Small-Scale Autonomous Racing Vehicles

Sponsored by Dr. Yaser P Fallah's Research Group (CAVREL)

Group 32

Israel Charles	Casey Jack	Asa Daboh	Owen Burns	Tevin Mukudi
Computer Engineering	Electrical Engineering	Computer Engineering	Computer Science	Mechanical Engineering

Mentor & Sponsor

Dr. Yaser P Fallah

ECE/CS Faculty

Reviewers

Dr. Chinwendu Enyioha	Dr. Truong Nghiem	Dr. Yaser P Fallah
ECE/CS Faculty	*ECE/CS Faculty*	*ECE/CS Faculty*

Table of Contents

List of Figures

List of Tables

List of Equations

1 - Executive Summary

1.1 - Who we are

We are a multidisciplinary team of five engineering students united by a shared passion for robotics, automation, and intelligent systems. Our diverse backgrounds help us approach challenges from multiple technical perspectives. We are consisted of:

- 2 Computer Engineering students (Israel and Asa)
- 1 Computer Science student (Owen)
- 1 Electrical Engineering student (Casey)
- 1 Mechanical Engineering student (Tevin)

Many of us have chosen to minor in Intelligent Robotic Systems, equipping us with skills in robotics, decision-making algorithms, and how machines interact with their environment. Some of us have hands-on experience building robots, designing control systems, and applying machine learning in real-world scenarios.

1.2 - Our Why

In addition to fulfilling our degree's capstone requirement, this project reflects our personal interests and professional goals. Our motivation comes from several factors:

- **Cutting-Edge Challenge:** Autonomous racing brings together complex topics like computer vision, real-time decision-making, and control systems. The fast-paced race environment demands precise execution and constant optimization, pushing us to innovate and refine our skills.
- **Passion for Robotics:** This project allows us to combine our expertise in mechanical design, control systems, and intelligent sensing to build a vehicle that can navigate, make split-second decisions, and race independently. It's a perfect way to challenge what we've learned and grow as future engineers.
- **Real-World Applications:** Autonomous vehicles are transforming industries, from transportation to robotics. Through this project, we are gaining skills and familiarization of technologies like LIDAR and SLAM that are essential for the future of automation.
- **Interdisciplinary Collaboration:** Working across disciplines gives us valuable experience in teamwork and problem-solving, preparing us for careers in the robotics industry. Combining electrical, mechanical, and software expertise makes our project stronger and teaches us how to collaborate on real-world engineering problems.
- **Community Contribution:** A key goal for us is to share our knowledge. We will provide easy-to-follow, step-by-step instructions so others can replicate our project,

helping bridge the knowledge gap around autonomous vehicles.

1.3 - Project Overview

Our primary goal is to build two small-scale autonomous race cars capable of competing on unknown racetracks. One will have a higher physical speed limit, allowing us to test the efficiency of the autonomous algorithms.

Each car will navigate autonomously, making real-time decisions while competing with other cars. Throughout the design, implementation, and optimization of the vehicles, we focused on two key priorities:

- **Maximum Speed:** Push the vehicle to perform at the highest speed possible.
- **Robust Obstacle Avoidance:** Maintain safe and efficient navigation by avoiding obstacles in real time.

The project will also serve as a foundation for future autonomous racing efforts at UCF.

1.4 - Legacy

We aim to leave a lasting impact at UCF by organizing workshops in collaboration with IEEE and ACM. These workshops teach others about autonomous vehicles, encouraging students to get involved in robotics and automation.

This project is not just about building cars, it's about creating opportunities, developing practical solutions, contributing to the advancement of robotics, and the legacy we can leave behind for others to build on.

1.5 - Main Technologies

To achieve our goals and objectives, we carefully researched and implemented the most suitable technologies. The key technologies utilized in our project are listed below:

1.5.1 - Hardware Components

- **Vehicle Chassis:** Traxxas Slash 4X4 Ultimate Edition
- **Electronic Speed Controller (ESC):** Traxxas Waterproof Brushless ESC or VESK 6 MKV
- **Power Board:** Custom-designed board using ESP32
- **System Status Indicator Module:** Custom-built using ESP32 for real-time system monitoring
- **Main Computing Unit:** NVIDIA Jetson Xavier NX for high-performance computing
- **Sensors:**

- **LiDAR:** Hokuyo UST-10LX Scanning Laser Rangefinder
- **IMU:** VESK 6 MKV or Holybro Pixhawk 6c
- **Camera:** Intel RealSense D345i

1.5.2 - Software Components

- **Simulation Tools:**
  - **Gazebo Simulator:** 3D physics-based simulation for realistic and real-time simulation
  - **F1Tenth Gym ROS Simulator:** 2D lightweight simulation environment for fast and easy and simulation setup
- **Middleware Framework:**
  - **ROS 2 Foxy:** For physical unit and 2D lightweight simulations
  - **ROS Melodic:** For 3D physics-based simulation
- **Programming Languages:**
  - **Python:** High-level software development for the main computing unit
  - **C++/C:** Low-level programming for the power board and system status indicator
- **Algorithms:**
  - **Computer Vision:** OpenCV for image processing and object detection
  - **Mapping and Localization:** SLAM (Simultaneous Localization and Mapping)
  - **Control Algorithms:** Model Predictive Control (MPC)
  - **Path Planning:** Rapidly Exploring Random Trees (RRT)

1.6 - Project Scope

Our project is divided into four phases, each building on the previous one to ensure a structured and efficient development process.

1.6.1 - Research and Planning

- Conduct technology comparisons and literature reviews on autonomous racing algorithms.
- Analyze and define system requirements and constraints.

1.6.2 - Hardware Setup

- Obtain and test individual hardware components to ensure compatibility.
- Assemble the vehicle chassis with sensors and the computing unit.

1.6.3 - Algorithm Development

- Set up the ROS (Robot Operating System) environment and simulation tools.
- Implement and optimize key algorithms:
    - SLAM (Simultaneous Localization and Mapping)
    - State lattice planner
    - Obstacle avoidance strategies for safe navigation
    - Conduct unit tests in simulation environments and fine-tune algorithms.

1.6.4 - Real-World Testing and Optimization

- Set up physical racetracks to evaluate performance in real-world conditions.
- Test, validate, and refine algorithms based on physical performance.

1.7 - Deliverables

- **Comprehensive Documentation:** A detailed report covering system design, algorithms, simulations, and testing results.
- **Physical Autonomous Racing Vehicles:** Two fully functional small-scale race cars capable of demonstrating reliable performance.
- **Presentation and Live Demonstration:** Showcase the vehicles racing autonomously, with reliable obstacle avoidance and real-time decision-making.

2 - Project description

2.1 - Background

We are a multidisciplinary team of five engineering students with a shared passion for robotics, automation, and intelligent systems. Our diverse backgrounds across several engineering disciplines allow us to approach challenges from different technical perspectives. Here's a breakdown of our team's composition:

- **Two Computer Engineering Students (Israel and Asa):** Our computer engineers specialize in hardware-software integration, embedded systems, and real-time computing. They have experience in developing low-level software, working with sensors, and using microcontrollers. They also have some experience in designing application-specific printed circuit boards. With experience on both the software

and hardware side of things, they make the perfect bridge for interfacing the software that will run the project and the hardware that the project will be run on. Their background is crucial in designing and implementing some of the vehicles' systems, interfacing data acquisition from some sensors, and ensuring real-time processing necessary for autonomous navigation.

- **One Computer Science Student (Owen):** Our computer science student has a passion for artificial intelligence, machine learning, and computer vision. They are proficient in algorithms, data structures, and technical expertise that allow vehicles to make real-time intelligent decisions. They have also worked directly with small-scale autonomous vehicles, having published a research paper on autonomous vehicle platooning at the Vehicular Technology Conference (VTC 2024 - Fall). This student will focus on developing the perception, planning, and control components of the systems, enabling the vehicles to recognize objects, track lanes, and navigate independently at high speeds.

- **One Electrical Engineering Student (Casey):** Our electrical engineering student has a strong focus on circuit design, power systems, and signal processing. He brings expertise in sensor integration and power management, which are essential for the different components of the vehicles to receive power. This student will ensure that the systems remain efficient in terms of power consumption.

- **One Mechanical Engineering Student (Tevin):** Our mechanical engineering student brings expertise in dynamics, control theory, and mechanical design. He is responsible for designing and optimizing the physical chassis of the vehicles, ensuring they remain lightweight, have good acceleration while also handling well while racing through a track. His understanding of the vehicles' mechanical behavior ensures that the autonomous control systems can interact effectively, efficiently, and with ease with the physical world.

As robotics and automation enthusiasts, the majority of us have chosen to minor in **Intelligent Robotic Systems**. This has equipped us with a deep understanding of how robots interact with their environment, make decisions based on sensor input, and perform complex tasks autonomously. Some of us have gained hands-on experience in building robots, designing control algorithms, and integrating machine learning into real-world systems.

2.2 - Motivation

Our motivation for choosing the Small-Scale Autonomous Racing Vehicles project comes from several factors:

- **Cutting-edge Challenge:** Autonomous racing combines various complex domains such as computer vision, real-time decision-making, and control systems. The fast-paced environment of a race demands the highest level of precision and optimization, which pushes us to challenge our technical skills and innovate. Autonomous vehicles are a rapidly evolving field with applications in industries like automotive engineering, robotics, and Artificial Intelligence (AI) driven

transportation. Working on this project positions us at the forefront of this technological frontier.

- **Passion for Robotics:** As students passionate about robotics, we see this project as an opportunity to bring together our knowledge in intelligent systems, sensing, control, and mechanical design. Building an autonomous vehicle that can navigate, make split-second decisions, and race on its own is the ultimate test of our combined knowledge and skills. This project allows us to apply what we've learned throughout our studies in a highly tangible and exciting way.

- **Real-World Applications:** Autonomous vehicles are shaping the future of transportation and robotics. By working on this project, we are contributing to the development of technologies that have the potential to revolutionize industries. Whether it's improving self-driving cars or enhancing robotics in industries such as logistics, manufacturing, or space exploration, the skills and experience we gain from this project are directly applicable to solving real-world problems.

- **Interdisciplinary Collaboration and Professional Growth:** This project allows us to collaborate across disciplines, combining expertise from electrical, mechanical, computer engineering, and computer science. Practicing this kind of complex collaboration now, especially with complex and relevant technologies and techniques like Light Detection and Ranging (LIDAR) and Simultaneous Localization and Mapping (SLAM), puts us a step ahead in preparing to work in the robotics industry. Performing well in this project and potential competitions that we plan on participating in, such as the F1Tenth Competition, also allows us to showcase our skills to potential employers and graduate schools.

- **Community Contribution:** One of our goals is to make this project available for anyone to use and learn from. We want to contribute to bridging the gap of knowledge when it comes to autonomous vehicles. We aim to create simplified detailed step-by-step instructions on how to replicate our final product. This way, anyone with basic robotic knowledge can learn and implement our project. One of our objectives to accomplish that goal is to work with the Institute of Electrical and Electronics Engineers (IEEE) and Association of Computing Machinery (ACM) chapters here at the University of Central Florida (UCF), along with some other chapters, to ensure autonomous vehicle racing maintains a presence at UCF.

2.3 - Related Work

For our Small-Scale Autonomous Racing Vehicle project, some notable projects, communities, and existing products can serve as inspiration or reference points. These projects and existing products showcase a range of technologies in autonomous driving, artificial intelligence, machine learning, robotics, and control systems. Below are some relevant examples:

- **Indy Autonomous Challenge**
    - **Overview:** The Indy Autonomous Challenge is a global competition where

university teams design autonomous race cars to compete on full-scale race tracks [1].

- **Key Technologies:** Autonomous driving relies on high-end sensors, such as Light Detection and Ranging (LIDAR), Radio Detection And Ranging (RADAR), and cameras. Autonomous driving also relies on deep learning for real-time perception, and control systems optimized for high-speed maneuvers.

- **Relevance:** Although full-sized, this project demonstrates the cutting-edge in autonomous vehicle racing, including the use of real-time Artificial Intelligence (AI) and control under high-speed conditions. Some competitors at the Indy Autonomous Challenge had started out with small-scale racing autonomous vehicles. Therefore, we believe that the techniques used there can be scaled down to our project.

- **Gymnasium (OpenAI Gym) and Reinforcement Learning Projects**

  - **Overview:** Gymnasium (formerly OpenAI Gym) is a toolkit for developing and comparing reinforcement learning algorithms. Many projects use the Gymnasium environment to simulate and develop control algorithms for autonomous vehicles in racing scenarios[2].

  - **Key Technologies:** Reinforcement learning, deep Q-networks (DQN), and policy gradient methods are applied to optimize the driving behavior of autonomous vehicles, especially for tasks that require real-time decision-making in dynamic environments[3].

  - **Relevance:** Using reinforcement learning to optimize the driving strategy of our autonomous vehicle, especially in a competitive race, can be crucial for improving performance. Leveraging simulated environments like OpenAI Gymnasium can help us refine algorithms before deploying them on our vehicle.

- **Audi Autonomous Driving Cup**

  - **Overview:** The Audi Autonomous Driving Cup is a university competition where teams develop autonomous driving software on a 1/8th scale Audi car. The cars must navigate a complex environment with obstacles, stop signs, and intersections while following the rules of the road[4].

  - **Key Technologies:** The cars use Light Detection and Ranging (LIDAR), ultrasonic sensors, and stereo cameras for object detection and navigation. Artificial intelligence (AI) and machine learning are used for path planning, while control algorithms ensure that the cars make smooth and accurate movements.

  - **Relevance:** This project involves building robust and efficient AI and

control systems for small-scale autonomous vehicles, similar to our project's goals. The integration of sensors and control in dynamic environments makes it highly relevant.

- **F1Tenth Autonomous Racing Platform**
    - **Overview:** F1Tenth is a popular autonomous racing platform used in university courses and research. The platform consists of 1/10th scale autonomous race cars equipped with sensors (LIDAR, camera) and a robust software stack. Teams participate in races where the cars must navigate tracks autonomously at high speeds[5].
    - **Key Technologies:** The F1Tenth cars use algorithms for Simultaneous Localization and Mapping SLAM, path planning, and control optimization. LIDAR and computer vision are used for localization and obstacle detection, while the framework Robot Operating System (ROS) is commonly used for communication and coordination between components.
    - **Relevance:** This is one of the most directly applicable projects to our goal. The F1Tenth platform is designed specifically for small-scale autonomous racing and can provide a wealth of resources, both hardware and software, to support our project development.
- **NVIDIA JetRacer**
    - **Overview:** NVIDIA JetRacer is a small-scale AI-powered autonomous racing car built on the NVIDIA Jetson Nano platform. It comes with built-in support for AI-based driving and deep learning models[6].
    - **Key Technologies:** JetRacer uses deep learning models for real-time object detection, steering, and obstacle avoidance. It also supports advanced algorithms for racing in dynamic environments using computer vision.
    - **Relevance:** This platform is directly applicable to our project. It demonstrates how Artificial Intelligence (AI) models and real-time processing can be implemented on a small-scale car. The Jetson platform's compact size and powerful AI capabilities make it a similar solution for our racing vehicle.
- **Donkey Car (Open-Source Self-Driving Car Platform)**
    - **Overview:** Donkey Car is an open-source Do it Yourself (DIY) platform for building 1/10th-scale autonomous cars. It's widely used in the maker community and by robotics enthusiasts to build and race small self-driving cars[7].
    - **Key Technologies:** The system uses a Raspberry Pi for processing, a camera for vision, and a Remote Control (RC) car chassis for mobility. It

employs deep learning for steering, throttling, and obstacle avoidance. Donkey Car uses TensorFlow and Keras to train models on driving data.

