, Col 1 Spaces: 4 UTF-8 LF Python

```
#!/usr/bin/env python3

import os

from ament_index_python.packages import get_package_share_d
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch_ros.actions import Node
from launch.substitutions import LaunchConfiguration

def generate_launch_description():
    # Define launch arguments
    channel_type = LaunchConfiguration('channel_type', defa
    serial_port = LaunchConfiguration('serial_port', defaul
    serial_baudrate = LaunchConfiguration('serial_baudrate'
    frame_id = LaunchConfiguration('frame_id', default='las
    inverted = LaunchConfiguration('inverted', default='fal
    angle_compensate = LaunchConfiguration('angle_compensat
    scan_mode = LaunchConfiguration('scan_mode', default='S

    rviz_config_dir = os.path.join(
        get_package_share_directory('rplidar_ros'),
        'rviz',
        'rplidar_ros.rviz')  # Ensure this RViz configurati

    return LaunchDescription([
        # Declare the launch arguments
        DeclareLaunchArgument('channel_type', default_value
        DeclareLaunchArgument('serial_port', default_value=
        DeclareLaunchArgument('serial_baudrate', default_va
        DeclareLaunchArgument('frame_id', default_value=fra
        DeclareLaunchArgument('inverted', default_value=inv
```

Figure 12 VS Code environment with installed extensions.

All system logic was developed using Python 3, with pip used to manage package installations. No C++ nodes were implemented at this stage. Development included common scientific and machine learning libraries such as OpenCV, NumPy, PyTorch, and pandas, supporting object detection, image processing, and data handling.

Sensor integration was achieved using the pyrealsense2 SDK to interface with the Intel® RealSense™ D435i depth camera, enabling simultaneous access to RGB, depth, and IMU streams. The rplidar_ros package was integrated to support the RPLIDAR A3M1, providing reliable 360° obstacle detection. Visualization of sensor outputs, such as LiDAR point clouds and camera frames, was performed in RViz2, allowing the team to validate sensor alignment and frame transforms.

While advanced tools like rqt_graph, ros2 bag, or simulation in Gazebo were not used during the current implementation, they are planned for future development. No automation scripts or custom launch files were implemented, though environment variables were configured by sourcing setup.bash in .bashrc.

In summary, this software stack reflects the project's emphasis on modularity, real-time data handling, and AI readiness, forming a scalable platform for future upgrades including simulation, autonomy layers, and additional sensors.

Note:

Although object detection using YOLOv5 was planned and partially configured, full integration was not completed due to hardware limitations. The algorithm is reserved for future work once a compatible depth camera is available.

6.4 Implementation Steps

The development of the autonomous golf cart system followed a practical, hands-on approach with step-by-step integration of individual components. The team prioritized hardware testing before software integration to ensure system stability and reduce debugging complexity during full system assembly. The following steps outline the progression of implementation.

Step 1: Motor Driver Setup and Manual Testing

The implementation process began by verifying the operation of the five DC motors through manual testing. The motor drivers (MDDS30 for the four-wheel motors (two front and two rear) and MD20A for the steering motor) were tested individually using the built-in manual control buttons provided by the manufacturer (shown in figure 13) Power was supplied through the recommended wiring harnesses, and each motor was tested independently to confirm proper mechanical operation. The rear wheels were tested through one channel of the MDDS30, while the front wheels used a second channel. The steering motor, connected to the MD20A, was also tested independently to confirm rotational control. This step ensured that all motors responded correctly to manual inputs without any issues related to direction, wiring, or overheating, and established a hardware baseline for future programming and control.

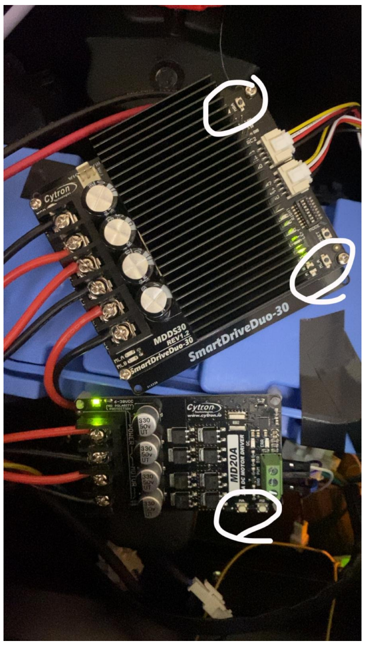


Figure 13 manual control buttons provided by the manufacturer in the cytron drivers

Step 2: Arduino Programming and Motor Control Integration

After completing manual testing, the team transitioned to automated control using Arduino Uno boards. For the MD20A steering driver, the integration was straightforward, involving direct signal connections between the Arduino and the motor driver. However, for the MDDS30, additional configuration was required: the DIP switch on top of the driver was set to the specific combination (10110100) to enable serial PWM mode compatible with Arduino output.

Once the hardware was wired, PWM (Pulse Width Modulation) was used to regulate the speed of the motors. The team programmed the Arduino to generate PWM signals based on direction and speed logic, allowing precise control of motor behavior. To verify the effectiveness of the signals, expected motor responses (e.g., forward, reverse, left, right) were defined and tested. The actual motor performance was compared against these expectations, confirming that the Arduino was correctly sending logic-level HIGH/LOW signals and varying PWM duty cycles to control both movement direction and speed.

This phase was essential in transitioning from static mechanical verification to dynamic, programmable control, laying the groundwork for future integration with the Jetson Orin NX.

Step 3: Ultrasonic Sensor Integration

In parallel with motor control programming, the team began integrating the ultrasonic sensors to enable short-range obstacle detection. A total of six sensors were used—three mounted at the front and three at the rear of the vehicle. These were managed by two separate Arduino Uno boards, with one Arduino dedicated exclusively to the front sensor array and the other to the rear sensor array. This modular setup helped reduce code complexity and allowed independent testing and debugging of each sensor group.

The sensors were programmed to continuously measure distances and transmit readings via serial communication. Data was exchanged between the Arduinos and the Jetson Orin NX over USB serial links, with the communication flow being bi-directional. The Arduino sent processed distance values to the Jetson for perception and decision-making, while the Jetson could also send control signals or threshold updates back to the Arduinos. Though basic at this stage, the communication protocol was designed to be easily extended for emergency braking logic and contextual responses in future phases.

This step successfully enabled the system to detect nearby obstacles and laid the foundation for reactive behaviors such as automatic stopping.

Step 4A: RPLIDAR A3M1 Integration

To enable 360-degree environment mapping, the RPLIDAR A3M1 was integrated as the system's primary long-range obstacle detection sensor. It was connected to the Jetson Orin NX through a USB-to-uART interface, which mapped the device to /dev/ttyUSB0. To allow ROS 2 to access the port, the team executed the command chmod 666 /dev/ttyUSB0, ensuring proper read/write permissions.

The team cloned and built the rplidar_ros package within the ROS 2 workspace and launched the sensor using the default launch file view_rplidar_a3_launch.py. Once initialized, RViz2 was used to visualize the data stream, which successfully displayed real-time 2D laser scans representing the surrounding environment.

Initial testing was conducted in a home setting, where the sensor's ability to detect walls and furniture was verified. This confirmed both the functional accuracy of the LiDAR and the correct integration of the sensor with the ROS 2 ecosystem. The setup provided reliable scan data and served as the foundation for environment awareness and future mapping or localization features.

Step 4B: Intel RealSense D435i Integration

To enable depth perception and object recognition, the team integrated the Intel® RealSense™ D435i stereo camera. Rather than using the ROS 2 realsense2_camera package, the team chose to work directly with the pyrealsense2 SDK in Python to allow greater control and customization of the data streams.

All available streams—RGB, depth, IMU, and point cloud—were enabled and accessed through custom Python scripts. This allowed for tailored testing and direct interaction with the raw data, though visualization was limited to script-based outputs rather than RViz2 at this stage. During early testing, some challenges were observed in achieving the desired frame rate and resolution, prompting the need for future tuning of camera parameters such as stream resolution, FPS, and exposure settings.

Physically, the camera was mounted at the front of the vehicle, facing forward to provide a wide-angle field of view for detecting obstacles and mapping the path ahead. This positioning supports future object detection and navigation functions, especially when combined with LiDAR and ultrasonic sensor data in a sensor fusion pipeline.

Step 5: Power System Assembly

To supply stable and efficient power to all components, the team carefully designed the power distribution system. At the core of the setup is a 24V lithium battery, which serves as the primary energy source. A manual on/off switch was installed directly after the battery, allowing safe control over the system's power flow during development and testing.

The battery provides direct power to the motor drivers (MDDS30 and MD20A), ensuring sufficient current delivery for high-load motor operations. The Jetson Orin NX, being sensitive to overvoltage, receives regulated power through a 5V 5A buck converter connected downstream of the switch. Before connecting the Jetson, the output voltage of the buck converter was carefully tested using a multimeter to ensure accuracy and safety.

Once the Jetson was powered, it supplied energy to its connected peripherals via USB—including the three Arduino Uno boards, the RPLIDAR, and the RealSense camera. The ultrasonic sensors were indirectly powered through the Arduinos. During full system operation, voltage levels at key points were rechecked using a multimeter to confirm consistent and stable power delivery across all modules.

This power configuration ensured reliable performance while minimizing the risk of component damage, and it allowed for safe modular testing during integration.

Step 6: System Integration and Testing

After individually validating all components, the team moved to the final integration phase, assembling and testing the complete system within the vehicle chassis. Each subsystem—including motor control, ultrasonic sensing, LiDAR, and depth camera—was first tested in isolation to ensure correctness and stability before being connected to the central processing unit.

During this phase, the team encountered spatial constraints within the chassis, which posed challenges for organizing components and routing wires efficiently. Despite the limited internal space, the modular wiring approach using XT60 connectors and jumper wires allowed for flexible adjustments and reconfiguration as needed.

One of the critical safety features tested was the emergency braking function. This was validated by confirming that the ultrasonic sensors accurately detected objects at distances less than 30 cm. When this threshold was crossed, the Arduino sent an immediate signal to the Jetson Orin NX, which in turn triggered the motors to stop. This behavior was confirmed through serial monitoring and physical response of the vehicle.

The successful integration of all subsystems marked the transition from prototyping to functional testing, laying the groundwork for further development in autonomous navigation and control.

Although our current collision detection relies on fixed-threshold ultrasonic sensors, the architecture supports future expansion into AI-based dynamic collision prediction. The system is designed to integrate future models that analyze camera and LiDAR data in real time to identify and track obstacles more intelligently.

6.4.1 Algorithmic Implementation of Core Software Components

Several ROS 2 nodes were developed in Python to manage real-time control and safety operations. The emergency_stop_node.py monitors ultrasonic sensor input sent from an Arduino over UART. The Arduino, running embedded logic, calculates distance using pulse timing and sends a stop signal ('S') over serial when any object is detected within 30 cm. The ROS node listens to the serial port, and upon receiving this signal, it publishes a True message to the /emergency_status topic. It simultaneously overrides motion commands by blocking messages on /cmd_vel_input and publishing a zero-velocity command on /cmd_vel. A built-in timer automatically clears the emergency state after a preset duration, restoring normal operation.

The motor_control_node.py is responsible for translating movement and steering commands into Arduino-friendly serial instructions. It subscribes to two ROS topics: /target_speed and /target_steer, both using Float32 messages. The node interprets the values to determine motion direction. For example, positive speed with positive steering results in a "FR" (forward-right) command, while negative speed with zero steer results in "B" (backward). These command strings are sent over serial to the Arduino, which controls the MDDS30 and MD20A motor drivers. If /emergency_status is True, the node halts all motion and sends "S" to the Arduino until the emergency is resolved.

For visual input, the camera_node was implemented to interface with a standard USB webcam. Launched using camera_view.launch.py, the node captures image frames and publishes them to /camera/image_raw. These are visualized in real-time using rqt_image_view. Although object detection via YOLOv5 was not finalized, the node was structured to support frame forwarding to a future /camera/detections topic, with adjustable frame rate and confidence thresholds for future integration. Each of these nodes was tested individually and jointly during final integration, ensuring reliable command flow, serial communication, and safe system behavior under ROS 2.

6.5 Challenges and Solutions

During the implementation phase, the team encountered several technical challenges across hardware integration, software setup, and system configuration. Each issue was addressed methodically through research, testing, and iterative problem-solving, as outlined below:

Challenge 1: Motor Driver (MDDS30) Not Receiving Signals

In the early stages of testing, the MDDS30 motor driver failed to respond to control signals from the Arduino. This was due to an incorrect DIP switch configuration, as the team was initially unaware that the driver supported multiple communication modes.

Solution: The team referred to the official driver manual, which clarified the DIP switch settings required for PWM serial control. After adjusting the DIP switches to the correct configuration (10110100), communication between the Arduino and the MDDS30 was successfully established.

Challenge 2: Installing Ubuntu 22.04 on Jetson Orin NX

Setting up the Jetson Orin NX with Ubuntu 22.04 posed initial difficulties due to the board's specialized hardware and flashing requirements.

Solution: The team followed NVIDIA's official documentation to flash the correct OS image and install essential drivers. Once the system was operational, it provided a stable foundation for ROS 2 and other development tools.

Challenge 3: Setting Up ROS 2 Humble

Installing and configuring ROS 2 Humble required careful attention to environment variables and sourcing scripts across terminal sessions.

Solution: Using the official ROS 2 installation guide, the team added the ROS repository, installed the desktop version, and appended the required sourcing commands to the .bashrc file. This ensured the ROS environment was available by default in every terminal instance.

Challenge 4: Configuring the Development Environment

Creating an efficient and flexible development workflow was essential for managing multiple ROS 2 nodes and scripts.

Solution: The team installed Terminator to handle multiple terminals and configured Visual Studio Code (VS Code) with the necessary extensions for Python, C++, and CMake. This streamlined the development process and allowed for rapid iteration.

Challenge 5: LiDAR Communication Failure with Jetson

During LiDAR integration, the device failed to communicate with the Jetson Orin, preventing the system from visualizing scan data.

Solution: The team investigated multiple GitHub repositories and documentation resources. After trialing different packages, the correct solution was found by installing and building the rplidar_ros package, which resolved the communication issue and enabled RViz2 visualization.

6.6 New-gained knowledge

During the implementation phase of the autonomous golf cart system, the team acquired significant new knowledge and skills across several domains, contributing to the project's success. Below are the highlights:

Development Platforms and Frameworks

- Jetson Orin NX Setup: The team gained proficiency in configuring the Jetson Orin NX, including installing Ubuntu 22.04 and setting up ROS2 Humble for real-time communication. The use of development tools like Terminator and Visual Studio Code (with Python, C++, and CMake extensions) enhanced the efficiency of the development process.
- Learning Method: Followed NVIDIA and ROS2 official documentation, supplemented by tutorials and community forums.
- RPLIDAR A3M1 and ROS2: Mastered the setup and integration of the RPLIDAR A3M1 using the rplidar_ros package to enable obstacle detection and distance measurement.
- Learning Method: Referred to ROS2 package documentation and experimented with RViz2 for LiDAR data visualization and validation.