- **Relevance:** Donkey Car's open-source framework provides an excellent alternative for small-scale autonomous racing vehicles. We can learn from its existing framework, AI model, and control systems to build our vehicle.

2.4 - Goals and Objectives

Our primary goal is to develop two small-scale autonomous race cars that can compete on an indoor racetrack alongside other cars with similar goals. One vehicle will have enhanced physical capabilities, including a higher maximum speed, offering distinct challenges in control and performance.

For our stretch goal, we want to improve the vehicle's performance by replacing our comparatively basic planning and control implementations with more advanced alternatives.

Below is a detailed breakdown of the objectives for achieving this project across design, software, mechanical systems, and long-term impact.

2.4.1 - Electrical Design

The goal of the electrical design is to ensure that the race cars are robust, durable and reliable, capable of withstanding multiple races without failures such as power issues, or short circuits.

Basic objectives:

- Design two Power Management System (PMS) capable of efficiently routing power from the battery(ies) to the main drive motor, the steering servo, the companion computer, the real-time flight controller, and all other sensors and electronic components of the vehicle.
    - One Power Management System (PMS) must be fine-tuned for the enhanced vehicle and be able to appropriately provide power to a significantly bigger motor.
    - Both Power Management Systems must also provide surge protection to prevent electrical malfunctions.
- Develop a System Status Indicator module capable of interfacing with the flight controller (Pixhawk 6C running PX4 firmware) to provide visual real-time feedback on the status of the vehicle system. The module would be able to display color patterns and signal issues such as low battery or power faults.

Stretch objectives:

- Create easy to follow instruction manuals on how to design the electrical components of the vehicles such as the Power Management Systems and the

System Status Indicator.

2.4.2 - Software System

The software must support the three core components of autonomous driving: perception, planning, and control [8]. These elements will allow the race cars to sense their environment, plan routes, and execute decisions in real-time.

Basic objectives:

- Develop a software implementation that would allow our vehicles to reactively follow the contours of the race track before a full map is created without getting stuck in dead ends.
- Implement Stanley for controlling the vehicle's steering.
- Implement our simultaneous localization and mapping (SLAM) software module to be able to take in points from a LiDAR or depth camera and fuse them into a unified map representation in real-time.
- Our software system must be capable of computing an optimal race line once the full race track is mapped, and the control component must be capable of pursuing that race line while avoiding obstacles.
- Implement a pure pursuit algorithm, for route planning, to allow the vehicle to generate a route adhering to the race line as well as possible while avoiding obstacles.

Stretch objectives:

- Develop a Model Predictive Control (MPC) implementation for each of our race cars and use it to replace the Stanley steering controller.
- Replace the pure pursuit algorithm with a more sophisticated graph-based planner which compute splines between segments of the racetrack and can use Dikjstra's algorithm to route the optimal path between the current position and the finish line while avoiding obstacles.

2.4.3 - Vehicle Mechanical Systems

The vehicle optimization objectives focus on enhancing key performance areas to improve racing capability by optimizing vehicle acceleration, weight, suspension characteristics and weight distribution.

Basic objectives:

- Upgrade the drive motor of one of the vehicles and reduce the vehicle's weight to achieve a > 5% decrease in 0-14.5 mph acceleration times and a power to weight ratio of over 126.579 W/kg.
- Optimize the suspension system to achieve a roll-angle reduction effectiveness

greater than 1.25, minimizing body roll and improving cornering stability.

- Optimize the weight distribution to allow the vehicle to have good handling characteristics and across a wide variety of conditions and track layouts.

Stretch objectives:

- Create a parameter-based vehicle model that enables a user to determine a vehicle setup that would be optimal for specific track conditions and layouts.

2.4.4 - Competition

Beyond achieving immediate project goals, we aim to compete in the F1Tenth Grand Prix.

Objectives:

- With our completed vehicles, compete in both the closed and open categories at the F1Tenth Grand Prix.

2.5 - Description of Features and Functionalities

Performance Optimized Mechanical Design

By default, the 1/10th scale vehicles are equipped with a Velineon 3500Kv brushless motor that delivers 721.5 Watts of power continuously. The Velineon motor is known for its power and is even used as an example of the most powerful motor allowed for use in the F1-tenth competition. With performance optimization of the 1/10th scale car in mind, the cars feature a KingVal 4300Kv brushless motor upgrade selected for its 24% increase in power compared to the stock motor. The increased power enhances acceleration and overall vehicle performance on the track. The motor's higher power motor aims to improve the 0 to 14.5 mph time by at least 5% of the original time. To adapt the upgraded motor to the 1/10th vehicle drivetrain, the vehicles are fitted with a belt drive mechanism to transfer power from the KingVal motor to the stock vehicle transmission. The belt drive mechanism is a timing belt mechanism selected for its known high load-carrying capacity, efficiency, and zero slippage over short distances.

The vehicles are set up with a 50/50 weight distribution between the front and rear axles to provide good handling and stability through the corners of a track. This even distribution helps to minimize the effects of unwanted oversteer and understeer which can negatively affect the vehicles' performance on track. To achieve this balance, lightweight 3D-printed PLA and ABS materials are used as manufacturing materials for mounting solutions of key components. PLA is known for its low density and good strength, whereas ABS is known for a lower density than PLA, but a lower ultimate tensile strength. Strategic use of both materials gives flexibility in weight positioning - enabling the targeted 50/50 weight distribution between the vehicles' axes.

Lastly, the vehicles feature a modified center of gravity (CG) and roll center position achieved by positioning critical vehicle components/masses as low as possible within the chassis and by the addition of ballast to the vehicle and by modifying the suspension setup/

geometry. By lowering the center of gravity, through adjusting the position of the vehicles' masses, the vehicles will experience less roll as the suspension system will have an effective increase in stiffness to oppose the vehicles' body roll. Reducing roll improves stability through corners; thereby, improving the overall performance.

Autonomous Sensor System and Closed-Loop Operation

As part of a closed-loop system, the vehicle will rely heavily on sensors to gather real-time environmental data, enabling precise autonomous decision-making. The sensor suite will include the following components:

- **LiDAR (Light Detection and Ranging):** The LiDAR system will emit and receive laser pulses to measure the distance between the vehicle and nearby objects, providing precise proximity data. This will be essential for collision avoidance and spatial awareness, helping the vehicle navigate tight spaces and crowded tracks.
- **RGB-D Camera (Red, Green, Blue + Depth):** The RGB-D camera will provide both color information and depth perception, enhancing the car's ability to recognize objects and interpret its surroundings. This will improve computer vision capabilities for tasks such as identifying track boundaries, signage, or obstacles. The depth data from the camera will complement the LiDAR's range measurements, improving the vehicle's understanding of its 3D environment.
- **IMU (Inertial Measurement Unit):** The IMU will monitor the vehicle's orientation, acceleration, and angular velocity, contributing to precise localization and motion tracking. This data will help the vehicle stay on course, especially during high-speed turns or when reacting to sudden changes in track conditions.

SLAM (Simultaneous Localization and Mapping)

The vehicle's SLAM algorithms will integrate data from the LiDAR, RGB-D camera, and IMU to create and continuously update a real-time map of its surroundings. This enables the car to localize itself accurately within the map, even on dynamic or previously unseen tracks. The SLAM system will also support:

- Dynamic Obstacle Avoidance: As new obstacles appear during a race, the SLAM system will detect and account for them in real time, allowing the car to quickly adjust its path.
- Real-Time Path Adjustments: SLAM will enable the vehicle to recalculate its route on the fly, ensuring it follows the optimal race line while avoiding obstacles and maintaining speed.

Path Following and Control System

The vehicle will be equipped with advanced path-following functionality to navigate the race track based on the map generated by the SLAM (Simultaneous Localization and Mapping) algorithm. Once the SLAM system finishes creating the map, the vehicle will compute an optimal race line which minimizes the time it takes to complete each lap. Once

that race line is computed, the car will begin continuously pursuing the point on the race line a certain distance ahead of it, computing paths to avoid obstacles as it does so.

The path-following algorithms will optimize the trajectory by factoring in the vehicle's dynamic capabilities, such as:

- Maximum speed limits
- Acceleration and deceleration rates
- Cornering limits to maintain stability at high speeds

This system ensures that the car not only follows the most efficient race line but also adapts its speed dynamically to maintain stability and control through sharp turns, straightaways, and other track conditions.

External Control and Communication System

Although the vehicle is designed to operate autonomously, it will also include an external control and communication system to provide versatile operational modes.

- **Manual Control with RC Controller:** The vehicle can be operated manually via a remote RC controller for testing, debugging, or when manual intervention is needed during development. This feature ensures flexibility in diagnostics and system adjustments without requiring full autonomy during early-stage testing or emergency situations.
- **Real-Time Monitoring Systems:** Onboard displays will allow the team to monitor critical performance metrics throughout testing and races. These displays will provide instant feedback on:
    - **Power Distribution and Battery Health:** Status indicators will monitor power levels to detect surges, drops, or battery depletion in real-time, ensuring smooth operation.
    - **Vehicle Speed:** A speed monitoring system will provide continuous feedback on how the vehicle performs relative to its expected race targets, helping to evaluate the effectiveness of the control algorithms.

These monitoring systems will improve safety and performance assessment by giving the team quick access to essential information during races and troubleshooting sessions.

Software Architecture

All the vehicle's software systems will run on a high-performance onboard computer equipped with a GPU (Graphics Processing Unit). This architecture ensures that all key processes are handled with low latency, critical for real-time decision-making during high-speed races. The software stack will include:

- **ROS (Robot Operating System):** ROS will act as the middleware to coordinate communication between the sensors, path-planning algorithms, and control systems.

- **Nav2 (ROS 2 navigation package):** This framework will act as the structure organizing the control of the vehicle. It will determine what behavior pattern the vehicle should be following at any given time, and handle calls to the modular components that represent individual parts of the navigation process (e.g. path planning).

- **SLAM:** The SLAM module will continuously update the vehicle's map and localization data, enabling the vehicle to react to changes in its environment.

- **Path Planning Algorithms:** These algorithms will compute optimal paths and re-plan routes dynamically to avoid obstacles or adjust for changing track conditions.

- **Controller:** The controller will take in the desired steering angle and smoothly guide the vehicle's steering towards it, preventing oscillations.

The GPU-accelerated system will enable rapid processing of sensor data, including LiDAR and RGB-D inputs, ensuring that perception, planning, and control algorithms run efficiently. This allows the vehicle to maintain a high degree of accuracy while making split-second decisions essential for competitive racing.

Integration Performance

By combining high-performance mechanical components with advanced sensors, real-time decision-making, and sophisticated control algorithms, the vehicle will maintain precision, speed, and agility under diverse racing conditions. The seamless synergy between hardware and software systems enables the vehicle to dynamically respond to track environments, avoid obstacles, and continuously pursue the optimal path. This integration will provide a competitive edge on the racetrack, ensuring that the vehicle can adapt to challenging layouts and high-speed scenarios while maintaining peak performance.

The path-following system, enabled by SLAM (Simultaneous Localization and Mapping) algorithms, will guide the vehicle along the most efficient race line, adjusting steering and speed in real-time based on dynamic track conditions. The use of external control options, such as manual operation through a remote RC controller, offers flexibility during testing and troubleshooting. Additionally, real-time performance monitoring will provide insights into critical metrics like power distribution, battery health, and speed, ensuring safe and efficient operation throughout races and development phases.

A high-performance onboard computer with GPU acceleration will ensure that all processes, from sensor data acquisition to path execution, are handled with low latency, enabling the vehicle to make fast, accurate decisions under pressure. Key software components, including ROS middleware, SLAM, and path-planning algorithms, will work together to maintain smooth communication between sensors, controllers, and actuators.

This integrated approach ensures the vehicle remains reliable and responsive in all scenarios. Whether autonomously navigating through a race or under manual control for testing, the vehicle will adapt to changing environments and maintain optimal performance throughout each phase of the race.

System Safety and Reliability

Safety is a core priority in the design and operation of the vehicle. The design process of the vehicle's hardware, software, and control systems encompass ways to prevent malfunctions, mitigate risks, and respond effectively to unexpected situations during high-speed racing. Several safety measures will be integrated across the vehicle's architecture to ensure reliable, safe, and efficient operation.

- **Collision Avoidance and Fail-Safe Feature:**
  The LiDAR and RGB-D camera systems will work together to detect obstacles, monitor proximity to other vehicles, and prevent collisions. In addition to autonomous operation, the vehicle will include a manual control override. This would allow direct control in case of software failure or to assist with recovery during. A kill switch or emergency stop feature will be integrated to cut power to the drivetrain and halt the vehicle immediately if unsafe conditions arise, preventing damage to the vehicle.

- **Power and Performance Monitoring:**
  Real-time monitoring on key metrics, such as speed and power, giving insight into the vehicle's status allowing monitoring of faults before they arrive. The power distribution system will be able to detect faults, surges, or low voltage before they lead to performance issues or failures.

- **Robust Control Stability and Stability:**
  The vehicle's control algorithms will adjust steering, braking, and acceleration to prevent loss of control during rapid maneuvers and high-speed racing. This is especially critical when navigating sharp corners or uneven sections of the track.A finely tuned suspension system will help maintain stability and traction, reducing the risk of rollovers or skidding during aggressive driving.

With a robust combination of collision avoidance, manual override options, power management, stability control, and real-time monitoring, the vehicle prioritizes safety without compromising performance. This comprehensive safety framework ensures the vehicle can operate reliably in both autonomous and manual modes, protecting the integrity of the car and the racing environment.

Community Contribution

This project aims to make a meaningful and lasting contribution to the autonomous racing community, inspiring other teams, universities, and individuals to engage in autonomous vehicle design and racing, especially here at the University of Central Florida.

- **Comprehensive Instruction Manuals:**

To ensure the continuity of the project or interest and facilitate knowledge transfer, detailed instruction manuals will be developed. These manuals will serve as both a technical guide for building the vehicle and a resource for future teams or enthusiasts interested in autonomous racing. They will cover every aspect of the project, from hardware design to software implementation and testing protocols, ensuring the project can be replicated, improved, or adapted by others.

- **Workshop and Knowledge Transfer:**
  Collaboration with organizations such as the Institute of Electrical and Electronics Engineers (IEEE) and the Association of Computing Machinery (ACM) will be made to host educational workshops. These sessions will provide hands-on training on topics such as autonomous vehicle development, ROS, SLAM, and model predictive control.

- **Registered Student Organization:**
  The creation of a Registered Student Organization (RSO) for autonomous racing at the University of Central Florida (UCF) will ensure the project's continuity beyond the senior design phase. The RSO will provide future students with access to resources, mentorship, and competition opportunities to develop their own vehicles.

- **Competitions:**
  Our Senior Design Team will compete in the F1Tenth Grand Prix in 2025.

By doing these things the project aims to inspire others to engage in autonomous racing, fostering innovation and collaboration. This legacy will ensure the project's impact continues to grow, driving forward the development of autonomous vehicle technology both within the University of Central Florida and beyond.

2.6 - Key Specifications Table

The table below outlines the key performance parameters and design specifications that will guide the development of the autonomous race vehicles. These specifications ensure that the vehicles achieve a balance between speed, efficiency, and control while meeting the technical demands of high-speed racing.

Each item addresses a critical aspect of the vehicle's operation. Tracking those specifications will allow us to understand and have a broad idea of where we stand on the design of key components. Meeting or exceeding these benchmarks will be essential to optimizing the vehicle's agility, reliability, and endurance throughout the race.

Table 2.6.1: Key Specifications

Item	Parameter	Specification
Vehicle weight and weight distribution	Static weight distribution (front to rear)	50/50
Race line	Latency	< 10s

computation		
Race line following algorithm	Replanning time	< 0.5s
System Status Indicator module	Number of statistics on display	5
Power distribution board	Efficiency	> 60%
Odometry estimations	Accuracy	± 15cm
Battery	Runtime	> 10 minutes
Vehicle Performance	0 - 14.5 mph time	> 5% decrease
Vehicle Suspension	Roll-angle reduction mechanism effectiveness	> 1.25
Vehicle weight and weight distribution	Power-to-weight ratio	> 126.579 W/kg
Battery	Capacity	> 5000mAh
Power distribution board PCB	Number of layers	2
Model Predictive Control Set-point tracking	Overshoot	< 20%
Model Predictive Control Set-point tracking	Rise-time	< 1 second

2.7 - Hardware Block Diagrams

In the design and development of our project, understanding the interplay between various hardware components and properly interfacing them is important. The block diagrams in this section serve as a blueprint for the main hardware components of the project, offering a structured representation of the physical systems that work in unison to power the vehicle, process data, and respond to environmental stimuli. The block diagrams in this section also serve as visual representations of the critical hardware elements and their interconnections.

Overview

The hardware block diagram below provides a high-level view of the components that form the foundation of the vehicle's operation. Each subsystem plays a critical role in ensuring seamless performance, from power management and the computing unit to sensors and control systems. The integration of components enables the vehicle to execute precise maneuvers while maintaining speed and stability. These hardware components are essential for ensuring efficient power distribution and supporting the advanced autonomy required for competitive racing.

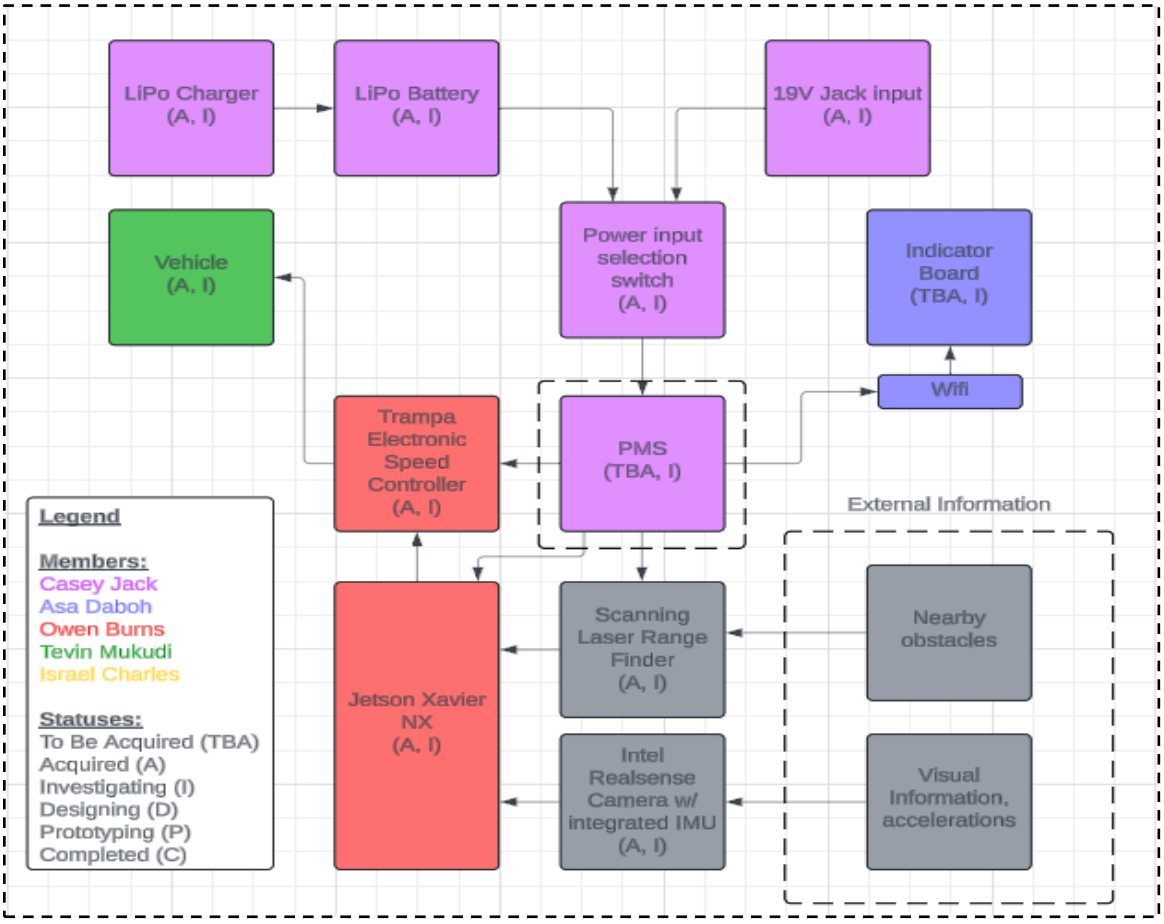


Figure 2.7.1: Hardware Block Diagram for Autonomous Vehicle

Power Management System (PMS)

The power management system is a critical component of the vehicle, ensuring efficient distribution of electrical power to all subsystems. The block diagram above illustrates the flow of power from either the batteries or the power input jack to key components such as the speed controller, companion computer, and sensors. The Power Management System (PMS) plays a central role by regulating voltage levels, routing power efficiently, and preventing surges.

Additionally, battery monitoring mechanisms are in place to track power levels in real-time, ensuring safe operation and providing alerts for low battery conditions. An optional 19V jack input will be placed as a secondary power input for testing and diagnostic use

only, not meant to be used during actual use of the vehicle. The power sources to the power management system can be selected using an onboard switch.

This integrated power management system is essential for maintaining stability, reliability, and peak performance.

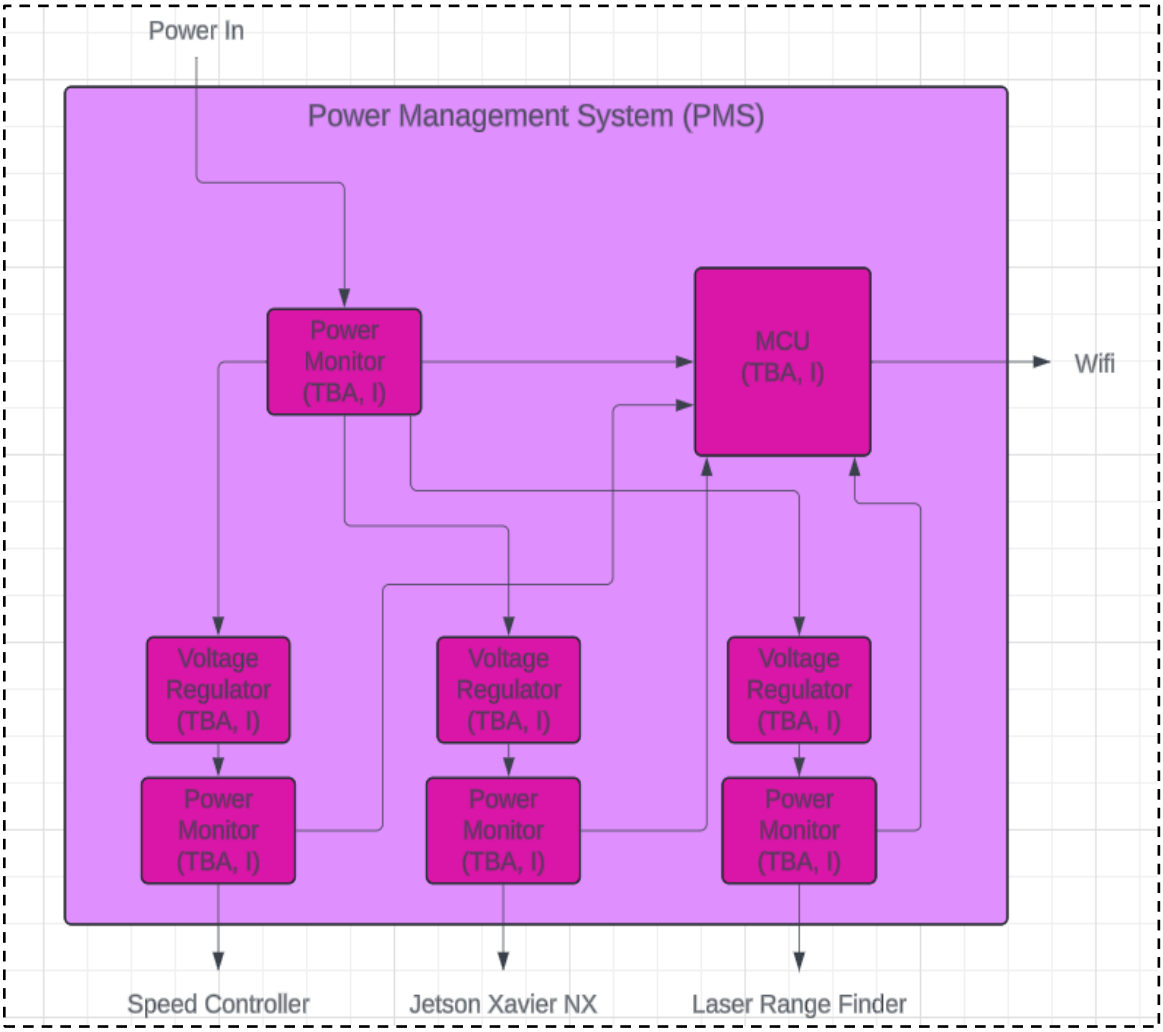


Figure 2.7.2: Hardware Block Diagram for Power Management System

System Status Indicator module

The System Status Indicator module is a crucial component in ensuring real-time monitoring and effective communication of the vehicle's performance and operational health. This module acts as the vehicle's "dashboard," providing vital feedback to operators or automated systems, enabling them to assess critical aspects of its status swiftly and accurately.

As illustrated in the hardware block diagram below, the System Status Indicator module is intricately connected to key subsystems, such as the power management unit, sensors, and control interfaces. This integration allows the module to gather and relay essential

information seamlessly during operation. The module's intuitive design prioritizes user-friendliness, ensuring that information is not only accurate but also easy to interpret, even under high-pressure conditions.

The module offers a wide range of indicators to monitor the vehicle's state effectively. These include, but are not limited to, battery health, power distribution levels, operational temperatures, and system error alerts.

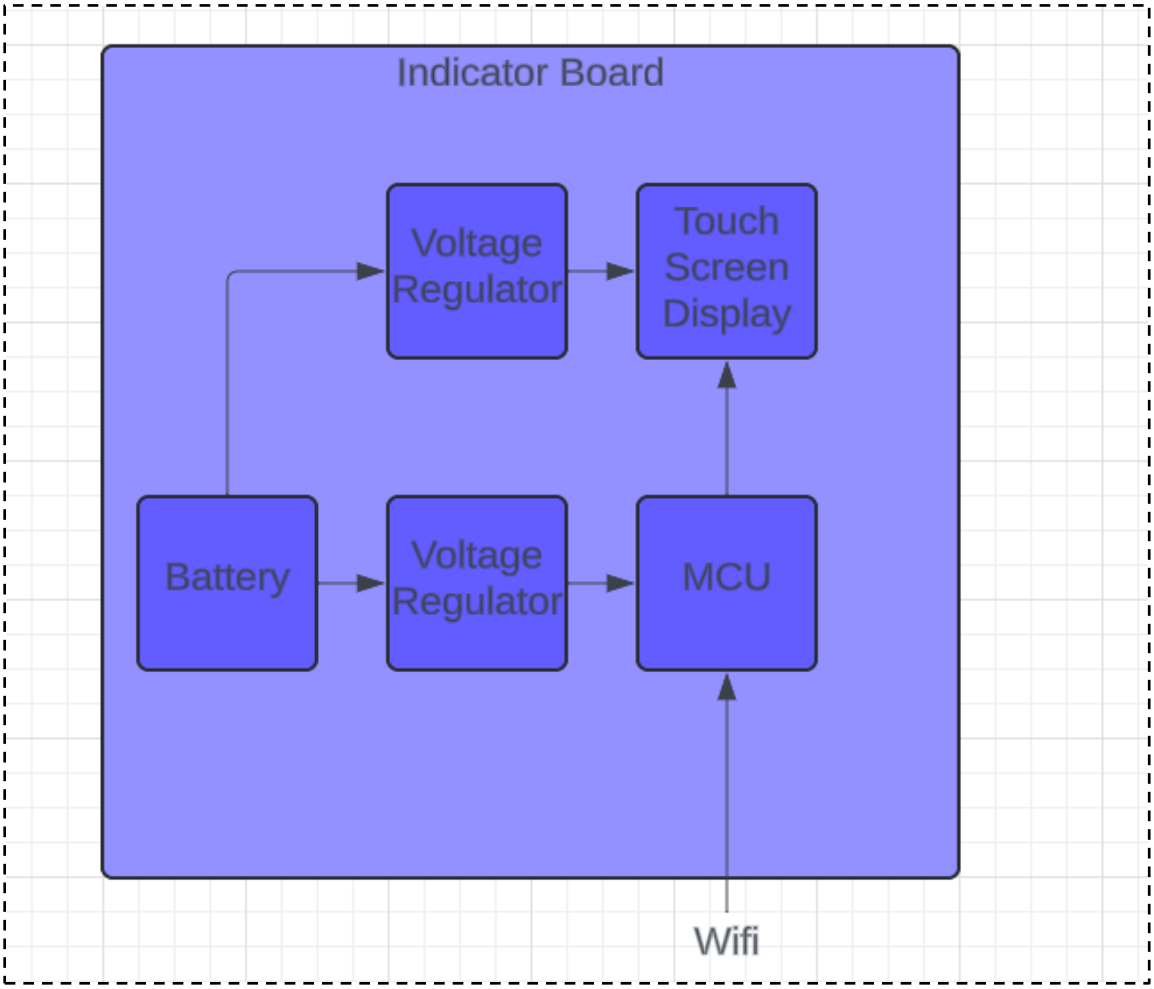


Figure 2.7.3: Hardware Block Diagram for indicator module

2.8 - Software Block Diagram

The software block diagram below outlines the architecture of the vehicle’s software systems, which are integral to its autonomous operation and overall performance.

This architecture includes several key modules: Robot Operating System (ROS) for middleware functionality, the SLAM (Simultaneous Localization and Mapping) algorithms for real-time mapping and localization, and the path-following control algorithms that dictate the vehicle’s navigation along the race line.

Each module communicates seamlessly with one another, facilitating the flow of data from sensor inputs (such as LiDAR and RGB-D cameras) to decision-making processes and ultimately to actuator commands for the drivetrain and steering. This interconnected system enables the vehicle to respond rapidly to changing track conditions, execute precise maneuvers, and maintain optimal speed and control.