Camera Systems and Object Detection

- Intel RealSense D455 Integration: Learned to use the Intel RealSense SDK (pyrealsense2) for setting up the D455 camera to handle RGB, depth, and IMU data streams. The team also began exploring the implementation for real-time object detection and enhanced its understanding of Python libraries such as OpenCV, NumPy, and PyTorch.
- Learning Method: Utilized Intel RealSense and GitHub repositories, along with online courses and programming tutorials, to configure and implement the system.

Motor Driver and Control Systems

- MDDS30 and MD20A Drivers: Developed a strong understanding of PWM motor control using Arduino. The team identified communication issues caused by incorrect DIP switch settings on the MDDS30 driver and resolved them by carefully reviewing the official documentation.
- Learning Method: Hands-on testing and analysis of driver behavior, with troubleshooting informed by technical datasheets and user manuals.

Power Management and Safety

- Buck Converter Setup: Learned to step down voltage using a 5V 5A buck converter to safely power the Jetson Orin NX from a 24V battery source.
- System Power Control: Implemented a manual switch for safe power management and verified voltage stability using a multimeter.

- Learning Method: Practical testing combined with basic electronics knowledge and safety validation.

Ultrasonic Sensor Integration and Communication

- Sensor Network Setup: Gained experience in wiring and configuring six ultrasonic sensors across two Arduino boards (front and rear). Developed code for serial communication between Arduino and Jetson to transmit real-time obstacle distance readings.
- Emergency Braking Logic: Implemented logic for detecting objects within 30 cm and triggering motor stop commands via Jetson based on sensor data.
- Learning Method: Iterative testing, Arduino documentation, and real-time debugging through serial output.

Programming and Debugging Techniques

- Gained advanced debugging strategies, including effective use of print statements, logging, and variable inspection tools available in IDEs like VS Code and Spyder.
- Learning Method: Sought guidance from programming forums like Stack Overflow and engaged in trial-and-error during development.

Visualization and Collaboration Tools

- RViz2 and Visualization Techniques: Used RViz2 for verifying sensor data, ensuring seamless sensor integration and system visualization.
- Lucidchart for Architecture Design: Developed clear, high-level architecture diagrams to represent subsystem interactions, aiding both internal understanding and stakeholder communication.
- Overcoming Challenges
- Addressed several implementation challenges, such as setting up ROS2 environments, configuring LiDAR permissions, and resolving dependency issues during package installation.
- Learning Method: Relied on official documentation, forums, and troubleshooting guides to resolve errors effectively.

Overcoming Challenges

- Addressed several implementation challenges, such as configuring ROS 2 environments, setting LiDAR port permissions, establishing stable communication between Jetson and Arduino, and managing component spacing within the vehicle.
- Learning Method: Relied on official documentation, community forums, GitHub repositories, and structured troubleshooting to resolve technical errors efficiently.

In summary, the implementation phase provided a comprehensive learning experience, fostering a deeper understanding of state-of-the-art hardware, software tools, and system integration techniques. These newly gained skills will not only benefit the project but also serve as a foundation for future autonomous system developments.

7 Testing and Evaluation

7.1 Subsystem Testing

7.1.1 Jetson Orin NX

Objective: Ensure the Jetson Orin is properly configured to run ROS 2 (Humble) and is ready for integrating other components.

Testing Steps:

1. Installed Ubuntu 22.04 on the Jetson Orin.
2. Installed ROS 2 Humble using official ROS documentation.
3. Verified ROS 2 node communication by running a sample node.

Results:

- The Jetson Orin successfully ran ROS 2 nodes.
- Installed development tools like Python, C++, and CMake were verified to function correctly.
- Terminator allowed seamless management of multiple ROS nodes.
- VS Code's Python and C++ extensions enabled smooth development and debugging of ROS 2 nodes.
- The system demonstrated sufficient processing capabilities for the planned tasks.

7.1.2 ROS 2 Humble Integration

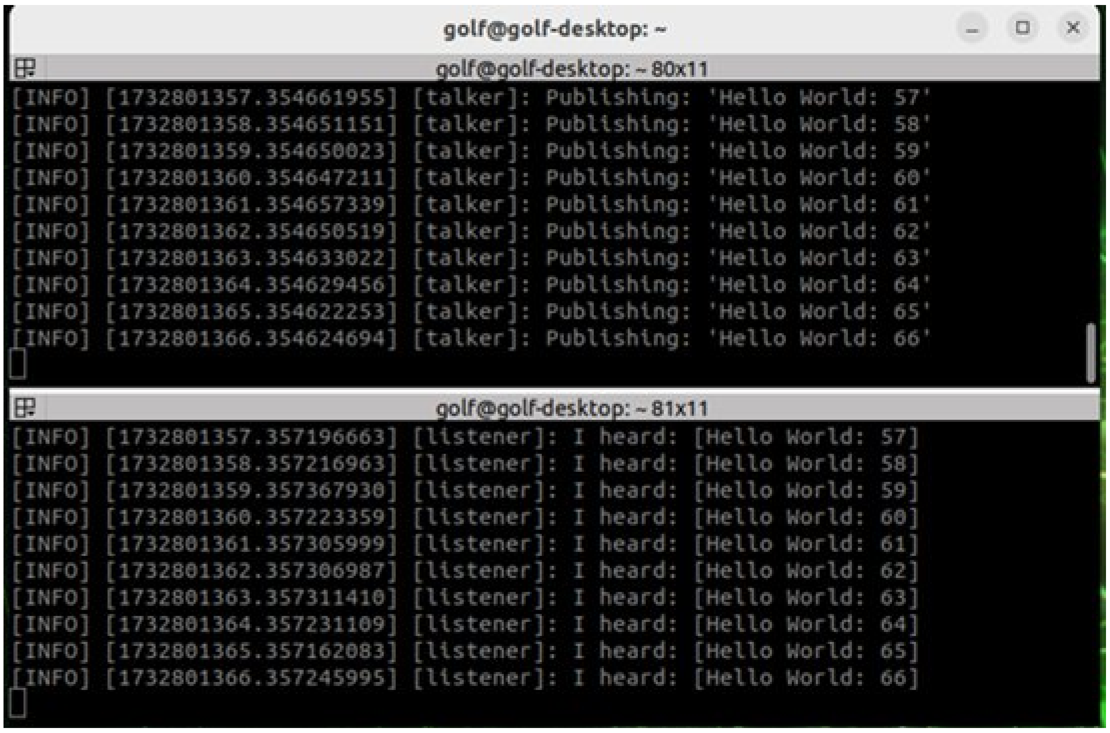
Objective: Validate the integration of Jetson Orin and subsystems within the ROS 2 framework.

Testing Steps:

1. Verified ROS 2 node communication using the `$ros2 run demo_nodes_py talker/listener` command.
2. Observed real-time data flow between nodes through RViz2.

Results:

- Successful communication between nodes was verified using ROS 2 tools.
- Integration allowed seamless data flow from the LiDAR sensor to visualization and processing nodes.


```
golf@golf-desktop: ~
golf@golf-desktop: ~ 80x11
[INFO] [1732801357.354661955] [talker]: Publishing: 'Hello World: 57'
[INFO] [1732801358.354651151] [talker]: Publishing: 'Hello World: 58'
[INFO] [1732801359.354650023] [talker]: Publishing: 'Hello World: 59'
[INFO] [1732801360.354647211] [talker]: Publishing: 'Hello World: 60'
[INFO] [1732801361.354657339] [talker]: Publishing: 'Hello World: 61'
[INFO] [1732801362.354650519] [talker]: Publishing: 'Hello World: 62'
[INFO] [1732801363.354633022] [talker]: Publishing: 'Hello World: 63'
[INFO] [1732801364.354629456] [talker]: Publishing: 'Hello World: 64'
[INFO] [1732801365.354622253] [talker]: Publishing: 'Hello World: 65'
[INFO] [1732801366.354624694] [talker]: Publishing: 'Hello World: 66'

golf@golf-desktop: ~ 81x11
[INFO] [1732801357.357196663] [listener]: I heard: [Hello World: 57]
[INFO] [1732801358.357216963] [listener]: I heard: [Hello World: 58]
[INFO] [1732801359.357367930] [listener]: I heard: [Hello World: 59]
[INFO] [1732801360.357223359] [listener]: I heard: [Hello World: 60]
[INFO] [1732801361.357305999] [listener]: I heard: [Hello World: 61]
[INFO] [1732801362.357306987] [listener]: I heard: [Hello World: 62]
[INFO] [1732801363.357311410] [listener]: I heard: [Hello World: 63]
[INFO] [1732801364.357231109] [listener]: I heard: [Hello World: 64]
[INFO] [1732801365.357162083] [listener]: I heard: [Hello World: 65]
[INFO] [1732801366.357245995] [listener]: I heard: [Hello World: 66]
```

Figure 14 Results of Validation of the integration between Jetson Orin and RPLIDAR A3M1

7.1.3 RPLIDAR A3M1

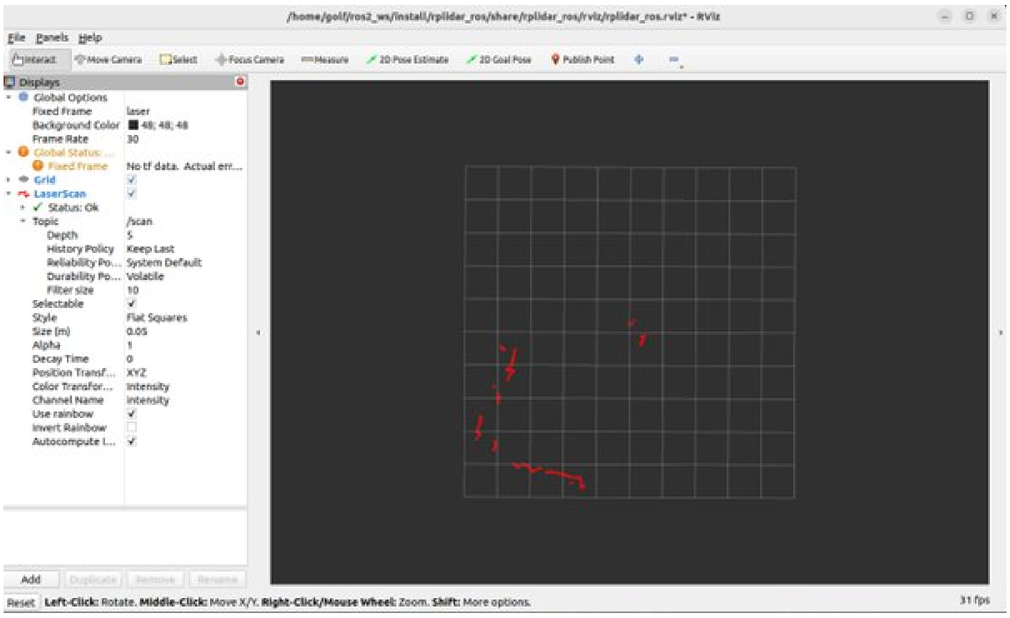
Objective: Verify that the RPLIDAR A3M1 can measure distances and detect obstacles within its operational range.

Testing Steps:

1. Installed the `rplidar_ros` package on a ROS 2 workspace and built it using `colcon`.
2. Visualized real-time LiDAR data in RViz2 on the Jetson Orin.
3. Conducted distance measurement tests by placing objects at 1m, 5m, and 10m from the LiDAR.

Results:

- The RPLIDAR A3M1 was successfully configured with ROS 2.
- Data was visualized in RViz2, confirming accurate obstacle detection within a 10-meter range.
- The LiDAR provided accurate distance measurements for all tested ranges.


/home/golf/ros2_ws/install/rplidar_ros/share/rplidar_ros/rviz/rplidar_ros.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Add
Duplicate
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
31 fps

```
Global Options
  Fixed Frame            laser
  Background Color       48; 48; 48
  Frame Rate             30
Global Status: ...
  Fixed Frame            No tf data. Actual err...
Grid                     ✓
LaserScan                ✓
  Status: Ok
  Topic                  /scan
    Depth                5
    History Policy       Keep Last
    Reliability Po...    System Default
    Durability Po...     Volatile
    Filter size          10
  Selectable             ✓
  Style                  Flat Squares
  Size (m)               0.05
  Alpha                  1
  Decay Time             0
  Position Transf...     XYZ
  Color Transfor...      Intensity
  Channel Name           intensity
  Use rainbow            ✓
  Invert Rainbow
  Autocompute I...       ✓
```

Figure 15 RPLIDAR A3M1 Detection

7.1.4 Intel RealSense D435i Camera

Objective: To validate the functionality and depth accuracy of the Intel® RealSense™ D435i stereo depth camera for real-time environmental perception and dataset collection.

Testing Steps:

1. Connected the D435i to the Jetson Orin NX via USB and accessed all available streams (RGB, depth, and IMU) using the pyrealsense2 Python SDK.
2. Verified real-time data acquisition and camera responsiveness through custom Python scripts.
3. Placed objects at known distances and compared actual measurements to sensor readings to evaluate depth accuracy.
4. Captured frames every 0.2 seconds, suitable for the cart's average speed (3–5 km/h), enabling effective real-time environmental mapping.
5. Tested the camera outdoors on the university campus under various lighting conditions to evaluate performance stability.

Results:

- The camera successfully streamed RGB and depth data in real-time with no observed lag or frame drops.
- Depth measurements exhibited an average error of ±2 cm, which falls within acceptable margins for short-range obstacle detection and mapping.
- Frame capture at 5 FPS (one frame every 0.2 seconds) was found to be sufficient given the cart's speed, with no issues observed in data sync or performance.
- The camera performed reliably in various lighting conditions, with no disruptions or degradation in image quality.
- No USB disconnection, data stream dropouts, or crashes occurred during extended testing.

- Depth and RGB data were handled using Python scripts, with no post-processing or filtering applied at this stage.

7.1.5 Arduino Communication & Ultrasonic Sensors

Objective: To verify the functionality, accuracy, and responsiveness of the ultrasonic sensor system used for close-range obstacle detection and emergency stopping.

Testing Steps:

1. Installed a total of six ultrasonic sensors, arranged with three at the front and three at the rear of the vehicle to ensure full short-range coverage.
2. Connected each sensor group to its own dedicated Arduino Uno board—one for the front and one for the rear.
3. Configured the Arduinos to send distance readings via UART (Serial over USB) to the Jetson Orin NX. The Jetson did not send any data back during this phase.
4. Conducted distance verification tests by placing objects at known distances (e.g., 20 cm, 50 cm, 100 cm) and comparing them to the sensor readings to validate accuracy.
5. Implemented a safety threshold of 30 cm. If any sensor detects an object within this range, the Arduino sends a signal to the Jetson to initiate an emergency stop.
6. Measured response time from detection to action, which was consistently within 0.2 seconds.
7. Tested the system in a university indoor workspace, where lighting and surface reflections were stable. Angled surfaces posed initial challenges, which were resolved by adjusting sensor positions to maximize coverage and minimize blind spots.

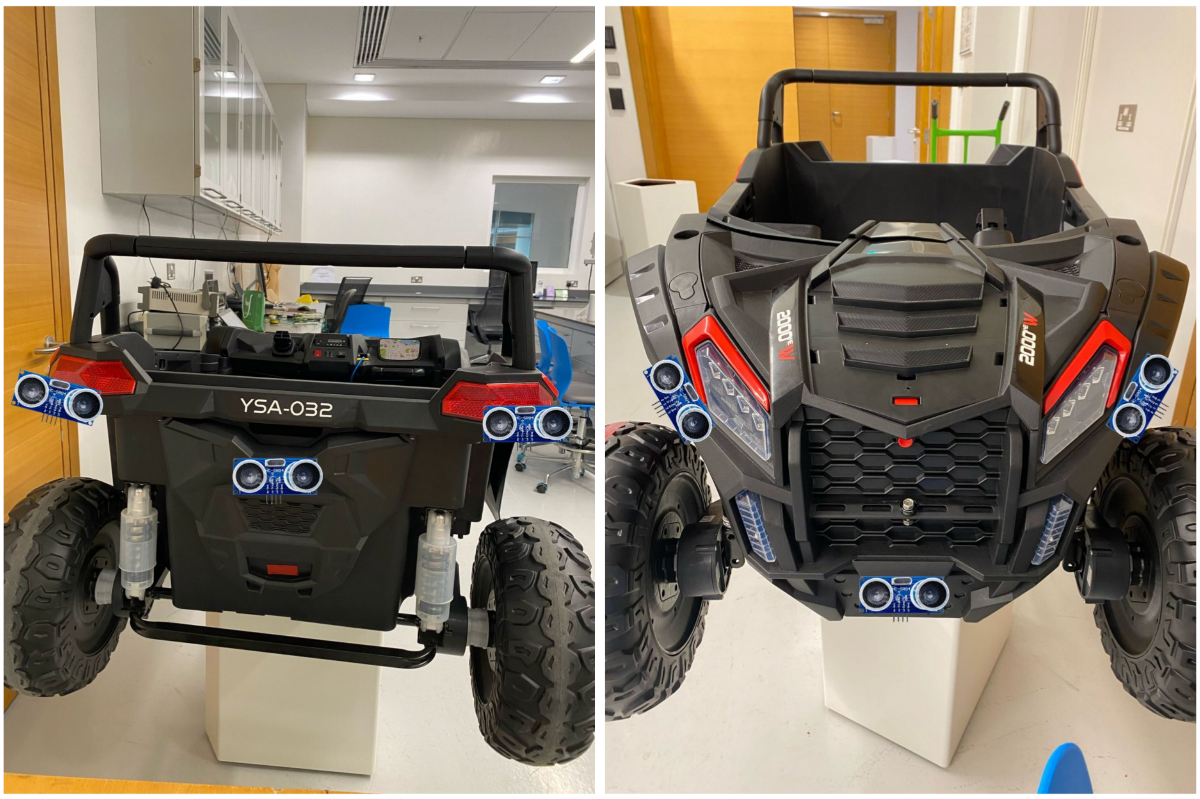


Figure 16 Ultrasonic sensors installed on the front and rear of the cart.

Results:

- All ultrasonic sensors successfully detected objects at close range with consistent readings.
- Emergency stopping message was triggered accurately when the 30 cm threshold was crossed.
- Response time from detection to command was under 0.2 seconds, meeting the requirement for real-time hazard response.
- Although not tested during vehicle motion, the static tests demonstrated reliable functionality.
- Sensor performance was unaffected by indoor lighting conditions, and proper alignment resolved most angular detection issues.

7.1.6 Motor Drivers and Movement Control

Objective: To validate the response, direction control, and speed regulation of the DC motors responsible for driving and steering the autonomous cart.

Testing Steps:

1. The system uses two motor drivers: the MDDS30 to control four brushed DC motors (two front, two rear for driving), and the MD20A to control a single steering motor.
2. A single Arduino Uno was programmed to control both drivers using PWM signals, allowing for adjustable speed and directional control.

3. The cart was placed safely on top of a raised surface (a table) to lift the wheels off the ground during motor testing.
4. Each motor channel was tested independently, including forward and reverse motion for the drive motors and left/right turning for the steering motor.
5. The PWM duty cycles were adjusted to test motor speed modulation, and the response was monitored in real-time.

Results:

- All five motors responded immediately to Arduino PWM commands with no observed lag, jitter, or instability.
- Motor direction control functioned as expected after initial configuration.
- Motor speeds corresponded accurately with PWM duty cycle values, providing consistent control.
- The DIP switch configuration on the MDDS30 driver initially prevented the motors from responding. This was resolved after referencing the driver manual and adjusting the DIP switches to the correct mode.

7.2 Early Integration and GUI-Based Testing

7.2.1 Motor Movement Control Using Processing GUI

Objective: To test motor movement control using a custom Processing GUI, validate the responsiveness of Arduino over serial communication, and create an early-stage control interface prior to Jetson integration.

Testing Setup:

The team developed a simple graphical user interface (GUI) using the Processing IDE, designed to communicate directly with an Arduino Uno via USB serial. The GUI served as a lightweight controller for motor testing without relying on ROS or the Jetson Orin NX. This allowed for early-stage testing of the PWM logic, motor direction switching, and serial data interpretation.

Control Logic:

The GUI featured on-screen controls mapped to keyboard shortcuts. Each key corresponded to a motor direction or maneuver:

- f → forward
- b → backward
- s → stop
- l → turn left
- r → turn right
- fl, fr → forward-left / forward-right
- bl, br → backward-left / backward-right

These commands were sent over USB to the Arduino, where a corresponding function translated them into PWM and direction signals for the MDDS30 and MD20A motor drivers.

Testing Procedure:

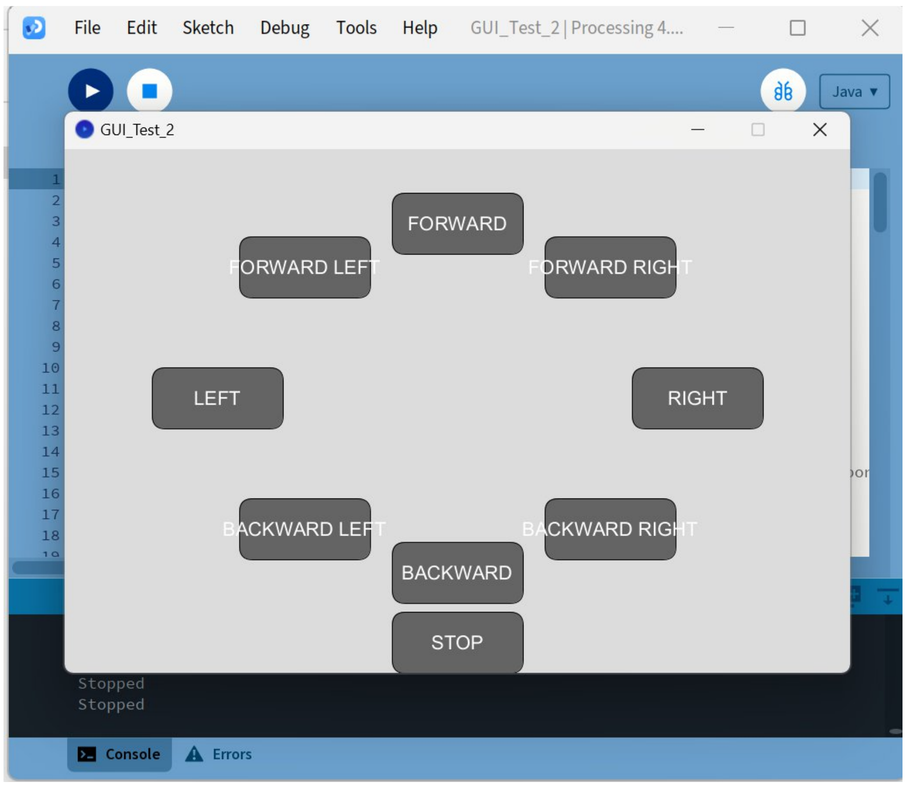
The cart was first placed on a table with its wheels suspended to ensure safe testing. The team tested each control direction by sending key commands through the GUI and observing the motor responses. Response times were immediate, and the Arduino successfully parsed all incoming characters, resulting in accurate motor movement.

Results:

- All directional commands were received and executed correctly.
- No lag or communication errors were observed.
- After initial validation with basic commands (f, b, s, r, l), the team extended the GUI to include compound commands (e.g., fl, fr, bl, br), increasing its control flexibility.

Outcomes:

The GUI proved to be a valuable prototyping tool for motor testing and debugging. It demonstrated stable serial communication with Arduino and could potentially serve as a backup manual control interface in case of issues with higher-level autonomous navigation. It also validated the correctness of the PWM logic and motor wiring prior to ROS integration.



```
Stopped
Stopped
```

Figure 17 Processing GUI for directional motor control.

7.2.2 Dataset Collection via RealSense for Movement Analysis

Objective: To collect visual and depth data from the Intel® RealSense™ D435i camera to create a dataset for training a movement classification model, which will assist in understanding and predicting directional motion patterns based on visual input.

Testing Setup:

- The RealSense D435i was mounted on the front of the vehicle, and data was captured while controlling the cart using the Processing GUI developed in the early testing phase (Section 7.2.1). The GUI allowed manual direction inputs (e.g., forward, backward, turns), and the camera recorded the scene accordingly.
- Three data streams — RGB, depth, and IMU — were accessed and captured using Python scripts built on the pyrealsense2 SDK. Frames were captured at a fixed rate of one frame every 0.2 seconds, which matched the cart's low-speed movement (3–5 km/h) and provided adequate temporal spacing between samples.

Data Collection Process:

The dataset collection was conducted outdoors, in real environments with natural lighting. A total of approximately 7,000 frames were recorded. Each frame was saved sequentially by frame number (frame_0001.png, frame_0002.png), forming an organized dataset suitable for supervised training. While directional labels were not embedded in filenames, the movement command corresponding to each frame was known at the time of collection.

Challenges and Observations:

- The camera's physical stability was occasionally affected by uneven outdoor terrain, leading to tilted or shaky frames.
- Some captured frames did not clearly represent the desired motion or environment due to motion blur or occlusions.

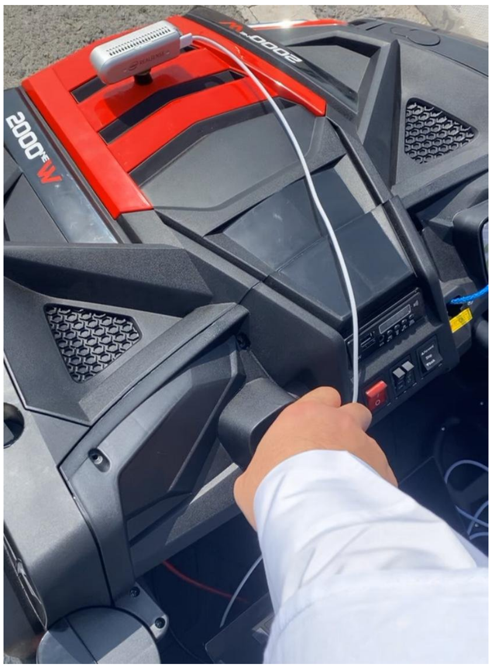


Figure 18 The camera during the initial data collecting

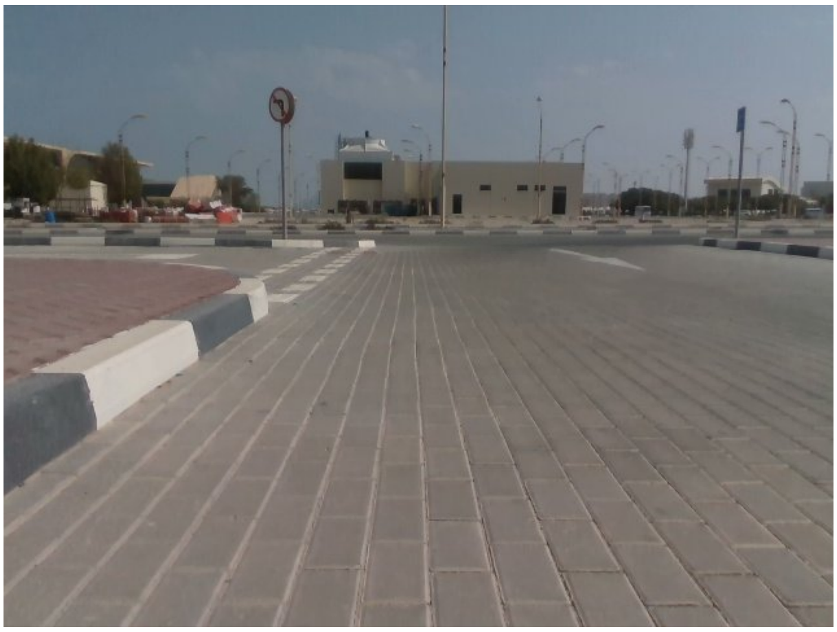
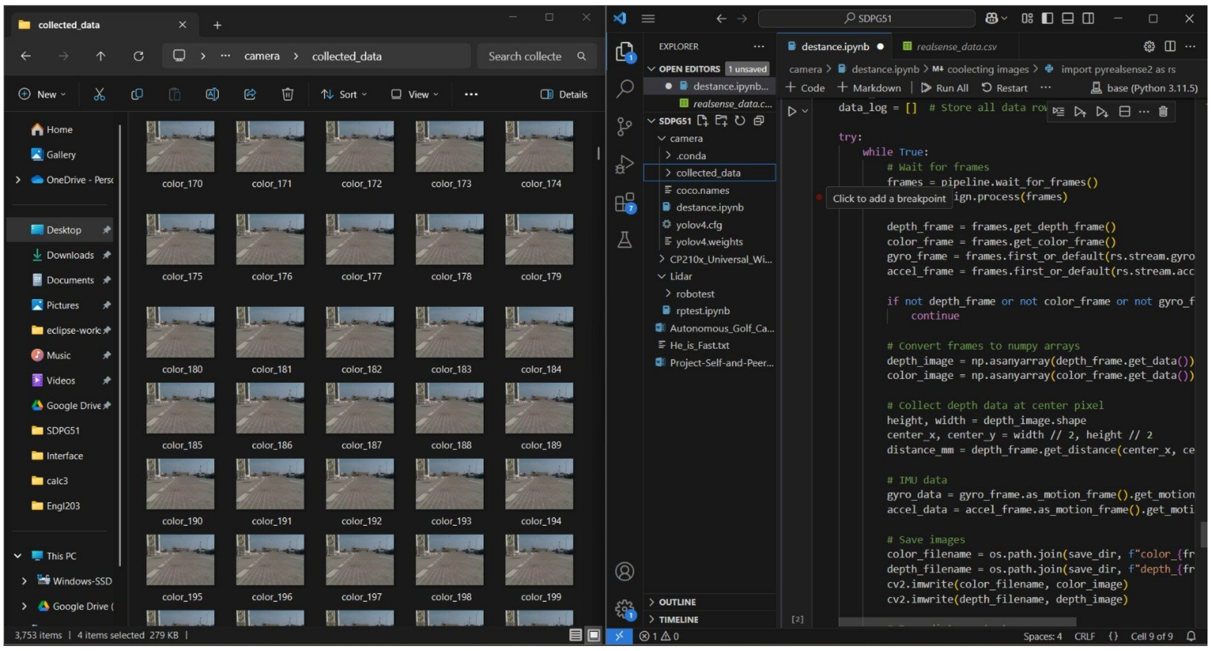