The software architecture is designed to maximize performance and efficiency, making it a critical component in achieving competitive racing outcomes while ensuring reliability and safety during operation.

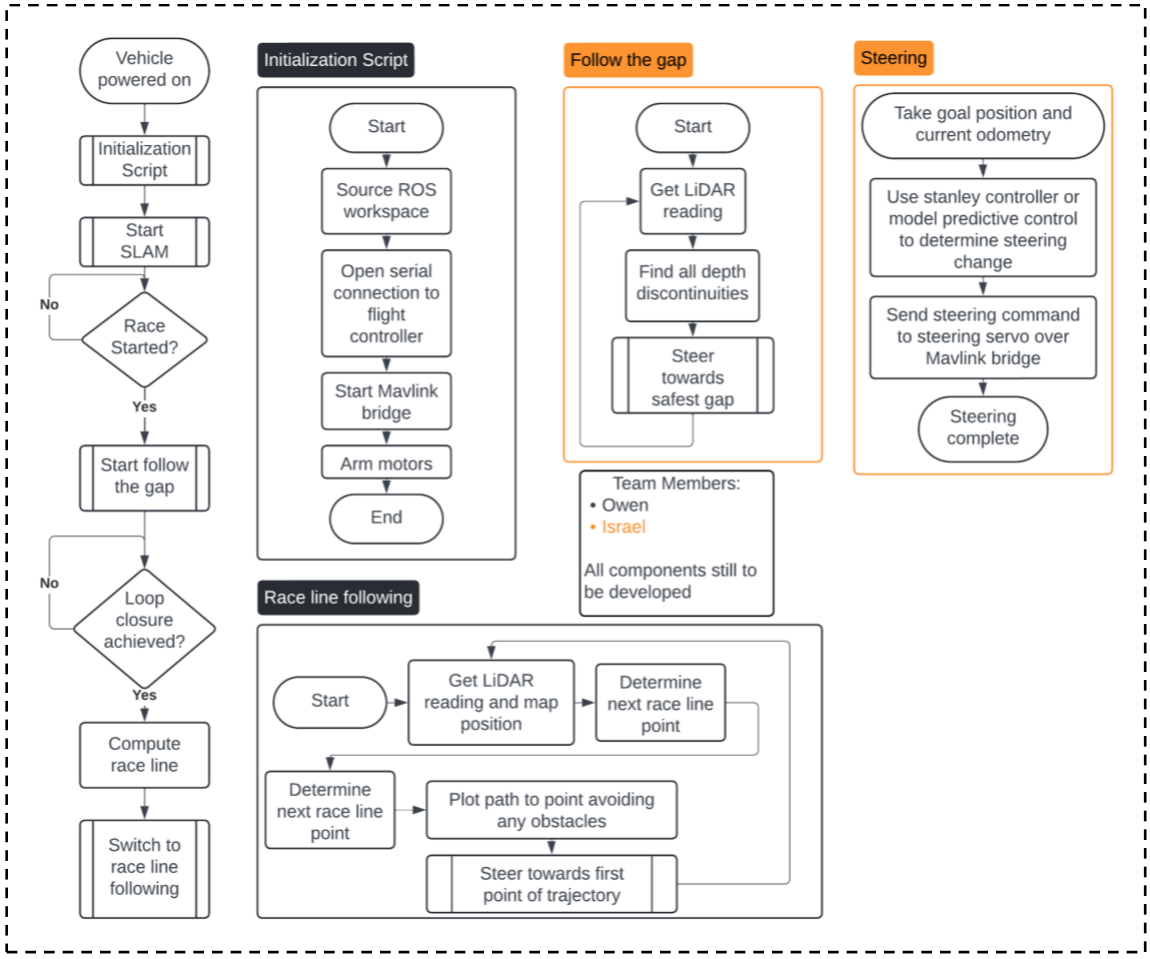


Figure 2.8.1: Software Block Diagram

2.9 - House of Quality

The House of Quality (HoQ) translates customer requirements into specific technical specifications for the vehicle's design and performance. The diagram below visually represents the relationship between various customer needs and the corresponding engineering characteristics that must be addressed to meet these expectations. The legends and keys used in the house of quality can be found in the table that follows the house of quality. While working on this project, the requirements for engineering are subject to change, but ultimately, we believe that this best summarizes both needs that can be met by

the House of Quality.

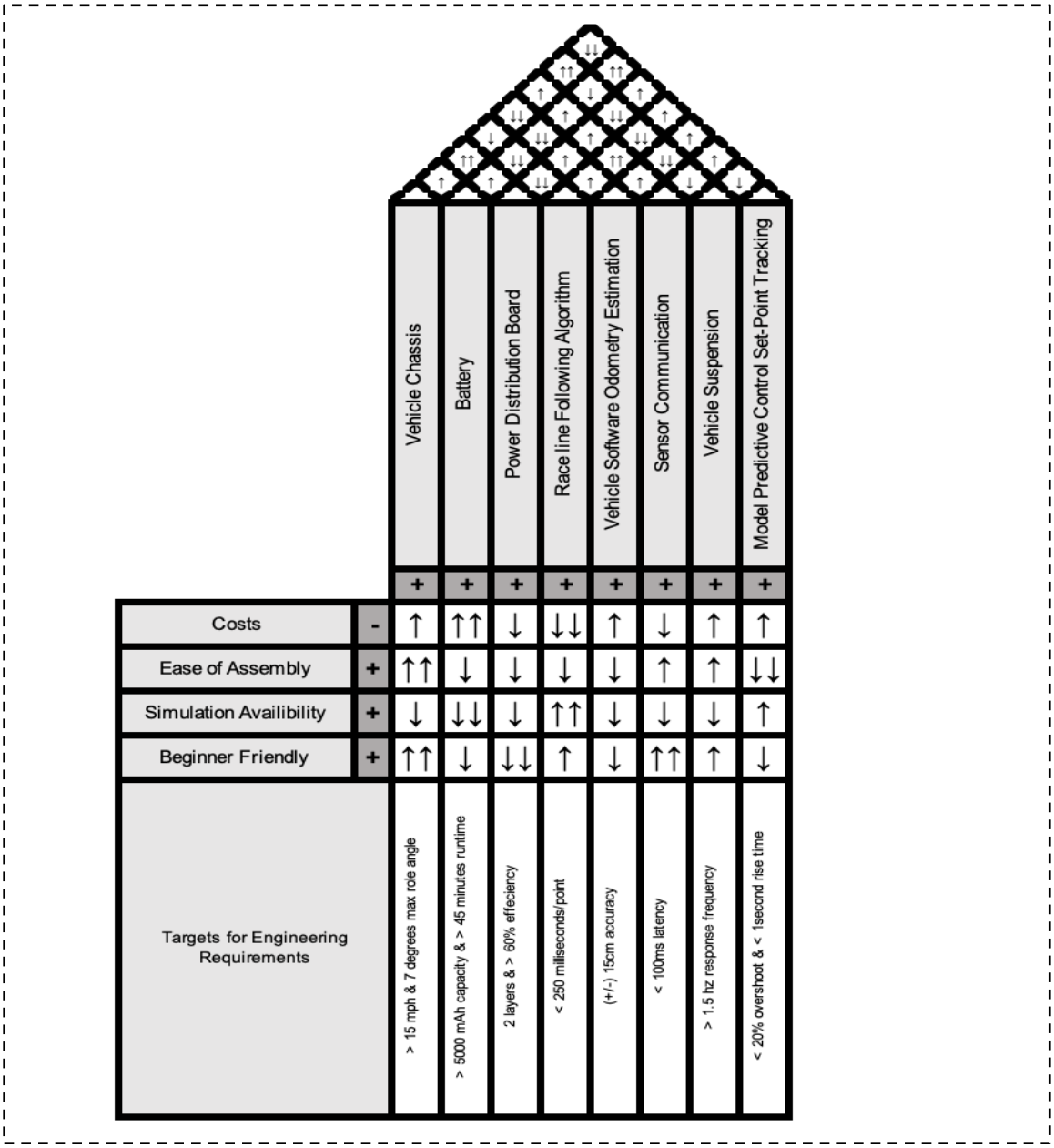


Fig 2.9.1: House of Quality

The table below outlines the legends and keys used in the House of Quality above.

Table 2.9.1: House of Quality Key

Symbol	**Meaning**
↑	Positive Correlation

↓	Negative Correlation
⇈	Strong Positive Correlation
⇊	Strong Negative Correlation
+	Needs to be added
-	Needs to be taken out
Targets for Engineering Requirements	
Costs, Ease of Assembly, Simulation Availability, Beginner Friendly	Marketing Requirements
Vehicle Chassis, Battery, Power Distribution Board, Race Line Following Algorithm, Vehicle Software Odometry Estimation, Sensor Communication, Vehicle Suspension, Model Predictive Control Set-Point Tracking	Engineering Requirements

3 - Research and Investigation

For our Small-Scale Autonomous Vehicle project, our research and investigation were critical to identifying and selecting the components and designs that best align with our goals. Through a methodical approach, we evaluated various options, analyzed their suitability, and established a strong basis for development. Our investigation encompasses the technical, functional, and practical aspects of each component, emphasizing their integration into a cohesive system. By carefully assessing available technologies and platforms, we aim to ensure that each choice supports the performance, durability, and scalability required for the project's success.

3.1 - Vehicle Chassis

One of the key steps in our research involves selecting an appropriate chassis for the small-scale autonomous vehicle. We explored a range of prebuilt platforms, particularly RC (Remote Control) cars, to identify a base that meets our specific requirements. The ideal chassis must be robust enough to endure repeated testing, adaptable for mounting additional components such as computing units, LiDAR, cameras, and IMUs, and simple to modify for custom configurations.

We also prioritized vehicles that resemble full-sized electric cars, both in design and components. Specifically, we looked for chassis with brushless motors and electric speed controllers (ESCs) since these provide greater efficiency, performance, and realism compared to brushed motors [69]. Additionally, we sought models with expandable battery options, as the extra components we'll be adding will demand more power.

3.1.1 - Part Comparison

Below are some of the notable findings of potential small scale vehicle chassis that could fit out need.

HSP 1/8th and 1/10th RC car Chassis

The HSP 1/8th and 1/10th RC cars, which are both solid options for our small-scale autonomous vehicle project. These RC cars are actually recommended by the Donkey Car open-source project, which is great since they're already known to work well for projects like ours [70]. The stock HSP 1/8th 94996 models are 20.5 inches in length, 15 inches wide and have a height of 9.8 inches. They have a 3300KV brushless motor, and a 7.4V 3800mAh LiPo Rechargeable Battery. As of today, October 24th, 2024, the HSP 1/8th 94996 is priced at $460 from the online store BomeToys [71]. For the stock HSP 94118 PRO 1/10th models are 18.7 inches in length, 9.9 inches wide and have a height of 6.5 inches. They have a 3300KV brushless motor and a Li-Po 7.4V 3500mAh battery [72]. As of today, October 24th, 2024, the HSP 1/8th 94996 is priced at $300 from the online store BomeToys. The HSP RC cars are manufactured by Hspacing Co., LTD, a company in China.

These cars are designed for researchers and hobbyists, meaning the parts are easy to swap out and upgrade. We'll be able to remove the body, adjust the motor, or add sensors like cameras, LiDAR, and IMUs without much hassle. The HSP chassis has enough room to fit additional components like a Raspberry Pi or Jetson Xavier (for computing) along with the LiDAR and IMU. The wiring won't feel too cramped, especially on the 1/8th model. Compared to other racing cars like the Traxxas Slash, the HSP models are more affordable. This would leave us with more budget to spend on the sensors and other electronics we need. Both the 1/8th and 1/10th models have good motors and speed control, so we can test algorithms like our Follow the Gap without worrying that the car will struggle to keep up. Since these models are used in projects like Donkey Car, we can rely on existing community support, tutorials, and forums if we run into problems. There are also step-by-step guides available for modifying the chassis.

However, some of the parts (especially suspension and steering components) are made of plastic, which might wear out faster if we push the car hard during testing. We might have to upgrade these parts if they break. The 1/10th model might be a bit tight if we want to fit heavier sensors like LiDAR or larger batteries. It could limit how many additional components we can add. While it's cheaper, the HSP cars aren't as tough as the Traxxas Slash. If we plan to run a lot of outdoor tests or on rough tracks, the HSP might need more repairs. The stock battery and motor might not last very long with all the extra sensors and electronics. We might need to upgrade the power system to ensure everything runs smoothly for extended tests.

Kyosho 1:8 4WD Racing Buggy and 1:10 4WD Fazer Mk2 Chassis

The Kyosho 1:8 4WD Racing Buggy and 1:10 4WD Fazer Mk2 are also both solid options for our small-scale autonomous vehicle project. Known for their high-quality builds, these Kyosho models are widely respected in the RC community and are a great fit for research

and robotics. They also have powerful motors and durable designs, giving us flexibility to experiment with different environments and algorithms.

Kyosho is a well-established manufacturing brand headquartered in Japan, known for high-quality RC cars with a reputation for performance and durability. The Kyosho 1:8 4WD Racing Buggy INFERNO MP9e is 19.5 inches long, 12.1 inches wide, and 7.8 inches high. It features a 2300KV brushless motor and are recommended to run on 2 units of 7.2V Ni-MH battery. As of October 24th, 2024, this buggy is priced at $550 from Kyosho's official online store [73]. The Kyosho Fazer Mk2 1:10 is 14.8 inches long, 8.3 inches wide, and 5 inches high. It is powered by a 550 Class G-Series Motor G14L brushless motor and recommended to run on a 7.2V Ni-MH battery [74]. This model is currently priced at $240 from the same store. Note that with the Kyosho RC cars, the batteries need to be bought separately.

These Kyosho cars are built with researchers and hobbyists in mind, making them easy to upgrade and modify. We'll have no trouble installing sensors like cameras, LiDAR, and IMUs or attaching a Raspberry Pi or Jetson Nano to handle computing tasks. With their sturdy chassis and flexible design, we can quickly swap out parts if needed and experiment with new components. Both models offer powerful motors and reliable control, allowing us to test advanced algorithms like Follow the Gap while maintaining speed and agility. The 1:8 Racing Buggy has extra space to accommodate larger computing units and sensors, giving us more freedom in our setup. The 1:10 Mk2 is more compact, making it ideal for indoor use or tighter tracks. Although Kyosho models are a bit more expensive than some other options like HSP, they provide excellent durability and smoother handling, reducing the need for frequent repairs. This means fewer headaches when testing outdoors or on rougher terrain, which makes them a solid long-term investment for the project.

Kyosho is a popular brand with plenty of community resources. While not as focused on open-source projects as Donkey Car, Kyosho's large user base means we'll have access to forums, tutorials, and repair guides if we need help or want to upgrade the chassis [75].

The 1:8 Racing Buggy is larger and heavier, which could drain the battery faster when we add sensors and computing units. We may need to invest in additional batteries or a more powerful power system to keep everything running smoothly. While the Fazer Mk2 is smaller and more affordable, it might not have enough space to fit larger sensors like a 3D LiDAR or an advanced onboard computer. This could limit the kind of experiments we can do. Although Kyosho cars offer better durability and performance, they are more expensive upfront. Especially with having to buy batteries separately. Although a lot of the other options involve buying batteries separately as well - which is understandable because battery needs might be different from one user to another, especially if it involves powering up other devices with the battery, for our case, the computing unit.

Tamiya TT-02 Chassis

The Tamiya TT-02 is another excellent option for our Small-Scale Autonomous vehicle project. It's the same model used in NVIDIA's JetRacer platform, which is built for AI-powered RC cars, making it a strong candidate for our needs. Known for its reliability and versatility, the TT-02 is well-suited for modifications and the addition of sensors, cameras,

and computing units like a Raspberry Pi or Jetson Nano [76]. The TT-02 comes with a 540 brushed motor and supports 7.2V NiMH or 2S 7.4V LiPo batteries, striking a good balance between performance and energy efficiency. The model measures 16.7 inches in length, 7.3 inches in width, and 5.8 inches in height, with a total weight of around 1.5 kg. As of today, October 24th, 2024, the Tamiya TT-02 kit is priced at $150, available through online retailers like Amazon, Hobby Town or directly from Tamiya [77].

One of the biggest advantages of the TT-02 is that it's lightweight and easy to customize. Since it's used in projects like NVIDIA JetRacer, we know it can handle modifications such as mounting a Jetson Nano, a camera module, or even a small LiDAR unit. The chassis is fairly spacious for its size, so there's room to run wires and attach lightweight sensors without feeling cramped. Its modular design means we can easily swap parts or make adjustments as we refine our algorithms. This car's design makes it a popular choice for projects that involve machine learning and AI because it allows smooth installation of components for object detection and autonomous navigation tasks.

However, there are a few limitations we need to consider. The TT-02's motor is brushed, which is not as powerful or efficient as the brushless motors found in higher-end models. This means it might not handle high speeds as well, especially if we load it with heavier sensors and computing units. If we plan on testing algorithms like Follow the Gap at higher speeds, we may need to upgrade the motor or modify the power system. Additionally, since the TT-02 is more of an on-road model, it performs best on smooth surfaces. If we want to test our vehicle outdoors or on uneven terrain, the lack of suspension travel and limited ground clearance could become an issue, potentially requiring additional modifications to the chassis.

Despite these limitations, the TT-02 offers excellent value for the price. At around $150, it is far more affordable than alternatives like the Kyosho models or Traxxas Slash, which allows us to dedicate more of our budget to important sensors and computing units. Repairs and replacements are also straightforward, with a wide availability of parts and a strong community of users who share tips, upgrades, and troubleshooting advice. Thanks to its connection with NVIDIA JetRacer, there are also pre-built code libraries and detailed guides available to help us implement AI-based features, saving us time during development.

In summary, the Tamiya TT-02 is an excellent option if we plan to focus on indoor testing or smooth track environments and need a lightweight, affordable chassis to integrate with advanced AI systems. Its ease of customization makes it ideal for our project, especially if we want to experiment with computer vision or reinforcement learning algorithms. However, if we plan on outdoor testing or need more power and ground clearance, we might need to either upgrade components or consider a more rugged alternative. Overall, the TT-02 offers a solid combination of affordability, community support, and compatibility with AI tools, making it a smart choice for our autonomous vehicle development.

Traxxas Slash 4x4 Ultimate Chassis

The Traxxas Slash 4x4 Ultimate is a top-tier option for our small-scale autonomous vehicle

project, especially since it's used and recommended in prestigious competitions like the F1Tenth Autonomous Racing Platform, which we plan on participating in [78], and the MIT Racecar Competition [79]. These competitions focus heavily on high-speed control, path planning, and obstacle avoidance, which makes the Slash 4x4 a great fit for our project's needs. Known for its ruggedness, power, and reliability, the Slash 4x4 is a short-course truck that excels in both performance and durability, giving us plenty of flexibility to modify it for autonomous driving.

The Traxxas Slash 4x4 Ultimate measures 22.4 inches in length, 11.7 inches in width, and 7.6 inches in height, with a ground clearance of 2.8 inches, making it well-suited for off-road environments. It comes equipped with a Velineon 3500 brushless motor and a 3S 11.1V LiPo battery can be purchased as an add on, delivering impressive speed and power for advanced driving algorithms. As of October 24th, 2024, the Slash 4x4 is priced at $500 on most online stores like Traxxas's official site and Horizon Hobby [80]. Traxxas, headquartered in Texas, is a highly regarded RC manufacturer, known for producing durable vehicles capable of handling rough conditions—something that will be valuable as we push our car to its limits in testing.

The Slash 4x4's durability and adaptability are two of its biggest strengths. It's built to handle high speeds and rough terrain, meaning we can test our algorithms outdoors without worrying too much about damage from collisions or obstacles. Its suspension system is excellent, providing the stability needed for fast turns, even on uneven surfaces. This makes it ideal for our project if we plan to run it in varied environments, such as indoor tracks, parking lots, or gravel paths. The Slash also offers plenty of space for mounting sensors, making it easy to attach a computing unit like a NVIDIA Jetson Xavier, along with a LiDAR sensor, camera, and IMU. With its sturdy chassis, we can add these components without compromising the car's performance or agility.

One of the biggest advantages of using the Slash 4x4 is that it's already proven in competitions like F1Tenth and MIT's racecar program. This means that much of the groundwork for integrating sensors and software has already been laid out, with open-source resources, detailed guides, and active forums to help us troubleshoot and improve our setup. We'll also have access to ready-made libraries for autonomous navigation, which can save us time and allow us to focus on tuning our algorithms, like Follow the Gap, rather than reinventing the wheel.

However, the Slash 4x4 does have some drawbacks. It's more expensive than other RC cars like the Tamiya TT-02 or HSP models, which could stretch our budget, especially if we need additional sensors or computing hardware. The high speed and power of the brushless motor also mean battery life might be a concern, particularly when running computationally intensive tasks like real-time image processing or SLAM. We may need to purchase spare batteries or an upgraded power system to ensure the vehicle performs consistently during long tests. Additionally, while the Slash is built for durability, repairs can be costly, and finding replacement parts might take time if we damage key components during aggressive testing.

Overall, the Traxxas Slash 4x4 Ultimate is a powerful and versatile platform that offers everything we need to build a competitive autonomous vehicle. Its proven track record in

high-profile competitions, combined with its rugged design and performance, make it a great long-term investment for our project. If we plan on outdoor tests, high-speed experiments, or challenging environments, the Slash 4x4 is the ideal choice. However, if budget constraints are a concern or if we only need a car for light indoor testing, a more affordable option like the Tamiya TT-02 or HSP models could be considered. But for a project that aims for excellence and performance under real-world conditions, the Slash 4x4 provides the perfect balance of speed, durability, and support to take our autonomous vehicle project to the next level.

LaTrax Rally 1/18 Scale 4WD Chassis

The LaTrax Rally is a compact and affordable option for our small-scale autonomous vehicle project. Manufactured by Traxxas, it offers a more accessible entry point into autonomous driving, especially for projects focused on indoor environments or smaller-scale testing. The LaTrax Rally is 1/18th scale, with dimensions of 10 inches in length, 4.7 inches in width, and 3.9 inches in height, and it weighs around 0.5 kg. It comes equipped with a 370 brushed motor and a 6.0V 5-cell NiMH battery, which offers enough power for basic driving tasks. As of October 24th, 2024, the LaTrax Rally is priced at $140, making it one of the most affordable RC cars that could be used. It is available on their official website and major hobby retailers [81].

The lightweight design and compact size of the LaTrax Rally make it a good fit for indoor testing or controlled environments like small tracks. It’s easy to handle, and its small frame allows for smooth navigation through tight spaces, which is ideal for developing and testing autonomous algorithms such as object detection or simple path planning. Thanks to its connection with Traxxas, the LaTrax Rally is modular and can be upgraded with ease. While it doesn't offer as much room as larger models like the Slash 4x4, it can still support small computing units like a Raspberry Pi along with a basic camera or IMU for data collection and control.

However, the compact size of the LaTrax Rally does come with some trade-offs. Its brushed motor isn’t as powerful as the brushless motors found in larger vehicles, which means it won't perform as well at high speeds or on rough terrain. Additionally, the battery life is limited, especially once we start adding computing hardware and sensors. This may require frequent battery swaps or upgrades to keep up with our testing needs. The smaller chassis also limits how many components we can add, so more advanced sensors, such as LiDAR or Jetson-based systems, might not fit comfortably. This makes it more suitable for lightweight applications where only minimal processing is needed, such as testing basic obstacle avoidance algorithms at low speeds.

The affordable price and ease of use make the LaTrax Rally a great option if we’re looking to experiment with entry-level autonomous systems without breaking the budget. However, for more complex tasks like outdoor navigation, high-speed testing, or multi-sensor integration, the Rally might struggle to meet the demands. It’s an excellent tool for early-stage development or indoor prototyping, but if we want to push our project further, we may need to transition to larger models that offer more power and room for components.

Traxxas X-Maxx Chassis

The Traxxas X-Maxx is on the opposite end of the spectrum, representing one of the most powerful and rugged RC platforms available for autonomous vehicle development. It's a 1/6th scale monster truck, with dimensions of 30.7 inches in length, 21.3 inches in width, and 13.8 inches in height, weighing a hefty 8.6 kg. The X-Maxx is equipped with a Velineon 1200XL brushless motor and supports up to 2 units of 4S LiPo batteries (29.6V), giving it unmatched speed, torque, and runtime. As of October 24th, 2024, the Traxxas X-Maxx is priced at $1,100, available from Traxxas and premium RC hobby stores (Batteries sold separately) [82].

The X-Maxx's large size and incredible power make it the ideal platform if we plan on running outdoor tests on rough terrain, gravel, or uneven surfaces. Its suspension system and high ground clearance allow it to tackle any environment, from dirt tracks to off-road trails. This durability gives us the freedom to push the vehicle to its limits during high-speed experiments without worrying about damaging components. The X-Maxx's spacious chassis offers plenty of room to mount a Jetson Xavier, LiDAR unit, camera, IMU, and additional power systems. The size also makes it easier to route wiring without the risk of components overheating or interfering with each other, giving us flexibility in designing complex setups.

The high power output and speed of the X-Maxx are especially useful for testing advanced algorithms like high-speed path planning, multi-sensor fusion, and collision avoidance, where a fast, responsive vehicle is essential. It's built to handle the most demanding projects, which makes it a good fit for outdoor autonomous competitions or scenarios that require real-time decision-making in tough environments. The durability and raw power of the X-Maxx ensure that it won't just survive rough conditions but thrive in them, reducing the downtime for repairs and letting us focus on refining our algorithms.

However, the size, weight, and price of the X-Maxx are factors we need to consider carefully. At around $1,000, it is one of the most expensive RC cars on the market, and this doesn't include the cost of additional batteries, sensors, or computing units, which could quickly push the total cost even higher. The larger size and weight also mean that transporting it could be challenging, especially if we need to bring it to different testing locations. While power is a major advantage, it also demands more energy. Running advanced sensors alongside the motor will require upgraded batteries, which could increase operational costs over time.

In conclusion, the Traxxas X-Maxx is the perfect choice if we are aiming for high-performance outdoor testing, especially for algorithms that require speed, stability, and power. It's a long-term investment that can handle rugged environments and complex setups, making it a great platform for competitive autonomous projects or real-world testing scenarios. However, for smaller-scale projects or those with limited budgets, the X-Maxx may be more than we need. If we plan to run lighter tests in indoor environments or with fewer sensors, more compact and affordable options like the LaTrax Rally or Tamiya TT-02 might be more practical. But if we're aiming for a robust, outdoor-ready autonomous vehicle, the X-Maxx is unmatched in power, performance, and versatility.

3.1.2 - Part Selection

Key Observation and Comparison Table

- HSP Models are affordable and offer good customization options but might not hold up to heavy outdoor testing.
- Kyosho Models provide a balance of power and affordability but have limitations with sensor space and require better power management.
- Tamiya TT-02 is great for indoor experiments and beginner AI projects but may struggle with outdoor terrain.
- Traxxas Slash 4x4 Ultimate is ideal for high-performance projects, especially in competitions, but comes at a higher cost.
- LaTrax Rally is perfect for small-scale indoor projects but lacks the power needed for demanding algorithms and environments.
- Traxxas X-Maxx offers unmatched performance and durability for outdoor environments but is a significant investment in terms of both cost and weight.

Below is a table that summarizes the key aspects of each chassis.

Table 3.1.2.1: Chassis Key Aspects

Model	Price (USD)	Size (L x W x H)	Weight	Manufacturer
HSP 1/8th 94996	$460	20.5" x 15" x 9.8"	~3.5 kg	Hspacing Co., LTD
HSP 1/10th 94118 PRO	$300	18.7" x 9.9" x 6.5"	~3 kg	Hspacing Co., LTD
Kyosho 1:8 Buggy	$600	19.7" x 12.2" x 7.9"	~3.8 kg	Kyosho
Kyosho 1:10 Fazer Mk2	$300	14.6" x 7.9" x 5.1"	~2.5 kg	Kyosho
Tamiya TT-02	$150	16.7" x 7.3" x 5.8"	~1.5 kg	Tamiya
Traxxas Slash 4x4 Ultimate	$550	22.4" x 11.7" x 7.6"	~2.9 kg	Traxxas
LaTrax Rally	$140	10" x 4.7" x 3.9"	~0.5 kg	Traxxas (LaTrax)
Traxxas X-Maxx	$1,000	30.7" x 21.3" x 13.8"	~8.6 kg	Traxxas

Below is a table that summarizes the key pros and cons of each chassis

Table 3.1.2.2: Chassis Pros and Cons

Model	Pros	Cons
HSP 1/8th 94996	Affordable compared to competitors, spacious chassis, brushless motor, community support via Donkey Car	Plastic parts may wear out, not as durable on rough terrain, may require motor and battery upgrades
HSP 1/10th 94118 PRO	Lightweight, budget-friendly, easier to modify, adequate motor for indoor tracks	Limited space for sensors and batteries, less durable for outdoor testing
Kyosho 1:8 Buggy	Powerful motor, good off-road capabilities, well-suited for outdoor testing	Expensive, requires more power management, repairs can be costly
Kyosho 1:10 Fazer Mk2	Affordable, reliable for indoor and smooth tracks, beginner-friendly	Limited off-road capabilities, motor may need upgrades for heavy sensors
Tamiya TT-02	Affordable, lightweight, compatible with Jetson Nano, easy to modify	Limited outdoor capabilities, brushed motor needs upgrades, small chassis for complex setups
Traxxas Slash 4x4 Ultimate	Rugged, powerful brushless motor, spacious chassis, proven in F1Tenth competitions	Expensive, repairs can be costly, batteries drain quickly with extra components
LaTrax Rally	Compact, affordable, easy for indoor testing, beginner-friendly	Limited space for sensors, not powerful enough for high-speed or off-road testing
Traxxas X-Maxx	Extremely durable, powerful motor, excellent off-road performance, spacious for complex setups	Very expensive, heavy and difficult to transport, batteries are costly and drain quickly

Our decision

We've decided to go with the Traxxas Slash 4x4 Ultimate for our small-scale autonomous vehicle project, and it's an excellent choice! This model is a proven platform for advanced robotics and autonomous driving, being widely used in prestigious competitions like F1Tenth and the MIT Racecar Project. Its combination of power, durability, and adaptability makes it the perfect foundation for a project that demands both performance and reliability.

Why We Picked the Traxxas Slash 4x4 Ultimate

The Traxxas Slash 4x4 Ultimate stands out as the most versatile option for our needs. Its rugged build and high ground clearance (2.8 inches) mean it can handle testing in a variety of environments, from indoor tracks to outdoor terrains like gravel, dirt, and asphalt. This adaptability is critical as we plan to test algorithms like Follow the Gap and advanced path-planning under realistic racing conditions.