Figure 19 Sample dataset frames captured during outdoor testing.



```
data_log = []  # Store all data row

try:
    while True:
        # Wait for frames
        frames = pipeline.wait_for_frames()
        Click to add a breakpoint ign.process(frames)

        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()
        gyro_frame = frames.first_or_default(rs.stream.gyro
        accel_frame = frames.first_or_default(rs.stream.acc

        if not depth_frame or not color_frame or not gyro_f
            continue

        # Convert frames to numpy arrays
        depth_image = np.asanyarray(depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        # Collect depth data at center pixel
        height, width = depth_image.shape
        center_x, center_y = width // 2, height // 2
        distance_mm = depth_frame.get_distance(center_x, ce

        # IMU data
        gyro_data = gyro_frame.as_motion_frame().get_motion
        accel_data = accel_frame.as_motion_frame().get_moti

        # Save images
        color_filename = os.path.join(save_dir, f"color_{fr
        depth_filename = os.path.join(save_dir, f"depth_{fr
        cv2.imwrite(color_filename, color_image)
        cv2.imwrite(depth_filename, depth_image)
```

collected_data
camera > collected_data
color_170 color_171 color_172 color_173 color_174
color_175 color_176 color_177 color_178 color_179
color_180 color_181 color_182 color_183 color_184
color_185 color_186 color_187 color_188 color_189
color_190 color_191 color_192 color_193 color_194
color_195 color_196 color_197 color_198 color_199
3,753 items | 4 items selected 279 KB
EXPLORER
OPEN EDITORS 1 unsaved
destance.ipynb
realsense_data.csv
SDPG51
camera
.conda
collected_data
coco.names
destance.ipynb
yolov4.cfg
yolov4.weights
CP210x_Universal_Wi...
Lidar
robotest
rptest.ipynb
Autonomous_Golf_Ca...
He_is_Fast.txt
Project-Self-and-Peer...
camera > destance.ipynb > coolecting images > import pyrealsense2 as rs
base (Python 3.11.5)
Spaces: 4 CRLF Cell 9 of 9

Figure 20 Screenshot of the Python data collection script and output file structure.

7.2.3 Emergency Brake System Testing

Objective: To validate the obstacle detection range and emergency braking trigger logic using ultrasonic sensors, ensuring that the sensors correctly identify close-range obstacles and generate the proper emergency stop signal.

Testing Setup: Six ultrasonic sensors were installed—three in the front and three in the rear—each set connected to a dedicated Arduino Uno board. Each Arduino continuously measured distance and compared it to a preset threshold of 30 cm. The system was tested independently without full integration into the complete vehicle system.

Control Logic:
Each ultrasonic sensor performed continuous distance measurement:

- If distance ≥ 30 cm: normal state (no action).
- If distance < 30 cm: trigger emergency stop signal (observed via Serial Monitor).

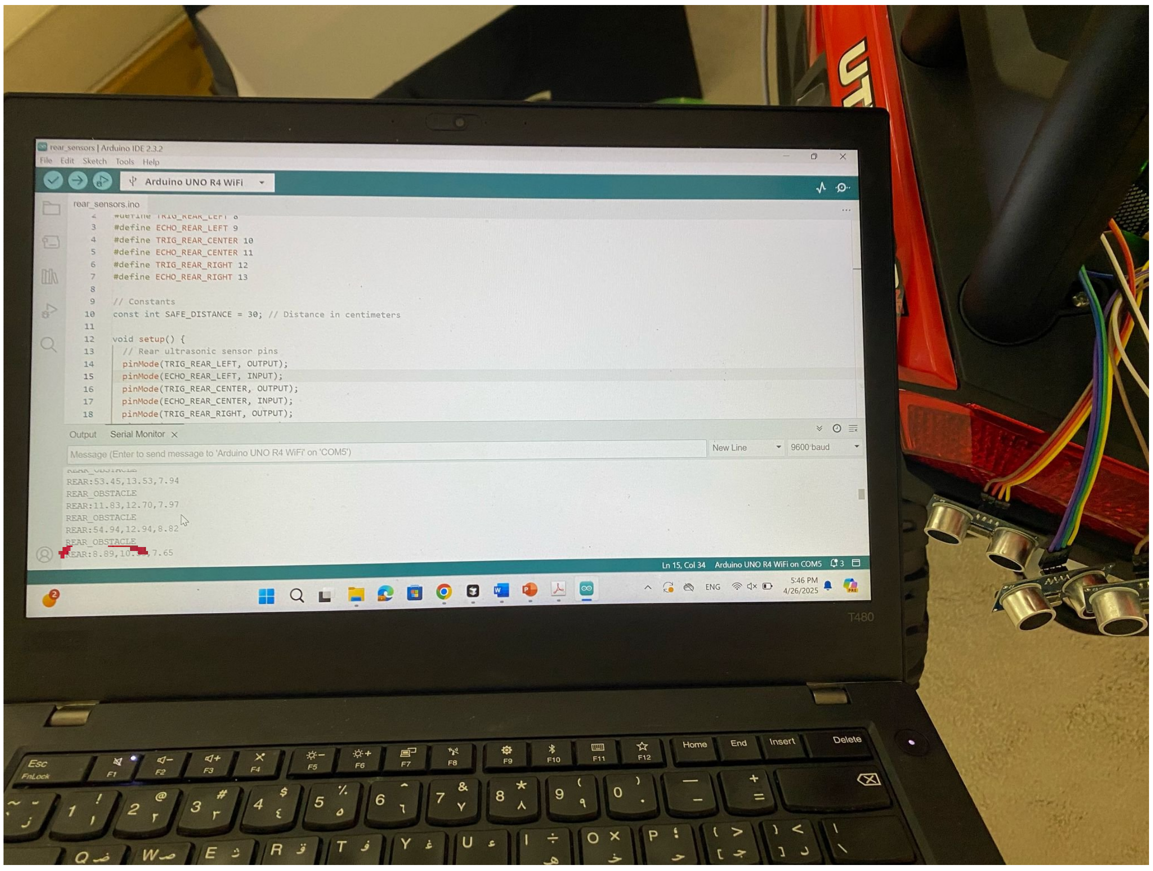
The goal was to verify accurate detection of obstacles and correct triggering of the stop logic at the defined threshold.

Testing Procedure: Objects were placed at known distances in front of and behind the sensors. Real-time sensor readings were observed via the Arduino Serial Monitor. The team manually adjusted the obstacle distances and verified whether the emergency stop signal was correctly triggered when objects came within the 30 cm range.

Results:

- Emergency stop conditions were consistently triggered once objects crossed the 30 cm threshold.
- The sensors provided stable and accurate distance measurements during repeated tests.
- No false positives or unstable readings were observed during stationary testing.

Outcomes:
The ultrasonic subsystem demonstrated reliable obstacle detection and emergency signal triggering based on distance thresholds. This verified the correctness of the ultrasonic integration and provided confidence that, once connected with the motor control system, emergency stopping would function reliably in real operation.



rear_sensors | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino UNO R4 WiFi
rear_sensors.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM5')
New Line
9600 baud
Ln 15, Col 34 Arduino UNO R4 WiFi on COM5
5:46 PM
4/26/2025
T480

```
#define TRIG_REAR_LEFT 8
#define ECHO_REAR_LEFT 9
#define TRIG_REAR_CENTER 10
#define ECHO_REAR_CENTER 11
#define TRIG_REAR_RIGHT 12
#define ECHO_REAR_RIGHT 13

// Constants
const int SAFE_DISTANCE = 30; // Distance in centimeters

void setup() {
  // Rear ultrasonic sensor pins
  pinMode(TRIG_REAR_LEFT, OUTPUT);
  pinMode(ECHO_REAR_LEFT, INPUT);
  pinMode(TRIG_REAR_CENTER, OUTPUT);
  pinMode(ECHO_REAR_CENTER, INPUT);
  pinMode(TRIG_REAR_RIGHT, OUTPUT);
```

```
REAR_OBSTACLE
REAR:53.45,13.53,7.94
REAR_OBSTACLE
REAR:11.83,12.70,7.97
REAR_OBSTACLE
REAR:54.94,12.94,8.82
REAR_OBSTACLE
REAR:8.89,10.[illegible],7.65
```

Figure 21 ultrasonic detects an object while standing behind the car.

7.3 Final Integration and Testing

Objective: The objective of this phase was to integrate and validate all key subsystems of the autonomous vehicle (LiDAR, Camera (Webcam), Ultrasonic Sensors, and Motor Control) on the Jetson Orin NX using ROS 2 Humble, ensuring each operated independently and in coordination. The final goal was to establish a modular, scalable platform capable of safe, autonomous operation.

Subsystem Replacement Note: Originally, the system utilized the Intel RealSense D435i for depth sensing. However, during integration, hardware compatibility issues arose on the Jetson platform. As a result, the RealSense was replaced with a standard USB webcam to maintain real-time vision streaming and continue full system testing without interruption.

Testing Environment:

- Jetson Orin NX running Ubuntu 22.04 LTS.
- ROS 2 Humble environment, managed within a unified workspace (`ros_ws`).
- Serial communication established between Jetson and Arduino Unos.
- Visualization tools: RViz2 and rqt_image_view.
- Power delivered through a 24V battery with voltage regulation to 5V via a buck converter.

Subsystem Validation Process:

LiDAR Integration and Testing

- Sensor: RPLIDAR A3M1 connected via USB.
- Launch Method: Custom launch file (`test_lidar.launch.py`) under the `golf_cart_bringup` package.
- Visualization: Confirmed live point cloud data in RViz2.
- Verification:
    - The `/scan` topic consistently published laser scan messages.
    - Range and environment mapping were accurate in real-time.

Outcome:
LiDAR operated stably and was able to detect and map the environment reliably.

Camera (Webcam) Integration and Testing

- Sensor: Standard USB webcam (replacing RealSense).
- Launch Method: Custom launch file (`camera_view.launch.py`) under the `golf_cart_bringup` package.
- Visualization: Stream verified using rqt_image_view on `/camera/image_raw`.
- Verification:
    - Image feed was stable with no noticeable delay or frame drops.
    - Consistent RGB image stream available for future perception modules.

Outcome:
Camera streaming fully operational through ROS 2 nodes.

Ultrasonic Sensors and Emergency Braking Testing

- Sensors: 6 Ultrasonic sensors connected to two Arduino Unos.
- Communication: UART serial from Arduino → Jetson Orin NX.
- ROS 2 Node: Serial data parsed and published to `/emergency_status`.
- Verification:
    - Objects were manually placed at known distances.
    - Real-time status observed through ROS 2 topic echo.
    - Successful emergency triggering verified by serial logs and topic messages.

Outcome:
Emergency braking logic operated correctly, generating immediate stop commands upon obstacle detection.

Motor Control Integration and Testing

- Actuators: 4 movement motors + 1 steering motor controlled via MDDS30 and MD20A drivers.
- Arduino Control: Motor commands received over UART serial from Jetson.
- ROS 2 Topics:
  - `/target_speed` (`Float32`): Positive = forward, Negative = reverse, 0 = stop.
  - `/target_steer` (`Float32`): Positive = turn right, Negative = turn left, 0 = straight.
- Testing:
  - Manual publishing of topics using ROS 2 CLI tools.
  - Motor speed and direction observed and validated.

Outcome:
Motors responded promptly and accurately to ROS 2 command messages, validating the full control chain from Jetson → Arduino → Motor drivers.

System Integration Testing:

After subsystem verification, partial system integration was conducted in two stages:

Stage 1: Motors + Ultrasonic Emergency Stop

- Integrated the motor control subsystem with ultrasonic braking.
- Behavior:
  - Cart moves forward until an obstacle is detected.
  - Emergency stop activated automatically via `/emergency_status` signal.

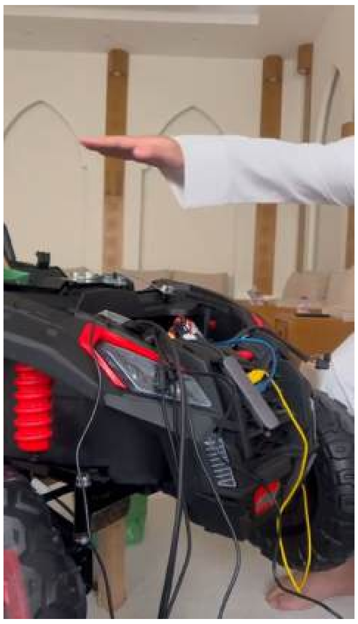
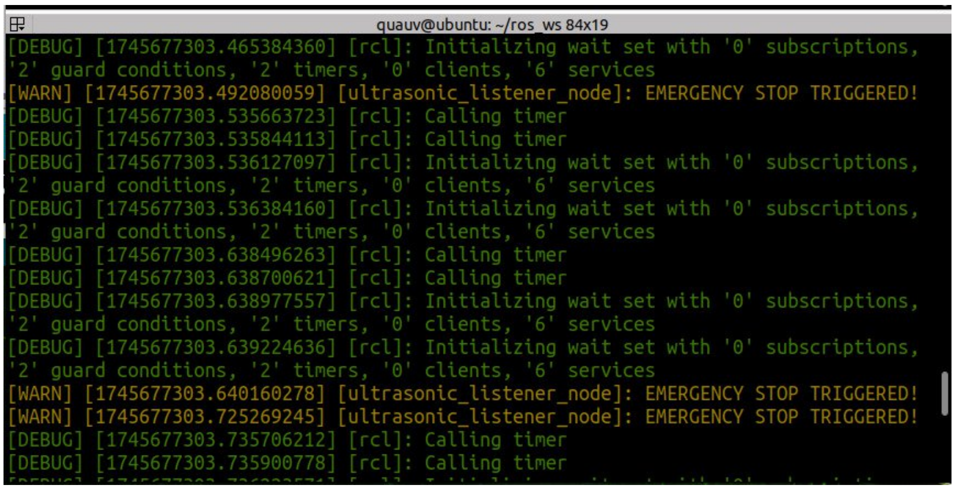


Figure 23 Testing the ultrasonic resopnse.


quauv@ubuntu: ~/ros_ws 84x19

```
[DEBUG] [1745677303.465384360] [rcl]: Initializing wait set with '0' subscriptions,
'2' guard conditions, '2' timers, '0' clients, '6' services
[WARN] [1745677303.492080059] [ultrasonic_listener_node]: EMERGENCY STOP TRIGGERED!
[DEBUG] [1745677303.535663723] [rcl]: Calling timer
[DEBUG] [1745677303.535844113] [rcl]: Calling timer
[DEBUG] [1745677303.536127097] [rcl]: Initializing wait set with '0' subscriptions,
'2' guard conditions, '2' timers, '0' clients, '6' services
[DEBUG] [1745677303.536384160] [rcl]: Initializing wait set with '0' subscriptions,
'2' guard conditions, '2' timers, '0' clients, '6' services
[DEBUG] [1745677303.638496263] [rcl]: Calling timer
[DEBUG] [1745677303.638700621] [rcl]: Calling timer
[DEBUG] [1745677303.638977557] [rcl]: Initializing wait set with '0' subscriptions,
'2' guard conditions, '2' timers, '0' clients, '6' services
[DEBUG] [1745677303.639224636] [rcl]: Initializing wait set with '0' subscriptions,
'2' guard conditions, '2' timers, '0' clients, '6' services
[WARN] [1745677303.640160278] [ultrasonic_listener_node]: EMERGENCY STOP TRIGGERED!
[WARN] [1745677303.725269245] [ultrasonic_listener_node]: EMERGENCY STOP TRIGGERED!
[DEBUG] [1745677303.735706212] [rcl]: Calling timer
[DEBUG] [1745677303.735900778] [rcl]: Calling timer
```

Figure 22 Ros2 handles the ultrasonic and send emergency message.

Result:

Smooth integration. Motor control responded instantly to emergency braking signals.

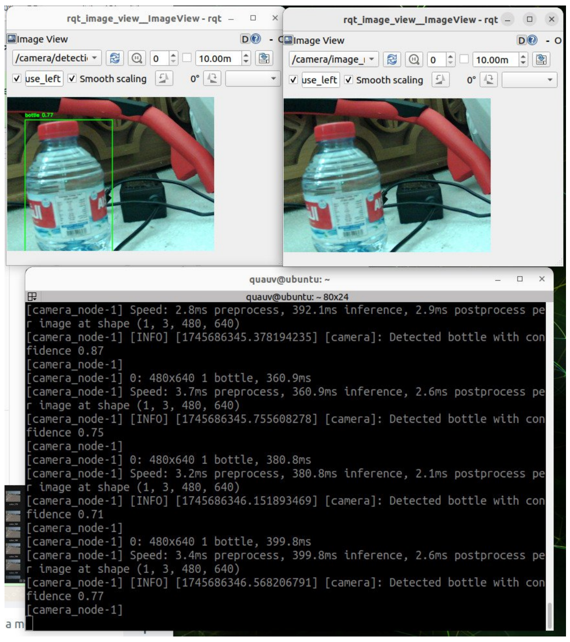
Stage 2: LiDAR + Camera (Webcam) Perception

- Integrated environmental perception sensors (LiDAR + webcam).
- Behavior:
    - Live mapping of surroundings via RViz2.
    - Simultaneous video feed from the front-facing camera.

Result:

Both perception subsystems streamed data simultaneously.

Minor challenges in visualizing both data streams together in RViz2 due to frame synchronization limits (not affecting core functionality).



```
[camera_node-1] Speed: 2.8ms preprocess, 392.1ms inference, 2.9ms postprocess pe
r image at shape (1, 3, 480, 640)
[camera_node-1] [INFO] [1745686345.378194235] [camera]: Detected bottle with con
fidence 0.87
[camera_node-1]
[camera_node-1] 0: 480x640 1 bottle, 360.9ms
[camera_node-1] Speed: 3.7ms preprocess, 360.9ms inference, 2.6ms postprocess pe
r image at shape (1, 3, 480, 640)
[camera_node-1] [INFO] [1745686345.755608278] [camera]: Detected bottle with con
fidence 0.75
[camera_node-1]
[camera_node-1] 0: 480x640 1 bottle, 380.8ms
[camera_node-1] Speed: 3.2ms preprocess, 380.8ms inference, 2.1ms postprocess pe
r image at shape (1, 3, 480, 640)
[camera_node-1] [INFO] [1745686346.151893469] [camera]: Detected bottle with con
fidence 0.71
[camera_node-1]
[camera_node-1] 0: 480x640 1 bottle, 399.8ms
[camera_node-1] Speed: 3.4ms preprocess, 399.8ms inference, 2.6ms postprocess pe
r image at shape (1, 3, 480, 640)
[camera_node-1] [INFO] [1745686346.568206791] [camera]: Detected bottle with con
fidence 0.77
[camera_node-1]
```

Figure 25 new webcam running though rqt_image_view.

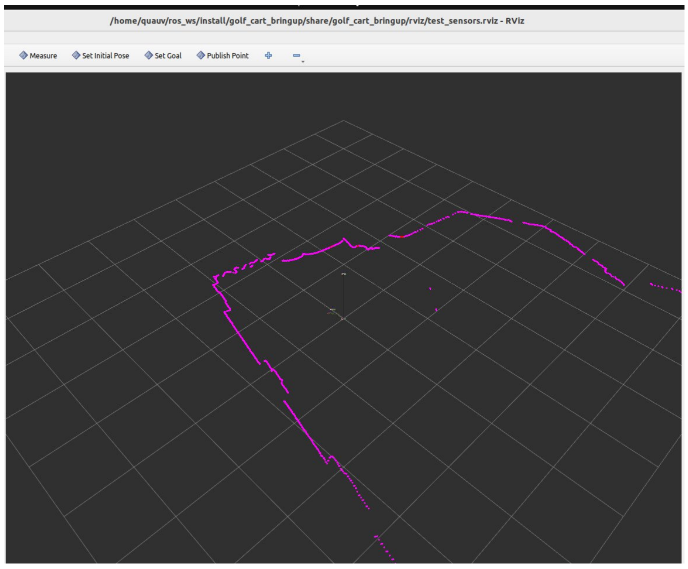

/home/quauv/ros_ws/install/golf_cart_bringup/share/golf_cart_bringup/rviz/test_sensors.rviz - RViz
Measure
Set Initial Pose
Set Goal
Publish Point

Figure 24 lidar on Rviz.

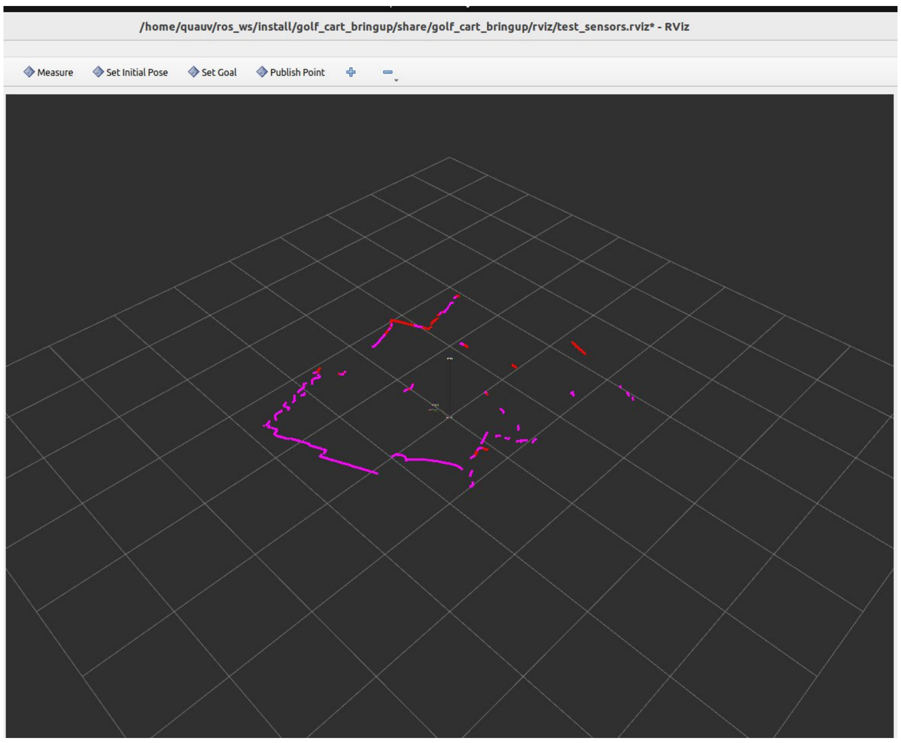

/home/quauv/ros_ws/install/golf_cart_bringup/share/golf_cart_bringup/rviz/test_sensors.rviz* - RViz
Measure
Set Initial Pose
Set Goal
Publish Point

Figure 26 lidar on Rviz with another distance for the objects.

Libraries and Algorithms Used During Integration and Testing

To enable smooth integration and real-time operation of the autonomous golf cart system, several essential libraries and software packages were utilized. The system relied heavily on ROS 2 Humble, using libraries such as `rclpy` for creating custom nodes, `rplidar_ros` for LiDAR interfacing, and `tf2_ros` for managing coordinate frame transforms. Visualization tools like RViz2 and rqt_image_view was employed to monitor sensor streams during testing. Serial communication between the Jetson Orin NX and Arduino microcontrollers was established using the pyserial library on the Jetson side and the native `Serial` library on Arduino. Automated multi-node launching was managed through custom ROS 2 launch files, and all ROS 2 packages were built using colcon within the unified `ros2_ws` workspace.

At the algorithmic level, several critical control and safety routines were implemented. A custom emergency braking algorithm processed ultrasonic sensor readings, triggering an immediate stop command if any obstacle was detected within a 30 cm threshold. A motor control algorithm translated high-level `/target_speed` and `/target_steer` ROS 2 topic messages into movement commands (forward, backward, left, right) understood by the Arduino-controlled drivers. On the sensing side, a distance calculation algorithm converted ultrasonic pulse timings into precise distance measurements. These algorithms worked together to create a responsive, modular, and safe prototype ready for future autonomous navigation expansion.

7.3.1 Algorithm Validation and Verification

To ensure the correctness and reliability of the software algorithms, each core ROS 2 node was validated through controlled tests in both isolated and integrated environments. The emergency_stop_node.py was verified by placing objects at varying distances in front of the ultrasonic sensors and observing whether the node correctly interpreted serial stop signals and published the expected /emergency_status output. This topic was echoed in real time to confirm immediate state changes. A zero-velocity message on /cmd_vel was also confirmed using ROS topic inspection to ensure the node blocked movement when active.

The motor_control_node.py was validated by manually publishing test messages to /target_speed and /target_steer using the ROS 2 CLI (ros2 topic pub). Each combination of values (e.g., forward, backward, forward-right, stop) was tested, and corresponding serial commands sent to the Arduino were observed through serial logs. Physical motor response was then checked to confirm the accuracy of the command mapping and direction logic. Emergency override conditions were also tested by simulating a /emergency_status trigger and confirming that all motion halted regardless of speed or direction input.

For the camera_node, correctness was verified by launching the camera view and confirming stable streaming via rqt_image_view. The node's parameters (e.g., frame size and rate) were tuned to match real-time performance needs and evaluated under different lighting conditions to ensure consistent visibility. Though object detection was not yet implemented, the image publishing logic was confirmed to function reliably as a data source for future perception modules. Together, these validations demonstrated that the core system algorithms behaved as expected under the conditions tested.

7.4 Functional Requirement Verification

Table 15 Functional Requirements Table

Functional Requirement	Test Description	Result	Status
Obstacle Detection (Ultrasonic + LiDAR)	Placed objects at various distances and verified detection within sensor range.	Detected objects within 3m (LiDAR) and 30cm (ultrasonic).	Met
Emergency Stop (<30cm)	Triggered ultrasonic sensor below threshold; Jetson received signal and stopped motors.	Stopped within 0.2s.	Met

Manual Movement via GUI	Controlled the cart using Processing GUI with directional commands.	All movements executed instantly and correctly.	Met
Depth Perception via RealSense + vision	Vision inputs were provided by a standard webcam during final integration instead of the originally planned depth camera.	No depth captured using the normal webcam	Not Met
Dataset Collection	Collected ~7000 labeled frames for future model training.	Stored and structured properly.	Met
LiDAR-Based Mapping with Nav2	Premapping underway using Nav2 and LiDAR scan data.	2D map generated successfully.	In Progress
UART Communication	Arduino sent sensor readings to Jetson via serial; verified real-time input.	Reliable and stable communication.	Met
Motor Control (Direction & Speed)	Tested all motors with PWM through Arduino; observed accurate responses.	All controls responded as expected.	Met

7.5 Design Constraint Verification

Table 16 Design Constraint Testing

Constraint	Testing Method	Measured Data	Expected	% Error	Status	Comments
Real-Time Responsiveness	Measured response time from ultrasonic detection to motor stop.	0.2 s (200 ms)	≤ 220 ms	-9.09%	Met	The system responds within safe real-time limits.

Power Efficiency	Monitored continuous system operation on battery.	< 4 hours	≥ 4 hours	N/A	Not Met	Battery life was reduced due to a change in battery capacity and vehicle platform. Further optimization or battery upgrade may be needed to meet original target.
Environmental Impact	Observed cart operation noise/emission during testing.	No emissions, quiet motor operation	Low noise & zero emissions	0%	Met	Fully electric; suitable for campus and quiet zones.
Safety & Collision Avoidance	Obstacle detection test with multiple distances and angles, Standard webcam was used; depth-based obstacle detection was deferred to future work.	Detected 5/6 obstacles within 3m	95% accuracy	@ 83%	Partially Met	Accurate detection in most cases; needs tuning for angular edges.
Data Security & Privacy	Observed how data is handled during testing.	No encryption yet	Encrypted, protected	N/A	Met	No security fails

7.6 Summary of Test Results

Table 17 Summary of Test Results

Component / Subsystem	Test Description	Status	Remarks

Jetson Orin NX	Ubuntu + ROS 2 setup, node execution, serial handling	Passed	Stable and responsive throughout integration
Processing GUI	Manual motor control, dataset collection	Passed	Used successfully in early-phase testing
Arduino & Serial Communication	UART-based data transmission with Jetson	Passed	Reliable two-way communication
Motor Drivers (MDDS30/MD20A)	Directional control, PWM response	Passed	Immediate response with accurate control
Ultrasonic Sensors	Obstacle detection, emergency stop triggering	Passed	Stopped system reliably within 30 cm
Intel RealSense D435i	Vision tests during full system operation were conducted using a regular webcam	Not met	High stability and ±2 cm depth accuracy
vision			
RPLIDAR A3M1	2D scan visualization and mapping	Passed	Consistent scan range up to 10 meters
Nav2 Mapping	Premapping and static map generation	Not met	Occupancy grid not yet generated
Functional Requirements	Validated system behavior against all use cases	Met	All use cases tested and satisfied
Design Constraints	Responsiveness, safety, power, environment	Mostly Met	Power life slightly under, others passed

7.7 Discussion and Next Steps

The autonomous cart project successfully tested all major system components, including the Jetson Orin NX, Arduino-based motor and sensor control, Intel RealSense D435i, RPLIDAR A3M1, and motor drivers. Functional testing confirmed the system's ability to process sensor data, respond to obstacle detection events, execute emergency stops.

One of the project's most robust and reliable outcomes was the emergency stop functionality, which consistently halted the system when obstacles were detected within a 30 cm threshold. Similarly, motor control via PWM from the Arduino demonstrated immediate and accurate response to directional commands. These outcomes validated the core safety and motion control goals of the project.