Another key factor in our decision was the spacious chassis. It provides enough room to mount essential components such as a Jetson Nano or Xavier, LiDAR, camera, and IMU without worrying about space constraints or overheating due to cramped wiring. This is especially important for a project that involves integrating multiple sensors and computational units.

The Slash 4x4 also comes equipped with a Velineon 3500 brushless motor, which delivers plenty of power and speed for high-performance testing. This motor allows the vehicle to reach impressive speeds while maintaining control, making it ideal for testing autonomous driving algorithms that require rapid decision-making and responsiveness. Its all-wheel-drive system ensures stable and smooth handling, even at high speeds or on rough surfaces, which is crucial for maintaining accuracy and avoiding collisions.

We were also influenced by the extensive community and resources available for the Traxxas Slash 4x4 Ultimate. Its popularity in F1Tenth and other autonomous competitions means we'll have access to open-source software libraries, tutorials, and forums. This support can significantly reduce our development time and help us troubleshoot issues quickly. While the upfront cost of $550 is higher than some other models, the long-term benefits in terms of durability, performance, and adaptability make it a worthwhile investment.

Stock Specifications of the Traxxas Slash 4x4 Ultimate

- Dimensions:
    - Length: 22.4 inches
    - Width: 11.7 inches
    - Height: 7.6 inches
    - Ground Clearance: 2.8 inches
- Weight: Approximately 2.9 kg
- Motor: Velineon 3500 brushless motor, known for high power and efficiency
- Battery: 5000mAh 11.1V 3S LiPo battery, providing excellent runtime (Option selected for maximum efficiency since the battery will also power the computing unit)
- Drive System:
    - Fully independent 4-wheel drive (4WD)
    - Adjustable suspension and torque control

- Top Speed: Up to 60 mph with a 3S LiPo and proper gearing
- Chassis Design:
    - Reinforced plastic chassis with enough space for sensors and computing hardware
    - Modular design for easy access and modifications
- Durability Features:
    - Waterproof electronics for weather resistance
    - High-quality shocks and suspension for rough terrain

What This Means for Our Project

The Slash 4x4's specs align perfectly with our project requirements. Its powerful motor and stable 4WD system make it suitable for testing high-speed algorithms and dynamic obstacle avoidance. The spacious chassis ensures that we can mount all the sensors and computing units we need without compromising performance. Its rugged design means fewer repairs and less downtime, allowing us to focus on refining our algorithms rather than constantly fixing the vehicle.

In addition, the availability of ready-made open-source libraries and tools for autonomous development will save us significant time and effort. With its proven success in competitive settings, the Traxxas Slash 4x4 Ultimate provides a level of confidence and reliability that is hard to match.

In conclusion, the Traxxas Slash 4x4 Ultimate is more than just an RC car; it's a professional-grade platform that will enable us to develop and test cutting-edge autonomous vehicle technologies. It's fast, reliable, and adaptable, making it the ideal choice for our ambitious project.

3.2 - Vehicle Mechanical Systems

For the vehicle mechanical systems, our overall goal is to maximize the performance of the two 1/10th vehicles through component additions and optimizations. The primary focus will be on the drivetrain, weight, and weight distribution, and the suspension systems of the cars. Through component additions and optimizations, we aim to improve the vehicles' acceleration in a straight line, strategically reduce the weight of the vehicle to increase their power to weight ratio, and evenly distribute the vehicle's weight while managing its placement to improve handling through corners and reduce the vehicles' body roll. The additions and optimizations would increase the base performance of the vehicles and provide the autonomy system with a good handling and stable vehicle.

Specifications we have determined would result in accomplishing the aforementioned goals are specified in our key specifications table, and the following research was conducted to find information on the best ways to achieve them. Generally, the approaches and technologies were compared and then selected based on their relative ability to lead to the determined specifications, although each aspect contained more specific selection criteria.

3.2.1 - Vehicle Drivetrain

The vehicle drivetrain plays a critical role in determining the performance of the vehicle, particularly in terms of acceleration and speed. For the second vehicle, the focus is on the optimization of drivetrain components to improve the vehicle's acceleration times. Improving the acceleration of the vehicle over the range of its expected operation speeds will improve the overall performance of the vehicle.

3.2.1.1 - Part Comparison

Table 3.2.1.1: Vehicle Performance Specifications

Parameter	Specification
0 - 14.5 mph time	> 5% decrease

The key drivetrain parameter for our vehicle is a 0 to 15 mph time decrease of at least 5% of the base vehicle's 0 to 14.5 mph time. This parameter is related to the acceleration of the vehicle, and efforts to decrease the time are efforts to improve the acceleration of the vehicle. Research into similar vehicles, at track layouts similar to what we expect to run the cars, shows that the car's speed ranges between 3 meters per second and 10 meters per second (roughly 6.7 mph to 22.3 mph). The average speed range, 6.5 meters per second or 14.5 mph is therefore a reasonable top speed to expect from the car at small-scale tracks.

The drivetrain modifications will focus on selecting a more powerful motor for the cars and selecting a mechanism for a belt drive extension that will adapt the new motor to the vehicles' gearbox and manage the angular velocity of the motor. The relation between motor power, torque, and angular velocity is given by (1): Where power (P) is directly proportional to torque (τ) and angular velocity (ω). This relation defines that given a constant motor angular velocity, as power increases, torque increases [52]. Additionally, the linear acceleration (a) of the car is related to the torque of the drivetrain as in (2). As the torque of the drivetrain is increased - given a constant tire radius (r) and vehicle mass (m) - the linear acceleration will increase [53]. Given ideal conditions (No traction loss, negligible drivetrain inefficiencies, and negligible aerodynamic drag), if the power of the motor increased by at least 5%, the linear acceleration of the vehicle should also increase by at least 5%. A factor of safety of 2 will be applied in motor selection to account for the unideal conditions the motor will operate in; therefore, a motor with at least 10% more power will be a selection criterion for the motor upgrade.

$$P = \tau \cdot \omega$$

Equation 1: Power, Torque, Angular Velocity

$$a = \frac{\tau \cdot R}{r \cdot m}$$

Equation 2: Linear Acceleration

The belt drive adapter for the new motor will be designed to handle the maximum load that the motor and setup can impart on the adapter. The belt drive gears and belt should be able to handle the maximum stresses produced by the motor and be sized appropriately to

deliver the 10% increase in power to the vehicle’s gearbox.

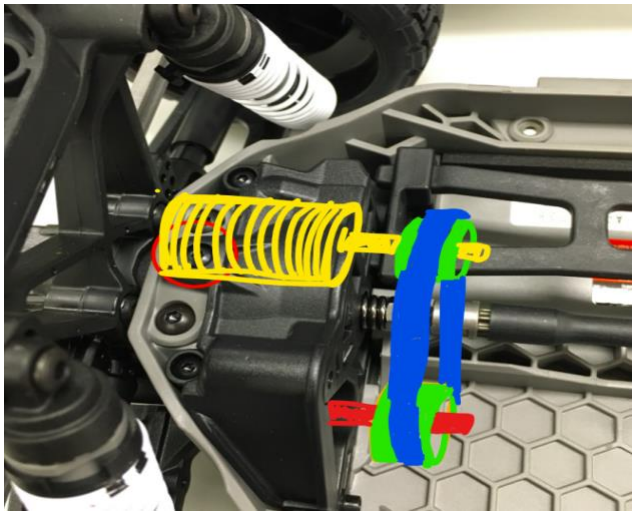


Figure 3.2.1.1: Concept illustration of the belt drive adapter for the new motor

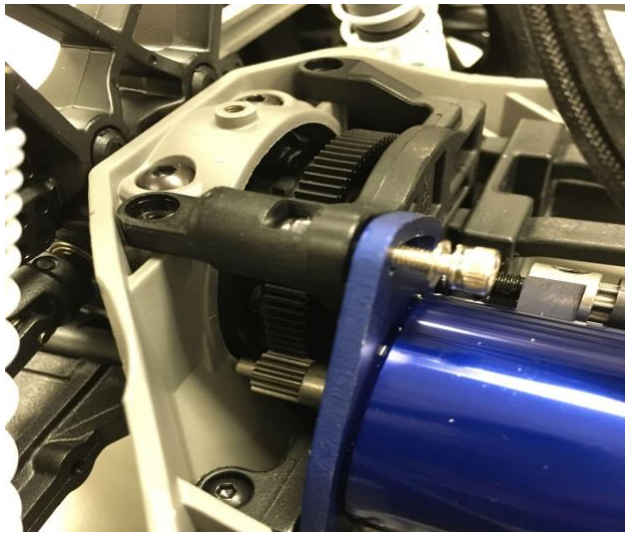


Figure 3.2.1.2: Assembly of the vehicles’ stock motor

The gears of the belt drive adapter will be sized in order to maintain/ keep constant the gear ratio of the stock motor and wheels. As in (2), modifying the gear ratio (R) between the motor and wheels will have a proportional effect on the linear acceleration of the car. In an effort to keep the power - therefore torque - the main independent variable of the drivetrain upgrade, the gear ratio between the new motor and the point/ gear that the old motor’s pinion gear interfaced will be 1:1. This will ensure that the overall gear ratio of the drivetrain remains unchanged, preserving the original relationship between the motor's rotational speed and the wheels' rotational speed. By keeping the gear ratio constant, any changes in the car's performance will be solely attributed to the increased power and torque of the new motor instead of changes in the mechanical advantage provided by the drivetrain.

Options for the new motor are the Surpass Hobby 3650, 4350 Kv brushless motor, KingVal 3670, 2650Kv brushless motor, and the KingVal 3650, 4300Kv brushless motor. Each one of these motors has a power rating greater than the stock Velineon 3500Kv motor of 721.5 Watts. Additional specifications can be found in Table 3.2.1.1.2.

Table 3.2.1.2: Motor Comparison

Motor	Power rating (Watts)	Cost (Dollars)	Source	Compatible with available motor controller	
				< 80A Continuous Current	11.1v to 60v
A. Surpass Hobby 3650, 4350 Kv	900	$71.89	Amazon	Yes	
				Yes	Yes
B. KingVal 3670, 2650Kv	1600	$37.68	Amazon	No	
				No	Yes
C. KingVal 3650, 4300Kv	900	$53.68	Amazon	Yes	
				Yes	Yes

All candidate motors have a power rating that is greater than 10 percent of the Velineon's 721.5 Watts. In fact, the lowest power rating is roughly 24.74 percent greater than the power rating of Velineon's power rating. Theoretically, and in ideal conditions, when incorporated into the drivetrain of the vehicle, the acceleration of the vehicle should see an improvement of 24.74 percent. The cost of the motors is an important factor and will be considered in selection. Although the KingVal 3670, 2750 is the cheapest, it is not compatible with the available motor controller. The KingVal 3650, 4300Kv is the second cheapest of the three, and it is compatible with the available motor controller.

The V-belt, flat belt, and timing belt are three different types of belt drive mechanisms and are the candidates for the vehicles' belt drive adapter.

V-belts are flexible belts with a trapezoidal cross-section. The trapezoidal shape of v-belts allow them to fit into a wedge shaped pulley with a tight fit. This fit and the high contact surface area between the belt and the pulley allows v-belts to have decreased surface shear stress as the load is distributed over the larger surface area. With decreased surface shear stress, the belt is less prone to slipping while transmitting load. Additionally, the v-shape

of the v-belt allows the belt to transmit the load on the belt perpendicularly to the pulley structure. This allows the belt to transmit higher loads as the pulley's material structure aids in carrying the load.

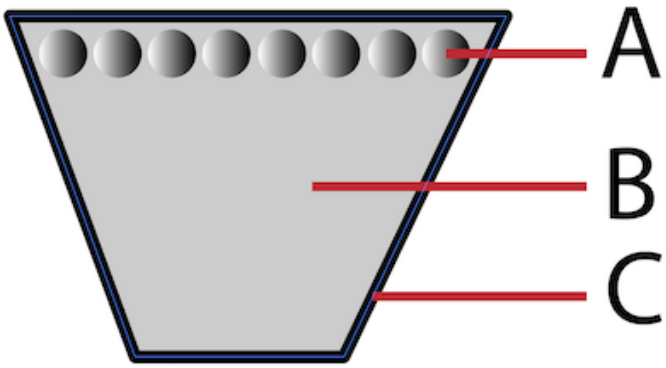


Figure 3.2.1.3: V-belt cross-section. Source: [59]

Flat belts are known for their ability to transmit large amounts of power/ load over larger distances, and can even be said to have no limit to the distance between the pulleys, but are not optimal for power transmission over short distances as are v-belts [60]. Additionally, in order to reduce slip, flat belts require operation at slower speeds.

Timing belts are similar to flat belts, but have teeth that mesh with the teeth on the pulleys that they drive. The toothed aspects of timing belts give them a few key advantages: They have no slippage, are highly efficient (up to 98 percent), produce minimal vibrations, and can maintain a constant velocity ratio. These characteristics make timing belts great for applications where reliability and predictability are optimal. Other characteristics of timing belts are that they are often more expensive, ideal for transferring relatively low power, and transmission of power at small distances [61].

In Table 3.2.1.3, the advantages and disadvantages of each type of belt drive mechanism are listed [54].

Table 3.2.1.3: Belt-drive Mechanism Advantages and Disadvantages

Belt-drive Mechanism	Advantages	Disadvantages
1. V-belt	- Transfers power well over small distances - Low slippage - Can transfer high power at high speeds	- High cost - Cannot transfer power at long distances
2. Flat belt	- Low cost - Transfer power well at large distances	- Not suitable for small distances - Tends to slip

	- Can withstand high loads	- Low mechanical efficiency
3. Timing belt	- No slippage - High mechanical efficiency	- High cost - Require initial alignment

Relatively high load-carrying capacity, high efficiency, and low to no slippage all over a short distance are the main concerns of the belt-drive design. The high load-carrying capacity will enable the drive mechanism to transfer loads from the motor to the vehicle's gearbox without failing. A high-efficiency belt drive mechanism will ensure that minimal energy is lost during load transfer - maximizing the power transferred to the wheels and therefore maximizing the vehicles' acceleration. Low/minimal slippage is important as any slippage is a waste of energy as the motor's torque is not being utilized to move the vehicle, but rather to spin the belt.

3.2.1.2 - Part Selection

In selecting the new motor for the vehicles, compatibility with the existing motor controller, cost, and power were considered and their importance was assigned in that order. A motor that is compatible with the existing motor controller, has the lowest cost and has the highest power is the most favorable. Table 3.2.1.4 ranks the motors based on this criterion where A, B, and C represent the Surpass Hobby 3650, 4350 Kv brushless motor, KingVal 3670, 2650Kv brushless motor, and the KingVal 3650, 4300Kv brushless motor, respectively. Since motor B is not compatible with the available motor controller, it was automatically eliminated from the selection list.

Table 3.2.1.4: Motor Scoring Based on Compatibility, Cost, and Power

Score	Compatibility	Cost	Power
2	A, C	C	A, C
1		A	

Motor A, the Surpass Hobby 3650, 4350 Kv brushless motor, got a score of 5. Motor C, the KingVal 3650, 4300Kv brushless motor, got a score of 6. With a higher score, the KingVal 4300Kv motor is the selected motor for the drivetrain modifications.