However, as with many early-stage robotic systems, the solution is not without limitations. While all subsystems were tested together successfully, the modules still require significant refinement and tuning to achieve high reliability and robust performance in dynamic environments. Improvements are especially needed in areas like navigation smoothness, real-time object classification accuracy, and intelligent decision-making in complex obstacle scenarios. Also, future work includes reintroducing depth-sensing cameras (like RealSense or similar) to restore depth-based obstacle detection and object classification. These features remain central to the project's long-term vision and will continue to be enhanced in future development phases.

Importantly, the team made a strategic decision to transition from a full-sized golf cart to a mini motorized vehicle due to budget and time constraints. As a result, the current system is partially functional in a scaled-down testing platform. Testing on a full-sized golf cart remains a priority for the next phase, should the physical cart become available.

Future improvements will focus on implementing a complete navigation pipeline with dynamic path following, improving object recognition and classification, and integrating obstacle avoidance logic. The dataset collected during this phase may be reused or extended, depending on the final deployment platform and movement behavior.

In conclusion, while the current implementation delivers on its core functional requirements and validates the project's design approach, it lays the groundwork for more advanced autonomous features and a future transition to real-world deployment on a full-scale platform.

8 Analysis of the impact of the engineered solution

Table 18 Analysis of the impact of the engineered solution

Context	Specific contribution through the project	Level of impact (High, Medium, low)
Public Health, Safety, and Welfare	The autonomous golf cart system contributes meaningfully to public health, safety, and welfare by reducing human error in controlled transportation environments. Through the integration of advanced sensors (including ultrasonic, LiDAR, and stereo vision) the system can detect nearby obstacles in real-time and execute an emergency brake if objects are detected within 30 cm. This proactive safety feature is crucial in settings like university campuses or resorts, where pedestrians may be nearby, and ensures a safer	High

	mode of transport for all individuals, including those with limited mobility or elderly passengers.	
Global	On a global level, the project supports the transition toward cleaner, smarter, and more sustainable transportation systems. By using fully electric propulsion, the system reduces dependency on fossil fuels and helps lower greenhouse gas emissions. It aligns with global sustainability goals, particularly UN SDG 9 (Industry, Innovation, and Infrastructure) and SDG 11 (Sustainable Cities and Communities), by introducing an innovative solution that enhances urban mobility and encourages the adoption of autonomous systems in public and private spaces. Its adaptable design allows it to be customized for various regions and applications around the world.	High
Societal	From a societal perspective, the system introduces the concept of autonomous transport to public institutions and recreational spaces, improving accessibility and operational efficiency. It reduces labor dependency by shifting roles from drivers to technicians, and it sparks community interest in robotics and AI technologies especially among students promoting innovation and technical engagement within Qatar. This fosters a culture of learning, development, and forward-thinking urban planning.	Medium
Environmental	In terms of environmental impact, the project supports clean-energy mobility by operating entirely on electric power. Unlike fuel-powered alternatives, this solution emits no pollutants, supports low-emission zones, and contributes to the overall goal of reducing the transportation sector's environmental footprint. Moreover, the components used such as the LiDAR and battery pose no significant health risks or environmental hazards, ensuring that safety is maintained not only for users but also for the ecosystem.	High
Economic	Economically, the solution presents a cost-effective and locally adaptable alternative to conventional transport systems. It can be assembled using components that are either available in Qatar or easily sourced, with the potential to stimulate local manufacturing and entrepreneurship. While the need	Medium

	for specialized components and technical expertise may present short-term barriers to commercial scalability, the long-term potential for job creation, innovation, and reduced operational costs makes the system a promising contributor to Qatar's economic diversification and smart mobility initiatives.	

9 Conclusion

The autonomous golf cart project successfully met most of its stated objectives, including the design, partially integration. The team built a prototype on a mini motorized vehicle platform, incorporating key systems such as LiDAR-based mapping, webcam was used ensuring basic vision capabilities and allowing full functional testing, ultrasonic obstacle detection, and real-time emergency stopping.

Subsystems including Jetson Orin NX integration, ROS 2-based sensor fusion, PWM motor control, and GUI-driven data collection were all validated through practical testing in a consistent outdoor environment on the Qatar University campus. While testing was limited to one environment due to access restrictions, it provided clear evidence of system stability and functional readiness.

The project's strengths lie in its modular architecture, its use of affordable yet powerful components, and its ability to execute core functions like obstacle detection, and safe stopping. The early use of a Processing GUI for low-level testing and dataset collection also offered a practical path for debugging and future development.

However, like any early-stage autonomous system, the solution still has shortcomings that will require further refinement particularly in the areas of path planning accuracy, sensor alignment, and response timing under diverse real-world conditions. Additionally, features like YOLOv5 object detection and full Nav2-based autonomous navigation needs to be fully implemented, they require further tuning and optimization to achieve production-level robustness.

The project also represents a novel step in Qatar's local development of autonomous vehicles, offering not only technical contributions but also educational value and long-term potential. It lays a strong foundation for future work in smart mobility, intelligent transport systems, and real-world autonomous deployments.

10 Future work

1. Transition to a Full-Sized Golf Cart:
    a. While the prototype was validated on a smaller vehicle, the original vision remains to implement this system on a real electric golf cart. This would enable further testing in real environments and demonstrate true operational viability.
2. Enhanced Object Detection and Avoidance:
    a. Future iterations will focus on completing the YOLOv5-based object detection pipeline and integrating it with dynamic obstacle avoidance logic. These features are critical for real-time decision-making in complex environments.
3. Advanced Path Planning and Navigation:

    a. Future work will include full autonomous route planning and path execution, with smooth re-routing in the presence of dynamic obstacles.

4. Improved System Calibration and Performance Tuning:
    a. Fine-tuning sensor positions, synchronizing data streams, and optimizing CPU/GPU usage on the Jetson will improve responsiveness and stability.
5. Power and Runtime Optimization:
    a. Battery performance will be enhanced through hardware upgrades and load balancing to meet or exceed the targeted 4-hour runtime.
6. Security and Data Handling Enhancements:
    a. Implementing encryption and secure communication protocols will help meet data security and privacy constraints.
7. Expanded Testing Conditions:
    a. Future testing will cover variable lighting, terrain, and pedestrian traffic to better simulate deployment conditions and improve reliability.
8. Reintegrating a depth-sensing camera (such as Intel RealSense) is planned to enhance object detection and depth-based navigation.

YOLOv5 Object Detection Integration

One of the key future enhancements involves implementing YOLOv5 (You Only Look Once, Version 5) for real-time object detection. YOLOv5 is a convolutional neural network architecture known for its speed and accuracy in identifying objects within RGB frames. The plan is to stream live camera data into a ROS 2 perception node, run inference using YOLOv5, and publish results to a /camera/detections topic. Each detection would include bounding boxes, object class, and confidence scores. These results can be used for obstacle labeling, navigation decisions, and even dynamic path replanning. The model will be pre-trained and fine-tuned using a custom dataset collected during Phase 1. Due to current limitations in sensor support (e.g., the Intel RealSense not functioning properly on Jetson), the integration is postponed but remains a priority for improving scene understanding in the next development cycle.

AI-Based Collision Detection and Avoidance

In future development, we plan to replace fixed-threshold emergency stop logic with a more advanced AI-based collision detection and avoidance system. Instead of relying solely on distance sensors, the upgraded system would analyze patterns in camera and LiDAR data using trained neural networks. These models would detect moving obstacles such as pedestrians, bicycles, or vehicles, predict their trajectories, and adjust the vehicle's speed or direction accordingly. Techniques such as bounding box tracking, optical flow, and spatiotemporal object prediction could be used to enhance reaction speed and reduce false positives. This would elevate the cart's perception capabilities, enabling smoother and safer autonomous navigation in dynamic environments.

11 Student reflections

Zabin’s Reflection

Working on the Jetson Orin as the central computational platform was a highly enriching experience. It provided me with the opportunity to configure and optimize a high-performance embedded system tailored to robotics applications. Through this project, I gained invaluable technical expertise, including setting up Ubuntu 22.04 and ROS 2 Humble, which are critical components for modern robotics systems. Additionally, I became proficient in middleware integration, utilizing ROS 2 to manage communication between nodes. My work also involved using advanced development tools such as Terminator for efficient multi-terminal management and Visual Studio Code for debugging and coding. These skills not only contributed to the project but also have immense value for my future career, especially in roles involving robotics and embedded systems.

Abdulaziz’s Reflection

Focusing on the RPLIDAR A3M1 allowed me to deepen my understanding of LiDAR systems and their integration within a robotics framework. This project taught me the critical role of precision in configuring and testing sensors to ensure functionality. I developed strong technical skills, including installing and configuring the rplidar_ros package, managing dependencies with tools like colcon, and using RViz2 to visualize real-time LiDAR data. Conducting range and accuracy tests further honed my abilities in testing and validation, ensuring that the LiDAR functioned as required. These experiences have significantly enhanced my skillset, preparing me for future efforts in robotics and autonomous systems.

On a personal level, this project helped me cultivate patience and analytical thinking, especially when troubleshooting LiDAR-related issues. It also emphasized the importance of effective communication and coordination within a team, which played a vital role in integrating the LiDAR successfully. Initially, I underestimated the time required for comprehensive testing and validation, which led to some stress. However, I learned to allocate time effectively and approach debugging methodically. Moving forward, I plan to carry this meticulous approach and attention to detail into future projects. Above all, this experience highlighted the value of collaboration and adaptability in achieving project goals and addressing challenges efficiently.

Abdulla’s reflection

In this project, I gained extensive knowledge in the field of software, particularly in understanding convolutional neural networks (CNNs) through the videos presented by our supervisor. I improved my skills in Python and developed a deeper understanding of microcontrollers. However, my most significant learning came from my work with hardware, as I was responsible for that aspect of the project. I learned about the essential components required for several systems, their functions, and how to prepare and program them for proper operation, including the steps necessary for fine-tuning. Additionally, I developed the ability to establish proper connections for the sprayer and gained several technical skills.

I also acquired problem-solving and diagnostic skills, especially when we encountered issues with the motors. We systematically tested each motor individually, ensured the current was reaching them, and resolved communication problems within the system. This taught me the importance of thoroughly checking all safety measures before engaging in hardware tasks for future projects. Moreover, I recognized the value of having a well-structured and clear plan to guide the project's progress effectively.

References

[1] FIFA, “Sustainability: Wider accessibility in Qatar,” *FIFA Publications*, 2022. Available: https://publications.fifa.com/en/final-sustainability-report/social-pillar/accessibility/wider-accessibility-in-qatar/. [Accessed: Aug. 25, 2024]

[2] B. W. Smith, "Human error as a cause of vehicle crashes," *Center for Internet and Society*, Dec. 18, 2013. [Online]. Available: https://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes/. [Accessed: Dec. 10, 2024].

[3] V. P. K. Sundram, N. Hashim, S. H. Shariff, A. Pujiati, and A. Ardiansari, "Sustainable transportation on university campus: A case at UiTM Selangor, Puncak Alam Campus, Malaysia and Universitas Negeri Semarang, Indonesia," *Asian Journal of University Education*, vol. 17, no. 2, pp. 262–272, Apr. 2021. [Online]. Available: https://files.eric.ed.gov/fulltext/EJ1304759.pdf. [Accessed: Dec. 10, 2024].

[4] Singh, S, “TRAFFIC SAFETY FACTS Crash.Stats,” National Highway Traffic Safety Administration, Mar. 2018.

[5] B. Vaidya and H. T. Mouftah, *Connected Autonomous Electric Vehicles as Enablers for Low-Carbon Future*. IntechOpen, 2019. Available: https://www.intechopen.com/chapters/65607. [Accessed: Aug. 25, 2024]

[6] United Nations, “The 17 Sustainable Development Goals,” *United Nations*, 2015. Available: https://sdgs.un.org/goals. [Accessed: Aug. 26, 2024]

[7] Maximizing the environmental benefits of autonomous vehicles, “Maximizing the environmental benefits of autonomous vehicles,” *University of Michigan News*, Feb. 15, 2018. Available: https://news.umich.edu/maximizing-the-environmental-benefits-of-autonomous-vehicles/. [Accessed: Aug. 26, 2024]

[8] Zion Market Research, “Autonomous Vehicle Market Size Report, Share, Trends, Growth, 2032,” *Zion Market Research*, 2024. Available: https://www.zionmarketresearch.com/report/autonomous-vehicle-market. [Accessed: Aug. 29, 2024]

[9] “Carteav - The Future of Autonomous Is Here,” *Carteav*, Nov. 14, 2024. Available: https://carteav.com/. [Accessed: Aug. 29, 2024]

[10] “T-Buggy® EV - Autonomous, Electric & Connected Golf Cart,” *T-Buggy*, Nov. 29, 2023. Available: https://tbuggy.com/. [Accessed: Aug. 29, 2024]

[11] Tesla, “Model Y | Tesla,” *Model Y*, 2019. Available: https://www.tesla.com/modely. [Accessed: Aug. 29, 2024]

[12] "GEM Electric Vehicles," *GEM - Electric Vehicles*. Available: https://www.gemcar.com/. [Accessed: Aug. 29, 2024]

[13] Purely Branded, "The Four Ps of Marketing," *Purely Branded*, 2017. Available: https://www.purelybranded.com/insights/the-four-ps-of-marketing/. [Accessed: Sep. 05, 2024]

[14] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International," *www.sae.org*, Apr. 30, 2021. Available: https://www.sae.org/standards/content/j3016_202104/. [Accessed: Aug. 20, 2024]

[15] M. Reke et al., "A Self-Driving Car Architecture in ROS2," *2020 International SAUPEC/RobMech/PRASA Conference*, Cape Town, South Africa, 2020, pp. 1-6, doi: 10.1109/SAUPEC/RobMech/PRASA48453.2020.9041020.

[16] M. Ahmed, R. Iqbal, S. Amin, O. Alhabshneh and A. Garba, "Autonomous Vehicle and its Adoption: Challenges, Opportunities, and Future Implications," *2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA)*, Karak, Jordan, 2022, pp. 1-6, doi: 10.1109/ETCEA57049.2022.10009804.

[17] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research", *Sensors*, vol. 19, no. 3, p. 648, 2019. doi: 10.3390/s19030648.

[18] H. Tremura and O. Toker, "Vehicle Level Software Design of the Florida Polytechnic Autonomous Golf-Cart," *SoutheastCon 2021*, Atlanta, GA, USA, 2021, pp. 1-4, doi: 10.1109/SoutheastCon45413.2021.9401891.

[19] M. -Q. Pham, H. -P. Ly, H. C. Quang, T. Nguyen-Van, D. T. Tung and M. Le-Hoang, "Transform an Electric Golf Cart into an Autonomous Vehicle," *2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, Phu Quoc, Vietnam, 2022, pp. 1-6, doi: 10.1109/MAPR56351.2022.9924975.

[20] D. Gaynor, T. Latham, I. Anderson, and C. Johnson, "Autonomous Golf Cart," *Proceedings of the 2013 ASEE North-Central Section Conference*, Ada, Ohio: Ohio Northern University, 2013.

[21] B. Sivakumar and N. T. P., "Design and Implementation of IoT-Based Golf Cart Autonomous Vehicles," *International Research Journal of Engineering and Technology (IRJET)*, vol. 8, no. 9, pp. 2081–2086, Sep. 2021.

[22] The article "A Comprehensive Review on Autonomous Vehicles: Challenges and Future Directions" by A. M. El-Sayed is published in the *Alexandria Engineering Journal*, Volume 63, Issue 1, 2024, on pages 1–15. The DOI for this article is 10.1016/j.aej.2024.01.001.

[23] E. Fayyad, J. Jaradat, and F. T. Ali, "Autonomous vehicles: Ethical dilemmas, social challenges, and technological opportunities," *Journal of Traffic and Transportation*

Engineering (English Edition), vol. 10, no. 1, pp. 1–13, 2023. doi: 10.1016/j.jtte.2022.03.007.

[24] F. Bonarini, G. Fontana, and D. Giglio, "A hierarchical framework for autonomous navigation in smart environments," *Robotics and Autonomous Systems*, vol. 157, pp. 104121, 2023. doi: 10.1016/j.robot.2022.104121.

[25] Driveblocks, "Fast Reaction Times: Why End-to-End Latency Matters in Autonomous Driving," Nov. 28, 2022. Available: https://www.driveblocks.ai/news/2022-11-28/fast-reaction-times-why-end-to-end-latency-matters-in-autonomous-driving. Accessed: Oct. 12, 2024.

[26] MIT News, "How fast humans react to car hazards," Aug. 7, 2019. Available: https://news.mit.edu/2019/how-fast-humans-react-car-hazards-0807. Accessed: Oct. 12, 2024.

[27] EcoTree Lithium, "Golf Cart Battery Lifespan," Sep. 2024. Available: https://ecotreelithium.co.uk/news/golf-cart-battery-lifespan/#:~:text=On%20average%2C%20a%20golf%20cart,terrain%2C%20and%20frequency%20of%20stops. Accessed: Sep. 25, 2024.

[28] S. Zang, M. Ding, D. Smith, P. Tyler, T. Rakotoarivelo and M. A. Kaafar, "The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car," in *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 103-111, June 2019, doi: 10.1109/MVT.2019.2892497.

[29] Yahboom, "Jetson Orin NX Study Resources," [Online]. Available: http://www.yahboom.net/study/Jetson-Orin-NX. [Accessed: 10-Nov-2024].

[30] Shanghai Slamtec Co., Ltd., "LD310 SLAMTEC RPLIDAR Datasheet A3M1," Version 1.9, Apr. 2021. [Online]. Available: https://www.slamtec.com/en. [Accessed: 10-Nov-2024].

[31] A. Haider et al., "A Methodology to Model the Rain and Fog Effect on the Performance of Automotive LiDAR Sensors," *Sensors*, vol. 23, no. 6891, Aug. 2023. [Online]. Available: https://doi.org/10.3390/s23156891

[32] Intel Corporation, "Intel® RealSense™ Depth Camera D435i," [Online]. Available: https://www.intelrealsense.com/depth-camera-d435i/. [Accessed: 28-Nov-2024].

[33] A. J. Smith, "Basics of Ultrasonic Sensors: Applications and Limitations," Sensors and Applications Journal, vol. 15, no. 3, pp. 50-57, 2023.

[34] H. Y. Chen, "Designing Reliable Ultrasonic Obstacle Detection Systems," IEEE Transactions on Instrumentation and Measurement, vol. 72, no. 8, pp. 1234–1245, Aug. 2023.

[35] J. L. Farrell, “The Global Positioning System & Inertial Navigation,” McGraw-Hill Professional, New York, 2022.

[36] J. Gao, Y. Xie, and J. Wang, “Enhancing GPS Accuracy in Urban Environments with Augmentation Systems,” IEEE Aerospace and Electronic Systems Magazine, vol. 39, no. 5, pp. 12–20, May 2023.

[37] Arduino, "UNO R3," *Arduino Documentation,* Dec. 6, 2024. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3/. [Accessed: Nov. 10, 2024].

[38] C. Smith, "8 Disadvantages of Arduino (Compared with Raspberry Pi)," *Chip Wired,* Dec. 18, 2019. [Online]. Available: https://chipwired.com/disadvantages-of-arduino/. [Accessed: Nov. 10, 2024].

[39] M. K. Kazimierczuk, "A Comparison of Popular Arduino Boards," *Maker Pro,* May 10, 2017. [Online]. Available: https://maker.pro/arduino/tutorial/a-comparison-of-popular-arduino-boards. [Accessed: Nov. 10, 2024].

[40] U.S. Department of Energy, "How Lithium-Ion Batteries Work," *Office of Energy Efficiency & Renewable Energy,* [Online]. Available: https://www.energy.gov/energysaver/articles/how-lithium-ion-batteries-work. [Accessed: Nov. 10, 2024].

[41] "Advantages and Disadvantages of Lithium-ion Batteries," *Science Struck,* [Online]. Available: https://sciencestruck.com/advantages-disadvantages-of-lithium-ion-batteries. [Accessed: Dec. 10, 2024].

[42] L. L. Christensen, M. R. Quigley, and D. M. Fernandez, "Robot Operating System 2: Design, Architecture, and Uses In The Wild," *arXiv preprint arXiv:2211.07752,* Nov. 2022. [Online]. Available: https://arxiv.org/abs/2211.07752. [Accessed: Nov. 5, 2024].

[43] A. Gutierrez and P. Gonzalez, "Impact of ROS 2 Node Composition in Robotic Systems," *arXiv preprint arXiv:2305.09933,* May 2023. [Online]. Available: https://arxiv.org/abs/2305.09933. [Accessed: Nov. 5, 2024].

[44] J. Smith, A. Brown, and M. Johnson, "Python in Robotics and Mechatronics Education," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA),* Stockholm, Sweden, May 2016, pp. 1234-1240. [Online]. Available: https://ieeexplore.ieee.org/document/7547108. [Accessed: Nov. 5, 2024].

[45] A. Nakamura, "PythonRobotics: A Python Code Collection of Robotics Algorithms," *arXiv preprint arXiv:1808.10703,* Aug. 2018. [Online]. Available: https://arxiv.org/pdf/1808.10703. [Accessed: Nov. 5, 2024].

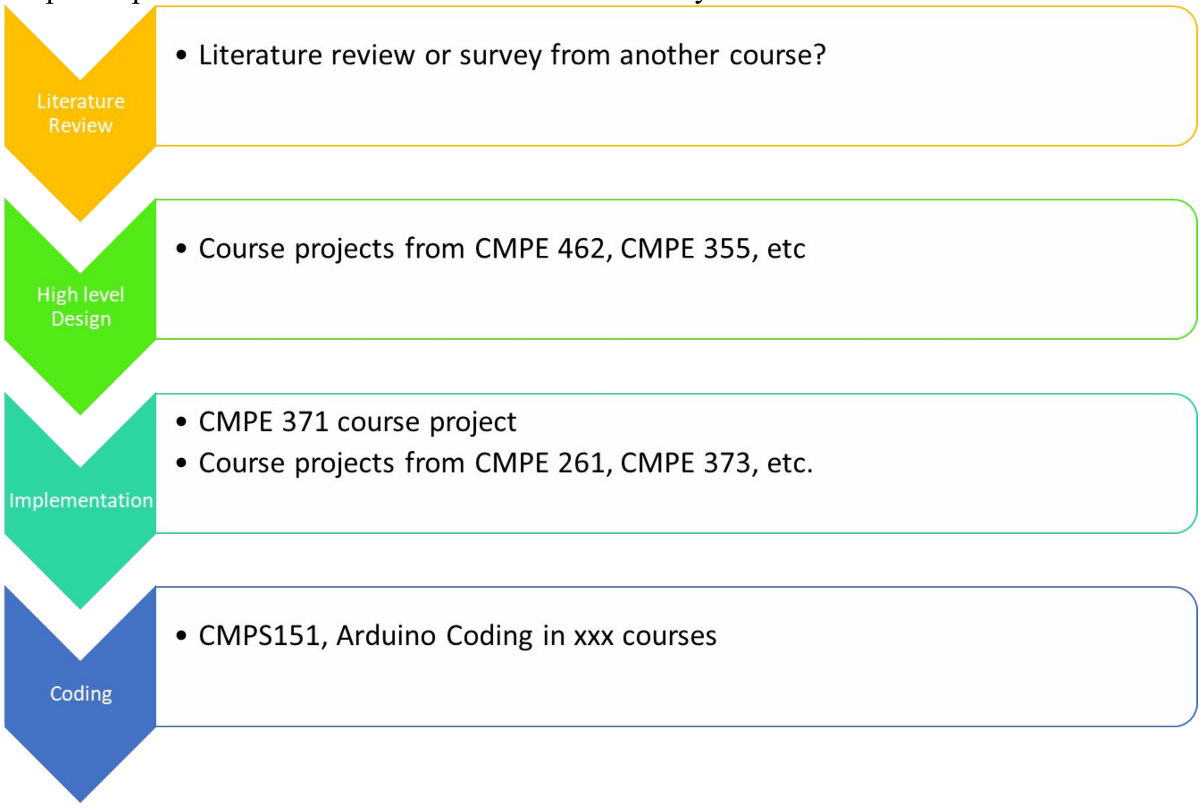
[46] R. Suarez Fernandez, J. L. Sanchez-Lopez, and P. Campoy, "Aerostack2: A Software Framework for Developing Multi-robot Aerial Systems," ar*Xiv preprint*

arXiv:2303.18237, Mar. 2023. [Online]. Available: https://arxiv.org/abs/2303.18237. [Accessed: Nov. 5, 2024].

[47] B. Sundaralingam, A. Jain, and A. K. Gupta, "PyRobot: An Open-source Robotics Framework for Research and Benchmarking," *arXiv preprint arXiv:1906.08236,* Jun. 2019. [Online]. Available: https://arxiv.org/abs/1906.08236. [Accessed: Nov. 5, 2024].

Appendix A – Culmination of Design Experiences

< **ToDo:**
Consider this as a backward look at your design through the lenses of the courses that you took during your studies at the CSE department. The culmination of design experience can contain a figure with a diagram/map showing examples of design activities that the students have gone through over the year. Students can also mention the relation of senior projects with the earlier courses that helped them to achieve in a better way their project goals. A sample map is shown below and can be modified to any level of details as needed:


Literature Review
• Literature review or survey from another course?
High level Design
• Course projects from CMPE 462, CMPE 355, etc
Implementation
• CMPE 371 course project
• Course projects from CMPE 261, CMPE 373, etc.
Coding
• CMPS151, Arduino Coding in xxx courses

/>

Other Appendices

Installing ROS 2 Humble

ROS 2 Humble was installed on the Jetson Orin following the official documentation. Below is a summary of the steps taken:

1. Add ROS 2 Repository:

```
$ sudo apt install software-properties-common
$ sudo add-apt-repository universe
$ sudo apt update && sudo apt install curl -y
$ sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo
$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
$ sudo apt update
$ sudo apt upgrade
```

2. Install ROS 2 Humble:

```
$ sudo apt install ros-humble-desktop
```

3. Install ROS Development Tools:

```
$ sudo apt install ros-dev-tools
```

4. Source ROS 2 Setup Script: to make ROS2 commands available in the terminal:

```
$ source /opt/ros/humble/setup.bash
```

5. Automate setup by adding the following line to the .bashrc file:

```
$ gedit ~/.bashrc
```

Add: source /opt/ros/humble/setup.bash

6. Verification:

```
$ ros2 run demo_nodes_cpp talker
$ ros2 run demo_nodes_py listener
```

Installing rplidar_ros Package

To enable LiDAR functionality, the rplidar_ros package was installed and configured:

1. Create a ROS 2 Workspace:

```
$ mkdir -p ~/ros2_ws/src
$ cd ~/ros2_ws/src
```

2. Clone the rplidar_ros Package:

```
$ git clone -b ros2 https://github.com/Slamtec/rplidar_ros.git
```

3. Build the rplidar_ros Package:

```
$ cd ~/ros2_ws/
$ source /opt/ros/humble/setup.bash
$ sudo apt install python3-colcon-common-extensions
$ colcon build --symlink-install
```

4. Set Up the Workspace Environment:

```
$ source ./install/setup.bash
$ echo 'source ~/ros2_ws/install/setup.bash' >> ~/.bashrc
$ source ~/.bashrc
```

5. Configure LiDAR Permissions:

```
$ sudo chmod 666 /dev/ttyUSB0
```

6. Verification:

```
$ ros2 launch rplidar_ros view_rplidar_a3_launch.py
```

Collecting data code:

```
import pyrealsense2 as rs
import numpy as np
import cv2
import os
import pandas as pd
import time


# Create directory to save data
save_dir = "collected_data"
os.makedirs(save_dir, exist_ok=True)


# Initialize RealSense pipeline
pipeline = rs.pipeline()
config = rs.config()


# Enable color, depth, gyro, and accel streams
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.gyro)
config.enable_stream(rs.stream.accel)

# Start streaming
pipeline.start(config)

# Align depth to color
align_to = rs.stream.color
align = rs.align(align_to)

frame_count = 0
data_log = []  # Store all data rows here

try:
    while True:
        # Wait for frames
        frames = pipeline.wait_for_frames()
        frames = align.process(frames)

        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()
        gyro_frame = frames.first_or_default(rs.stream.gyro)
        accel_frame = frames.first_or_default(rs.stream.accel)

        if not depth_frame or not color_frame or not gyro_frame or not accel_frame:
            continue

        # Convert frames to numpy arrays
        depth_image = np.asanyarray(depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        # Collect depth data at center pixel
        height, width = depth_image.shape
        center_x, center_y = width // 2, height // 2
        distance_mm = depth_frame.get_distance(center_x, center_y) * 1000  # in mm

        # IMU data
        gyro_data = gyro_frame.as_motion_frame().get_motion_data()
        accel_data = accel_frame.as_motion_frame().get_motion_data()

        # Save images
        color_filename = os.path.join(save_dir, f"color_{frame_count}.jpg")
        depth_filename = os.path.join(save_dir, f"depth_{frame_count}.png")
        cv2.imwrite(color_filename, color_image)
        cv2.imwrite(depth_filename, depth_image)

        # Draw distance text
        cv2.putText(color_image, f"Distance: {distance_mm:.2f} mm", (10, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
        cv2.imshow("RealSense D435i - Color Image", color_image)

        # Save to data log
        data_log.append([
            frame_count, center_x, center_y, distance_mm,
            accel_data.x, accel_data.y, accel_data.z,
            gyro_data.x, gyro_data.y, gyro_data.z
        ])

        print(f"Frame {frame_count} | Distance: {distance_mm:.2f} mm | "
              f"Accel: [{accel_data.x:.2f}, {accel_data.y:.2f}, {accel_data.z:.2f}] | "
              f"Gyro: [{gyro_data.x:.2f}, {gyro_data.y:.2f}, {gyro_data.z:.2f}]")

        frame_count += 1

        # Exit with 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        # Wait for 0.2s
        time.sleep(0.2)

finally:
    pipeline.stop()
    cv2.destroyAllWindows()

    # Save all collected data to CSV
    df = pd.DataFrame(data_log, columns=[
        "Frame", "Center_X", "Center_Y", "Distance_mm",
        "Accel_X", "Accel_Y", "Accel_Z",
        "Gyro_X", "Gyro_Y", "Gyro_Z"
    ])
    df.to_csv(os.path.join(save_dir, "realsense_data.csv"), index=False)

    print(f"\n✅ All data saved to: {save_dir}")
```