The ideal belt-drive mechanism as described in the previous section has a relatively high load-carrying capacity, has high efficiency, and exhibits no slippage over a short distance. The flat belt exhibits only a high load-carrying capacity. The V-belt and timing belt drive mechanisms are almost equal, the major and relevant difference being the timing belt mechanism has no slip whereas the V-belt has some slip. Since the ideal mechanism has no slip, the timing belt mechanism is the most ideal mechanism for the drivetrain modifications.

3.2.2 - Vehicle Weight and Weight Distribution

Vehicle weight and its distribution significantly impact overall performance, affecting acceleration, handling, and energy efficiency. We aimed to optimize the weight and weight distribution of the vehicle to achieve an improved power-to-weight ratio and static weight distribution target.

3.2.2.1 - Technology Comparison

The baseline power-to-weight ratio of the 1/10th scale vehicles was found through the calculation to be ~316.447 W/kg given the stock 721.5W motor and a weight of 2.28kg [26]. More weight will be added to the vehicles in the form of the hardware components used for autonomous navigation and the mounting solutions for those components. A weight budget of 3.42kg was estimated and added; This brought the total expected weight of the vehicle to 5.7kg and the expected power-to-weight ratio to 126.579W/kg. Through weight reduction, this power-to-weight ratio can be improved upon on the two vehicles.

Table 3.2.2.1: Vehicle weight and weight distribution specifications

Parameter	Specification
Power-to-weight ratio	> 126.579 W/kg
Static weight distribution (front to rear)	50/50

Generally, a higher power-to-weight ratio means better acceleration [27]. This is demonstrated in Newton's second law (3).

$$a = F/m$$ *Equation 3: Newton's Second Law of Motion*

$$F = P/V$$ *Equation 4: Force, Power, Velocity*

In (4) the vehicle's net force (F) is directly proportional to the vehicle's power (P). According to Newton's second law, acceleration (a) is directly proportional to the vehicle's force and inversely proportional to the vehicle's mass (m). Therefore, as the power increases and the weight decreases, the acceleration increases. For our vehicle, our power will remain constant while weight will be optimized. Optimization of the base weight of 5.7kg will therefore result in a higher power-to-weight ratio.

Oversteer occurs when the rear tires of a car lose traction, and understeer occurs when the front tires lose traction. As a car accelerates, weight is transferred to the rear, which reduces traction for the front tires, resulting in understeer. As the car decelerates, weight shifts to the front, reducing traction for the rear tires, leading to oversteer. If the weight distribution is biased, these effects can be made worse, as more weight will be transferred during acceleration and deceleration. An even load distribution between each tire ensures that the

majority of weight distribution results only from acceleration and deceleration [28]. This is why we specified a static load distribution of 50/50 (front to rear). Balancing the car's weight between the front and rear wheels can be accomplished by shifting mass or by utilizing materials of differing densities, achieving the desired effect without needing to physically relocate components. Minimizing unwanted oversteer and understeer will ensure that the car will have good handling through corners.

Weight optimization will be achieved through the use of lightweight materials for the 3D printing/ manufacturing of component mounting solutions. In general, we are looking for 3D printing materials with a low density. Other characteristics we are looking for are ultimate tensile strength, coefficient of thermal expansion, cost/kg, and printability or ease of use. If two materials are good candidates based on all the above properties and characteristics, and they have differing densities, even weight distribution can be achieved through the strategic use of both materials. Table 3.2.2.2 details the material characteristics of 7 common 3D-printing materials [29].

Table 3.2.2.2: 3D-printing material properties and characteristics

Material	Density (g/cm3)	Ultimate Tensile Strength (Mpa)	Coefficient of Thermal Expansion (µm/m°C)	Average Cost/kg ($/kg)	Printability/ Ease of use
PLA	1.24	65	68	25	9/10
ABS	1.04	40	90	25	8/10
Polypropylene (PP)	0.9	32	150	90	4/10
ASA	1.05	55	98	39	7/10
HIPS	1.03	32	80	28	6/10
Flexible (TPU)	1.21	23.8	157	50	6/10
PETG	1.27	53	60	40	9/10

3.2.2.2 - Technology Selection

Relative cost, printability, material density, material ultimate tensile strength, and the material coefficient of thermal expansion are key 3D printing material characteristics we considered during selection. We aim to minimize the cost of the 3D-printed parts. For 3D-printing, this means selecting a 3D-printing filament that is low cost and would not require extra materials or equipment to use. We assigned a higher score to materials that had a low relative cost and were known to be easy to print/ have a high printability rating. To achieve our lightweight requirements to improve the vehicle's power-to-weight ratio, we gave higher selection scores to materials with a low density. The material's strength is also an

important factor as it needs to withstand stresses induced by loads on a component. For this reason, we preferred materials with a higher ultimate tensile strength. To account for any higher than nominal (around the motors, motor controller, or other electronics that release a lot of heat) or unexpected temperature loading on the printed components, we gave higher scores to materials that would be able to withstand high temperatures without significant expansion or deformation. Such materials have a low coefficient of thermal expansion.

A ranking table, Table 3.2.2.3, was created to score the materials and determine the best material for our use.

Table 3.2.2.3: Ranking of 3D-printing materials

Score	Cost	Printability	Density	Strength	Thermal
6	PLA, ABS	PLA, PETG	PP	PLA	PETG
5	HIPS	ABS	HIPS	ASA	PLA
4	ASA	ASA	ABS	PETG	HIPS
3	PETG	HIPS, TPU	ASA	ABS	ABS
2	TPU	PP	TPU	PP, HIPS	ASA
1	PP		PLA	TPU	PP
0			PETG		TPU

Polylactic Acid (PLA) material had the highest combined score of 24. In comparison, Acrylonitrile Butadiene Styrene (ABS) scored 21, while Polypropylene (PP) followed with a score of 12. Acrylonitrile Styrene Acrylate (ASA) achieved a score of 18, and High Impact Polystyrene (HIPS) scored 19. Thermoplastic Polyurethane (TPU) had a lower score of 8, and Polyethylene Terephthalate Glycol (PETG) scored 19.

Generally, PLA and ABS are strong 3D-printing material candidates for use in our vehicles. Using both materials can also be beneficial to meeting our static load distribution specification. Varying the type of material used for a mounting solution at specific locations on the vehicle can be used to meet the 50/50 (front to rear) weight distribution specification.

In order to meet our power-to-weight ratio and weight distribution specifications, the total weight of the vehicles will be optimized through extensive use of PLA and ABS as the 3D-printing material for hardware mounting solutions. Varying where PLA and ABS are used will be beneficial to balancing the weight distribution of the cars. Additionally, the low density of ABS will be beneficial to keeping the vehicle's overall weight low.

3.2.3 - Vehicle Suspension System

The suspension system is an important vehicle system. A good suspension system or setup improves the vehicle's handling and stability - especially through corners. We aimed to optimize the suspension system to ensure effective weight transfer, enhance grip, and maintain stability during cornering.

3.2.3.1 - Technology Comparison

For racing vehicles, the roll angle is an important value that can help optimize vehicle handling. Typically, vehicles designed for racing (sports cars and touring cars) tend to have lower roll gradients or roll angles per unit of lateral acceleration [30]. The goal for our vehicles is to achieve a significantly lower roll-angle through the use of roll-angle reduction mechanisms, which we will quantify as the mechanism's effectiveness. We aim to have a mechanism effectiveness of greater than 1.25. We conducted research on 3 common ways roll-angle can be controlled: The use of Anti-roll bars, modification of a vehicle's center of gravity, and selection of suspension spring rates.

Table 3.2.3.1: Vehicle suspension specifications

Parameter	Specification
Roll-angle reduction mechanism effectiveness	> 1.25

Table 3.2.3.2: Typical roll gradients of vehicles

Vehicle Type	Roll Gradient (°/g)
Mid-size car	5-6
Sports car	3
Indy Car (2001)	0.1 - 0.2
Formula 1 Car (2002)	0.03 - 0.1

Generally, the lower the roll angle of a vehicle, the better the vehicle handles in corners and the more stable the platform. Anti-roll bars, center of gravity, and suspension springs, spring rates can all have an effect on the roll angle of a vehicle. Criteria for roll-angle reduction mechanism selection for our vehicles are that the mechanism needs to be cost-effective, easy to implement, and easy to adjust for varying results.

Anti-roll bars (Also known as sway bars) are an obvious method for controlling vehicle roll. Anti-roll bars are torsional springs that are attached to the front and rear of a vehicle and serve to allow the modification of vehicle grip levels between the front and rear/ the grip balance or balance of the vehicle.

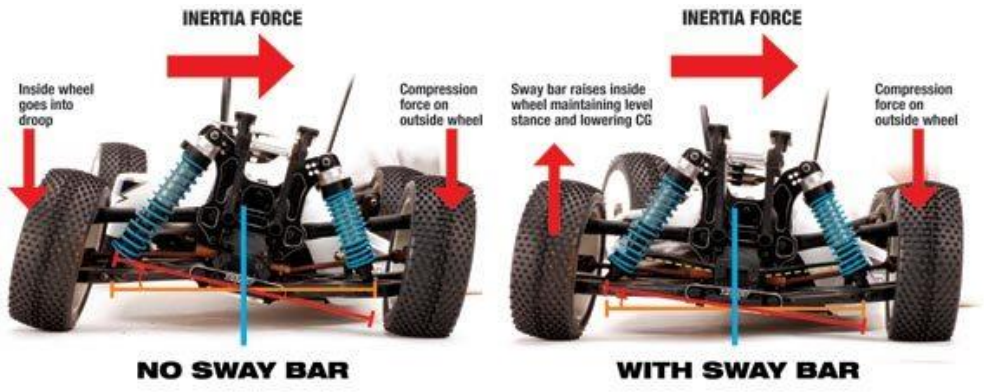


Figure 3.2.3.1: Illustration of Anti-roll bar effects on an RC car. Source: [31]

When the balance of the vehicle is more towards the front of the vehicle, the vehicle will likely exhibit oversteer. As mentioned in the weight and weight distribution research, oversteer occurs when the rear tires of a car lose traction. The loss of traction is caused by the rear wheels not having enough grip compared to the front tires. Conversely, when the balance of the vehicle is more towards the rear of the vehicle, the vehicle will likely exhibit understeer. Understeer occurs when the front tires lose traction. The loss of traction is caused by the front wheels not having enough grip compared to the rear tires.

Generally, a well-handling vehicle has an even balance. Anti-roll bars are used to tune the balance of the vehicle by modifying the stiffening of the front and rear suspensions and therefore controlling the roll angle of the vehicle. For both the front and rear suspensions, setting the bars to a stiffer setting will decrease the roll angle as is shown by the inverse proportional relationship between the roll angle (φ) and the roll stiffness due to anti-roll bars on the front and rear axle ($c_{Ro,S,f}, c_{Ro,S,r}$ - respectively) in equation [30, eq. 2.11].

On Amazon, an anti-roll bar kit for the Traxxas Slash 1/10th vehicle, made by 3rd parties goes for between $15 and $18 and the kits tend to include 3 settings of anti-roll bars with undocumented specifications. The bars can be easily attached to the vehicle using the provided mounting screws at locations designated/ pre-designed to hold the bars.

The center of gravity is also another aspect of a vehicle that can be modified to affect/ control the vehicle's roll angle. The center of gravity of a vehicle is the point/ location in a vehicle where the total weight of the vehicle is considered to be concentrated or the average location of the mass of a vehicle with respect to some reference frame [32]. Another important point/axis on a vehicle is the roll center. The roll center is the instantaneous pivot point around which a vehicle's body rotates [30]. In corners, inertial forces on the car will cause it to roll about the roll center.

The distance between the center of gravity of a vehicle and the roll center of a vehicle affects the roll angle of the vehicle. The roll angle is directly proportional to this distance between the center of gravity and the roll center [30, eq. 2.11]. From these relations, one can conclude that there are two ways to affect the roll angle of a vehicle by only varying the roll center and center of gravity location. Typically, the center of gravity of a vehicle is above the roll center of the vehicle; therefore, the two ways would be to raise the location

of the roll center of the vehicle or to lower the center of gravity of the vehicle.

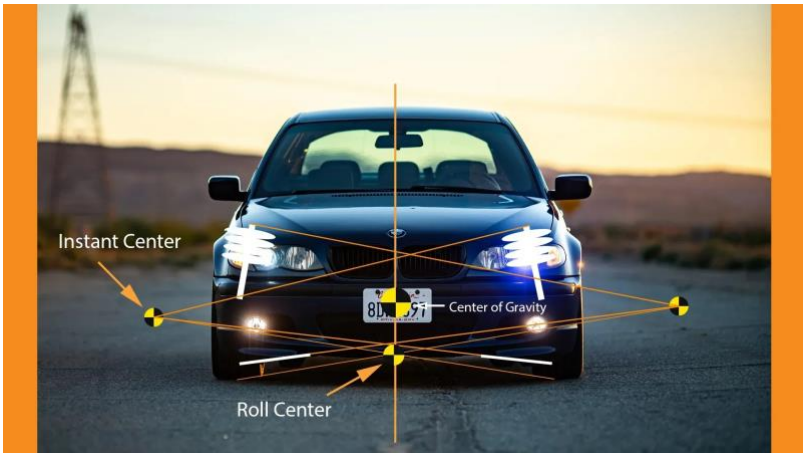


Figure 3.2.3.2: Roll center and center of gravity of a vehicle illustrated. Source: [33]

For our vehicle, modifying the position or height of the roll center is possible by adjusting the suspension camber links of the cars [57]. Although these adjustments are not continuous, the distance between the roll center and the center of gravity can be adjusted to affect change to the roll stiffness of the vehicle. Additionally, the roll center of the front and roll center of the rear axes of the vehicle can be adjusted independently. This adjustment can aid in further tuning the roll angle by allowing the roll stiffness due to the roll center position bias between the front and the rear.

Lowering the center of gravity of the vehicles can be accomplished using a couple of obvious ways: Adding ballast weight low on the vehicle or placing components on the vehicle as low as possible. Additionally, these methods are relatively cheap and allow for easy roll angle target adjustments. Any object with the right weight could be added as ballast to a calculated position on the vehicle to lower the center of gravity of the vehicle.

Similarly, movable components like the batteries, autonomous navigation computers, and PCBs can be mounted at specific positions in order to achieve a desired center of gravity location. The most significant cost incurred from affecting roll angle in this manner would be the cost of the mounting/securing solutions - which is already accounted for and expected with regards to 3D printing mounting solutions for the vehicle components - and can be considered negligible.

The third mechanism that could be used to control the roll angle of a vehicle is suspension spring rates. The spring rate of the suspension is a quantity that defines the amount of load required to compress a spring by a unit length (Typically N/m or lb/in is the unit of measure for spring rate).

High-performance vehicles tend to have high spring rates to - among other reasons - minimize the vehicle's roll gradient/ roll angle as seen in the table below. It's important to note that spring rates are directly proportional to the square of the vehicle's sprung mass natural frequency [34] - The higher a vehicle's suspension natural frequency, the higher the vehicle's suspension spring rates.

Table 3.2.3.3: Typical sprung mass natural frequencies of vehicles. Source: [30]

Vehicle Type	Sprung Mass Natural Frequency (Hz)
Passenger car	1-2
NASCAR	1.5-4
GT3	2.8-4
Formula 1	3.5-7
IndyCar	5-7

Roll angle is inversely proportional to the roll stiffness of a vehicle as in [30, eq. 2.11]. Additionally, the roll stiffness is directly proportional to the spring rate of the vehicle's suspension [35]. Generally, as the spring rate increases, roll stiffness increases and roll angle decreases. Therefore, in order to effectively control the roll angle of the vehicle, we would need to change the spring rate of our vehicle's suspension system.