camera_veiw.lunch:

```
#!/usr/bin/env python3

from launch import LaunchDescription
from launch_ros.actions import Node
from launch.substitutions import LaunchConfiguration
from launch.actions import DeclareLaunchArgument, ExecuteProcess
from ament_index_python.packages import get_package_share_directory
import os

def generate_launch_description():
    # Declare launch arguments
    camera_id = DeclareLaunchArgument(
        'camera_id',
        default_value='1',  # Using /dev/video1
        description='Camera device ID'
    )

    # Camera node
    camera_node = Node(
        package='golf_cart_perception',
        executable='camera_node',
        name='camera_node',
        parameters=[{
            'camera_id': LaunchConfiguration('camera_id'),
            'frame_width': 640,
            'frame_height': 480,
            'frame_rate': 15.0,
            'confidence_threshold': 0.5
        }],
        output='screen'
    )

    # Static transform for camera
    tf_camera = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        name='tf_camera',
        arguments=['0.3', '0', '0.35', '0', '0', '0', 'base_link', 'camera_link']
    )

    # rqt_image_view
    rqt_image_view = ExecuteProcess(
        cmd=['ros2', 'run', 'rqt_image_view', 'rqt_image_view'],
        output='screen'
    )

    return LaunchDescription([
        camera_id,
        camera_node,
        tf_camera,
        rqt_image_view
    ])
```

test_lidar.lunch:

```
#!/usr/bin/env python3

from launch import LaunchDescription
from launch_ros.actions import Node

from launch.substitutions import LaunchConfiguration

from launch.actions import DeclareLaunchArgument

from ament_index_python.packages import get_package_share_directory

import os

def generate_launch_description():

# Declare launch arguments

serial_port = DeclareLaunchArgument(

'serial_port',

default_value='/dev/ttyUSB0',

description='RPLIDAR serial port'

)

baud_rate = DeclareLaunchArgument(

'baud_rate',

default_value='256000',

description='RPLIDAR baud rate'

)

# RPLIDAR node

rplidar_node = Node(

package='rplidar_ros',

executable='rplidar_node',

name='rplidar_node',

parameters=[{

'serial_port': LaunchConfiguration('serial_port'),

'serial_baudrate': LaunchConfiguration('baud_rate'),
'frame_id': 'laser',
'inverted': False,
'angle_compensate': True,
'scan_mode': 'Sensitivity'
}],
output='screen'
)
# Static transform for LiDAR
tf_lidar = Node(
package='tf2_ros',
executable='static_transform_publisher',
name='tf_lidar',
arguments=['0.25', '0', '0.4', '0', '0', '0', 'base_link', 'laser']
)
# RViz2 node for visualization
rviz_config = os.path.join(
get_package_share_directory('golf_cart_bringup'),
'rviz',
'test_sensors.rviz'
)
rviz_node = Node(
package='rviz2',
executable='rviz2',
name='rviz2',
arguments=['-d', rviz_config],
output='screen'
)
return LaunchDescription([
serial_port,
baud_rate,
rplidar_node,
tf_lidar,
rviz_node
])
```

emergency_stop_node.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from std_msgs.msg import String, Bool
from geometry_msgs.msg import Twist
import serial
import time
import threading
class EmergencyStopNode(Node):
"""
Emergency Stop Node for autonomous vehicle
This node monitors ultrasonic sensor data via serial communication
and triggers emergency stop when obstacles are detected within threshold
```

```
"""
def __init__(self):
super().__init__('emergency_stop_node')
# Parameters
self.declare_parameter('serial_port', '/dev/ttyACM1')
self.declare_parameter('baud_rate', 9600)
self.declare_parameter('stop_duration', 5.0) # seconds
# Get parameters
self.serial_port = self.get_parameter('serial_port').value
self.baud_rate = self.get_parameter('baud_rate').value
self.stop_duration = self.get_parameter('stop_duration').value
# State variables
self.emergency_active = False
self.stop_timer = None
# Publishers
self.cmd_vel_pub = self.create_publisher(Twist, '/cmd_vel', 10)
self.emergency_status_pub = self.create_publisher(Bool, '/emergency_status', 10)
# Subscribers
self.cmd_vel_sub = self.create_subscription(
Twist,
'/cmd_vel_input', # Original commands before intervention
self.cmd_vel_callback,
10)
# Set up serial connection
try:
```

```
self.ser = serial.Serial(
port=self.serial_port,
baudrate=self.baud_rate,
timeout=1.0
)
self.get_logger().info(f"Connected to Arduino on {self.serial_port}")
# Start serial monitoring thread
self.serial_thread = threading.Thread(target=self.monitor_serial)
self.serial_thread.daemon = True
self.serial_thread.start()
except serial.SerialException as e:
self.get_logger().error(f"Failed to connect to Arduino: {str(e)}")
self.ser = None
# Timer for status updates
self.status_timer = self.create_timer(1.0, self.publish_status)
self.get_logger().info("Emergency Stop Node initialized")
def cmd_vel_callback(self, msg):
"""Process incoming velocity commands and block them during emergency"""
if not self.emergency_active:
# Pass through command if no emergency
self.cmd_vel_pub.publish(msg)
else:
# Drop command during emergency
self.get_logger().debug("Blocking movement command during emergency stop")
def monitor_serial(self):
```

```
"""Monitor serial port for emergency signals from Arduino"""
while rclpy.ok() and self.ser is not None:
try:
if self.ser.in_waiting > 0:
data = self.ser.readline().decode('utf-8').strip()
if data == 's' or data == 'S': # Stop signal
self.trigger_emergency_stop()
self.get_logger().debug(f"Serial data: {data}")
except Exception as e:
self.get_logger().error(f"Serial error: {str(e)}")
time.sleep(1.0)
def trigger_emergency_stop(self):
"""Activate emergency stop procedure"""
if not self.emergency_active:
self.get_logger().warn("EMERGENCY STOP TRIGGERED!")
self.emergency_active = True
# Publish zero velocity immediately
stop_cmd = Twist()
self.cmd_vel_pub.publish(stop_cmd)
# Publish emergency status
status_msg = Bool()
status_msg.data = True
self.emergency_status_pub.publish(status_msg)
# Set timer to release emergency after duration
if self.stop_timer is not None:
```

```
self.stop_timer.cancel()
self.stop_timer = self.create_timer(
self.stop_duration,
self.release_emergency_stop
)
def release_emergency_stop(self):
"""Release emergency stop after timeout"""
self.emergency_active = False
self.get_logger().info("Emergency stop released")
# Publish emergency clear status
status_msg = Bool()
status_msg.data = False
self.emergency_status_pub.publish(status_msg)
# Cancel timer
if self.stop_timer is not None:
self.stop_timer.cancel()
self.stop_timer = None
def publish_status(self):
"""Publish current emergency status periodically"""
status_msg = Bool()
status_msg.data = self.emergency_active
self.emergency_status_pub.publish(status_msg)
def cleanup(self):
"""Clean up resources"""
if self.ser is not None:
```

```
self.ser.close()
self.get_logger().info("Serial port closed")
def main(args=None):
rclpy.init(args=args)
node = EmergencyStopNode()
try:
rclpy.spin(node)
except KeyboardInterrupt:
pass
finally:
node.cleanup()
node.destroy_node()
rclpy.shutdown()
if __name__ == '__main__':
main()
```

motor_control_node.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from std_msgs.msg import Bool, Float32
from sensor_msgs.msg import Range
import serial
import time
class MotorControlNode(Node):
"""
Motor Control Node for the autonomous golf cart
Controls the motors based on ultrasonic sensor data and emergency status.
Implements safety features and motor speed control.
"""
def __init__(self):
super().__init__('motor_control_node')
# Declare parameters
self.declare_parameter('serial_port', '/dev/ttyACM2') # Motor control Arduino
self.declare_parameter('baud_rate', 9600)
self.declare_parameter('drive_speed', 80) # Default drive speed (0-255)
self.declare_parameter('steer_speed', 120) # Default steering speed (0-255)
# Get parameters
self.serial_port = self.get_parameter('serial_port').value
self.baud_rate = self.get_parameter('baud_rate').value
self.drive_speed = self.get_parameter('drive_speed').value
self.steer_speed = self.get_parameter('steer_speed').value
# Motor state
self.current_command = "S"
self.emergency_active = False
self.is_moving = False
self.is_steering = False
# Create subscribers
```

```
self.emergency_sub = self.create_subscription(
Bool,
'/emergency_status',
self.emergency_callback,
10)
self.speed_sub = self.create_subscription(
Float32,
'/target_speed',
self.speed_callback,
10)
self.steer_sub = self.create_subscription(
Float32,
'/target_steer',
self.steer_callback,
10)
# Create serial connection
try:
self.serial = serial.Serial(
port=self.serial_port,
baudrate=self.baud_rate,
timeout=1.0
)
self.get_logger().info(f"Connected to motor control Arduino on {self.serial_port}")
# Wait for Arduino to initialize
time.sleep(2)
```

```
except serial.SerialException as e:
self.get_logger().error(f"Failed to connect to Arduino: {str(e)}")
self.serial = None
# Create control timer
self.control_timer = self.create_timer(0.1, self.control_loop)
self.get_logger().info("Motor Control Node initialized")
def emergency_callback(self, msg):
"""Handle emergency status updates"""
self.emergency_active = msg.data
if self.emergency_active:
self.send_command("S")
self.get_logger().warn("Emergency stop - motors disabled")
def speed_callback(self, msg):
"""Handle target speed updates"""
if not self.emergency_active:
# Convert speed (0-1) to command
speed = msg.data
if speed > 0.1: # Forward
self.is_moving = True
if self.is_steering:
self.send_command("FL" if self.current_steer < 0 else "FR")
else:
self.send_command("F")
elif speed < -0.1: # Backward
self.is_moving = True
```

```
if self.is_steering:
self.send_command("BL" if self.current_steer < 0 else "BR")
else:
self.send_command("B")
else: # Stop
self.is_moving = False
if not self.is_steering:
self.send_command("S")
def steer_callback(self, msg):
"""Handle steering updates"""
if not self.emergency_active:
steer = msg.data
if abs(steer) > 0.1: # Significant steering input
self.is_steering = True
self.current_steer = steer
if self.is_moving:
if steer < 0: # Left
self.send_command("FL" if self.current_speed > 0 else "BL")
else: # Right
self.send_command("FR" if self.current_speed > 0 else "BR")
else:
self.send_command("L" if steer < 0 else "R")
else: # No steering
self.is_steering = False
if self.is_moving:
```

```
self.send_command("F" if self.current_speed > 0 else "B")
else:
self.send_command("S")
def send_command(self, command):
"""Send command to Arduino"""
if self.serial is not None and command != self.current_command:
try:
self.serial.write(f"{command}\n".encode())
self.current_command = command
self.get_logger().info(f"Sent command: {command}")
# Read response
response = self.serial.readline().decode().strip()
if response:
self.get_logger().info(f"Arduino response: {response}")
except serial.SerialException as e:
self.get_logger().error(f"Failed to send command: {str(e)}")
def control_loop(self):
"""Main control loop"""
if self.emergency_active:
self.send_command("S")
def cleanup(self):
"""Clean up resources"""
if self.serial is not None:
# Send stop command before closing
self.send_command("S")
```

```
time.sleep(0.1) # Give time for command to be sent
self.serial.close()
self.get_logger().info("Serial port closed")
def main(args=None):
rclpy.init(args=args)
node = MotorControlNode()
try:
rclpy.spin(node)
except KeyboardInterrupt:
pass
finally:
node.cleanup()
node.destroy_node()
rclpy.shutdown()
if __name__ == '__main__':
main()
```

```
// Front Ultrasonic Sensors (Arduino R3)
#define TRIG_FRONT_LEFT 8
#define ECHO_FRONT_LEFT 11
#define TRIG_FRONT_CENTER 9
#define ECHO_FRONT_CENTER 12
#define TRIG_FRONT_RIGHT 10
```

```
#define ECHO_FRONT_RIGHT 13

// Constants
const int MAX_DISTANCE = 400;  // Maximum distance in cm
const int MIN_DISTANCE = 2;    // Minimum distance in cm
const int EMERGENCY_DISTANCE = 30;  // Emergency stop distance in cm
const int WARNING_DISTANCE = 50;    // Warning distance in cm

// Variables
long duration;
int distance_fl, distance_fc, distance_fr;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize ultrasonic sensor pins
  pinMode(TRIG_FRONT_LEFT, OUTPUT);
  pinMode(ECHO_FRONT_LEFT, INPUT);
  pinMode(TRIG_FRONT_CENTER, OUTPUT);
  pinMode(ECHO_FRONT_CENTER, INPUT);
  pinMode(TRIG_FRONT_RIGHT, OUTPUT);
  pinMode(ECHO_FRONT_RIGHT, INPUT);

  // Ensure all trigger pins are LOW
  digitalWrite(TRIG_FRONT_LEFT, LOW);
  digitalWrite(TRIG_FRONT_CENTER, LOW);
  digitalWrite(TRIG_FRONT_RIGHT, LOW);

  delay(100);
  Serial.println("Front Ultrasonic System Ready");
}

void loop() {
  // Read all sensors
  distance_fl = readDistance(TRIG_FRONT_LEFT, ECHO_FRONT_LEFT);
  delay(10);  // Small delay between readings
  distance_fc = readDistance(TRIG_FRONT_CENTER, ECHO_FRONT_CENTER);
  delay(10);
  distance_fr = readDistance(TRIG_FRONT_RIGHT, ECHO_FRONT_RIGHT);

  // Check for emergency conditions
  if (distance_fl < EMERGENCY_DISTANCE ||
      distance_fc < EMERGENCY_DISTANCE ||
      distance_fr < EMERGENCY_DISTANCE) {
    Serial.println("S");  // Send emergency stop signal
  }

  // Send sensor data in format: FL:distance,FC:distance,FR:distance
  Serial.print("FL:");
  Serial.print(distance_fl);

  Serial.print(",FC:");

  Serial.print(distance_fc);

  Serial.print(",FR:");

  Serial.println(distance_fr);


  delay(200);  // Wait before next reading

}


int readDistance(int trigPin, int echoPin) {

  // Clear the trigger pin

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);


  // Set the trigger pin HIGH for 10 microseconds

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);


  // Read the echo pin and calculate distance

  duration = pulseIn(echoPin, HIGH);

  int distance = duration * 0.034 / 2;  // Convert to cm


  // Validate distance

  if (distance < MIN_DISTANCE || distance > MAX_DISTANCE) {
    return MAX_DISTANCE;  // Return max distance if reading is invalid
  }

  return distance;
}
```