Changing the spring rate of the vehicle's suspension system would require replacing the existing springs and dampers already on the vehicle. For the Traxxas Slash 1/10th scale vehicle, the cost of replacing the shocks ranges from $9 to $25 on Amazon. Unfortunately, for such components, the specifications such as spring rate are not readily available. The rates could be calculated through testing but would not be adjustable in order to achieve different target roll angles.

Table 3.2.3.4: Roll angle control mechanism characteristics

Roll angle control mechanism	Average Cost (dollars, $)	Ease of integration (Low, Medium, High)	Easy of adjustment/ variability (Low, Medium, High)
A. Anti-roll bars/ sway bars	16.5	High	Medium
B. Lower center of gravity/ Raise roll center	0	Medium	High
C. Suspension spring rates	17	High	Low

3.2.3.2 - Technology Selection

In terms of average cost, our team values the lowest-cost solution. We also value a high

ease of integration and a high ease of adjustment/ variability solution.

Table 3.2.3.5: Scoring of the three roll angle control mechanisms

Score	Average Cost	Ease of integration	Easy of adjustment/ variability
3	B	A, C	B
2	A	B	A
1	C		C

When scored according to these values, the low center of gravity mechanism has a score of 8, the anti-roll bar mechanism has a score of 7, and the suspension spring rate mechanism has a score of 5. According to the research and our values, the low center of gravity mechanism would be the best solution to use in order to achieve our target roll reduction mechanism of > 1.25 through an iterative application and testing approach.

3.3 - System Status Indicator

Our team decided to also include an system status indicator module for the vehicle so that we could keep track of the status of the vehicle's battery life, bluetooth and wifi status, and speed while racing on the track. We want this to help simplify the process of autonomous vehicle management while the vehicle is on the course, so that the vehicle can be managed better than if we had to just depend on waiting for the battery to just die, or for the vehicle signal to stop working, we want to be ahead of the curve.

Ideally, we want to provide an intuitive experience for the user of this vehicle so that they can effortlessly enjoy driving the vehicle without worrying about whether their car has the ability to complete the track or not. In order to do so, we wanted to use a capacitive touch screen with a large screen space in order to display the various metrics of our vehicle diagnostics. While this is more of a challenge to implement, with very little time, we are also considering using a simple LED digital display on a small sized handheld device for the user.

Using a capacitive touch screen will require us to write a Graphical User Interface that displays all the information in an easy to read and intuitive manner, all while housing a case for our PCB containing the MCU for the touch screen, connections for the breakout boards, bluetooth and wifi connectivity to the vehicle, and various sensors. All of this considered with costs and time constraints to implement such a complex idea, we decided that the capacitive touch screen would still be a more feasible route. With that in mind, we still needed to find the appropriate digital display to support everything that we wanted to display on our screen, leading to considerations of how to choose a microcontroller.

Since a capacitive touch screen is being used, this adds another layer of software work for this project, creating a graphical user interface that both properly displays the necessary information and has a modern and intuitive design. Fortunately, there are many libraries

that can support this task and can even work alongside the functionality of the microcontroller's connectivity to the capacitive touch screen through one of its peripherals. The end goal of this project is to not only add to the already impressive work that is being done to create a fully autonomous small scale racing vehicle, but to provide a potential solution for any and all autonomous vehicle users that hope to make the maintenance and status indication of their vehicle less of a hassle.

Taking into consideration all of the requirements of the ECE department for the Senior Design project, we have decided to design a PCB that mounts all of our components, such as the Microcontroller, wifi transmission module, capacitive touch screen, and enclose it into a shell casing using Fusion 360 to wrap the entire PCB into a sleek design. We wanted the user to have an easy and unique experience with using the indicator subsystem to quickly display the status of the small scale autonomous racing vehicle, without the hassle of having to plug the computer of the vehicle into another computer during a potential competition to check the vehicle. Taking this into consideration, we want to make the device as lightweight, portable, durable, and sustainable as possible.

When doing research on the Microcontroller that we would like to use for this project, we had to take into consideration power/energy consumption of the board, connectivity to bluetooth and wifi, size of the board, architecture, use cases, development tools, peripherals, and costs. This created the scope of feasible microcontrollers down to four possible options; the MSP430FR, STM32 board, ESP 32 board, Arduino Mega board. We considered all aspects of the functionality of the system status indicator module, the financial, and user experience. The microcontroller research was the most vital for this project, and we believe we made the right decision by choosing to go with two ESP32 boards.

Beyond the Microcontroller, we also needed to choose the appropriate peripherals for the indicator subsystem to be able to display the following: the battery level of the board, battery level of the vehicle, speed, location of the vehicle, Bluetooth connectivity, wifi signal, odometer, and vehicle performance measuring system. We hope to use the wifi and bluetooth peripherals already embedded on the ESP32 boards, along with the external capacitive touch screen to minimize costs and maximize functionality of the system status indicator module with as little materials as possible. The user, in the worst case, should be able to replace and/or fix the system status indicator module if something were to break or the functionality of the board ceases.

The capacitive touch screen should be able to appropriately display all the details of the vehicle diagnostics while also allowing the user to touch the screen, looking at the diagnostics with more depth and making decisions about how to repair or improve the vehicle performance. Ideally the capacitive touch screen should not consume too much power from the indicator module’s onboard battery, as the microcontroller may also potentially need to use a lot of power due to the amount of peripherals on board. Connecting the capacitive touch screen should be relatively simple, and adding a modern and user friendly graphical user interface with the abundance of GUI libraries that are available to the microcontroller world should not be an issue.

There are a variety of GUI options to create user-friendly and intuitive designs such as

LVGL (Light and Versatile Graphics Library), TFT_eSPI, GUIslice, and uGFX(microGFX). All of these libraries are commendable options for most microcontrollers since they are generally lightweight, contain a variety of graphical components and features to create a functional diagnostics display. Ideally, the framework cannot consume too much space since the microcontroller still needs room for wifi, bluetooth, battery level, vehicle tracking, and speed measuring diagnostics.

3.3.1 - Microcontroller Comparison

Let's look at in detail and compare the four boards we considered, and see why the board we selected is the best board for our project. Below are the block diagrams and schematics for each respective microcontroller that was and is being considered for the functionality of the indicator module. We need the microcontroller to have good processing speed, Wireless connection and bluetooth connectivity, relatively moderate to high power consumption is a reality, some basic to advanced peripherals, and ability to work with either the Arduino, Code Composer Studio, or some user friendly, commercial IDE. The cost is not much of a concern, as most microcontrollers are anywhere between $2 to $30 range.

When considering other aspects like power output, flexibility to work with various different peripherals required to work with this project, we can see that the ESP32 provides the best overall fit for our project needs. Below are the various boards with their respective technical performance details and why each board was considered when conducting this research. While all of the boards provided many great features that could also work with this project's needs, it was decided that the most important aspects came down to ease of use for the user, and the board's ability to work with a variety of functions on demand without requiring external supplies or support.

MSP430FR2676

The MSP430FR2676 is a microcontroller manufactured by Texas Instruments, created with robust functionality with the ability to connect to many various breakout boards, a capacitive touch screen option, and various peripherals. It is an ultra-low powered microcontroller, ideal for lower energy consumption requirements, while also boasting multiple deep sleep modes. Considering having worked with this board in the past, it has a user-friendly IDE and is very simple to set up the functionality of the watchdog timers and pin ports. Considering that the status indicator module will need at least five to seven peripherals due to the functionality of the board, this microcontroller is in high consideration for use.

Unfortunately this board does not have the embedded features of bluetooth module and wifi connectivity like its other microcontroller competitors. Luckily it still has the ability to connect with external modules for wifi and bluetooth features, but then it will increase the costs of the project. It also has I2C, SPI and UART communication available to use. Overall, this board is a powerhouse of a microcontroller and a great option for all of the functionality that we would need to accomplish with our status indicator module.

The MSP430 has valuable features such as its Active mode, Low Power Modes (LPM0 to

LMP4), and Wake-Up Time. The Active mode operates the microcontroller on a very low power mode while it runs, a very unique feature in the world of microcontrollers. The Low Power Modes zero through four start progressively going into deeper states of sleep, as zero state disables the CPU while everything else continues to run. The fourth and last state disables everything on the microcontroller except the RAM, allowing the different components to shut down one by one in order to minimize power consumption at each level. Lastly the Wake-Up time allows the microcontroller system to completely wake up within the span of less than one microsecond from deep sleep state, and return back to sleep in nearly the same time to optimize energy performance.

The RISC Architecture makes it ideal for reducing large chunks of code down to simple instructions that can be executed within the span of a few lines, giving the microcontroller some of the best performance specs seen by any of its competitors. This pairs with the on board peripherals available such as having a whopping 80 Digital I/O pins with pull up or down resistors, interrupts, timers that can generate PWM signals, measure time intervals, and perform time based tasks.

The flexible clock and oscillator system that is found on board allows the microcontroller to run various clocks from various pinouts, making it the perfect match for the status indicator module. The digital Communication Interfaces such as the UART, SPI, I2C, IrDA, and Capacitive Touch Sensing are also included in the list of on-board peripherals. The UART acts as a serial communication interface for communicating with other computers and MCUs, the SPI aids as an interface for data transfers with sensors, memory devices and other external peripherals. The I2C helps to connect low-speed peripherals like temperature sensors and EEPROMS, while the Capacitive touch sensing, in some MSP430 models, allows touch screen capability on-board, ideal for a project like this.

STM32

The STM32 Nucleo board has more advanced processing power with an ARM cortex M4 processor on board. It is another solid option for a viable microcontroller considering all the peripherals that will be on the status indicator module. It has 76 GPIO pins, 3 LEDs, and available UART, SPI, and I2C, making it easier to communicate with other peripherals that are external to the board's embedded features.

The STM32 comes in multiple different series as well, catering to various kinds of low power modes, high performance, general purpose, and wireless bluetooth needs for each user. The clock speeds of this microcontroller also give the user a large range to choose from, making it perfect for optimizing performance, energy efficiency, and memory. This project does not necessarily require an ultra-low power mode, nor does it need to have super high performance, since it is just displaying real time data to the status indicator module. A valuable feature that it does contain is the ability to have an onboard Wi-Fi module, allowing the user to save costs on external peripherals when transmitting data to and from the microcontroller connected to the vehicle’s onboard computer.

Another useful feature that comes on the board is its on board peripherals such as GPIOs, timers, SPI, I2C, UART, CAN, USB, Ethernet, and ADCs. This means the board can support wireless functionality, bluetooth functionality, a graphical user interface for the

capacitive touch screen display that is used to display the vehicle diagnostics. The interfaces are also very energy efficient and ideal for designs that are battery powered, which, considering the vehicle is powered by batteries, makes it an ideal choice to increase the life of our battery and decrease the overall costs of our vehicle.

When it comes to developing software on the STM32, they are able to be flexible and scalable since they work with a variety of IDEs such as STM32Cube software suite, STM32CubeMX, and many others like Keil, IAR, and Code Composer Studio. If two or more of the STM32 microcontrollers are being used, they actually become extremely scalable since they offer pin to pin compatibility between each other and don't require any redesigning of the microcontrollers to communicate effectively. While communicating, the microcontrollers also come with security features like secure boot, hardware cryptographic modules, and memory protection, which makes it a great fit for IoT applications where secure communication is extremely important.

Arduino Uno

This microcontroller is also another great option, boasting ease of assembly for users of any level and also using a variety of high level peripherals to aid in the functionality of the status indicator module. It is programmable through the Arduino IDE using C or C++, having a very simple, and extensive library options which can aid in a variety of projects. It is one of the most popular boards to use for many microcontroller based electronics projects for that reason, and with an 8-bit AVR microcontroller, it is suitable for many basic and mid level embedded applications.

While this is generally a great thing, for the status indicator module, it requires a bit more processing power to handle all of the peripherals that will be both communicating via bluetooth with the vehicle, and also displaying real time data to a screen. The clock speed of the microcontrollers' ATmega328P runs at 16 MHz, and flash memory of 32KB, SRAM of 2KB, and EEPROM of 1 KB which is useful for storing static data. The Digital and Analog I/O also presents another potential issue for this indicator board, as it only boasts around 14 digital pins, and 6 analog pins, much less compared to its competitors.

The Arduino Uno does not consume a lot of power, making it great for having many peripherals that need to be accounted for when designing an optimal custom PCB for the functionality of the status indicator module. Its low power consumption also means less costs need to be spent on managing batteries or charging costs.

ESP32

A small powerful, low cost microcontroller, that has many unique embedded features including bluetooth and wifi connectivity. Usually it is used for IoT projects, or projects in which one or more robust microcontrollers may be needed to control multiple things at once. Very useful for small electronic projects, or projects where low power consumption is needed. It has varying space options and is compatible with a variety of different peripherals.

Considering all the very ideal and useful features above, it is the most ideal for IoT projects due to its built in bluetooth and wifi modules, which none of the other microcontrollers

have embedded within, one needs to have external modules to have that functionality. This will affect the costs of not only the board itself, but also the creation of the custom PCB, saving costs and size of the board, leading to a potentially lighter and more sleek design for the status indicator module. The ESP32 unfortunately requires a lot of power consumption due to its smaller form factor, meaning that it will require another feasible power solution, whether through Li-Po battery or an external battery module for ease of use.

Overall this microcontroller is a great option from a user experience, costs, ease of development, and functionality perspective. Many of the other boards also, due to a lack of having a built in bluetooth module, create more of a challenge of connecting to the vehicle's computer to help the status indicator module receive real time data to track the vehicle's status. The ESP32 has the ability to connect with another ESP32 microcontroller and send data back and forth to and from each other, to gather real time data from the sensors collecting data on the vehicle.

Comparison Table

Below we have a comparison table for all of the microcontrollers that have been considered as feasible options for this project. Considering all of the comparison data, the ESP32 is the most optimal option considering its low cost, clock speed, wifi, bluetooth, pin count, and IDE, making it the best combination of traits that the status indicator module would need to efficiently function for our customer. The one factor that makes it the most optimal is its ability to communicate with other ESP32 microcontrollers with its built in bluetooth connectivity, something that none of the other boards did not have, which can save costs of external sensors needed, and also costs for manufacturing the final custom PCB for the status indicator module.

There are other boards that can be considered for their high performance for capacitive touchscreen support such as the Raspberry Pi 4, but it is priced at $60 on most marketplaces for the 4GB model, which would be the ideal amount of RAM for the status indicator module.

Table 3.3.1.1: Comparison of the most relevant Microcontrollers.

Feature	**MSP430FR2676**	**STM32**	**ESP32**	**Arduino Uno**
Core Architecture	16-bit RISC	32-bit ARM Cortex-M	32-bit Xtensa LX6 (dual-core)	8-bit AVR
Clock Speed	16 MHz to 25 MHz	32 MHz to 480 MHz	Up to 240 MHz	16 MHz
Flash Memory	1KB to 512 KB	64KB to 2MB	520 KB SRAM, 4MB Flash	32KB
RAM	128 B to 66 KB	16 KB to 512 KB	520 KB SRAM	2 KB
Power Efficiency	Ultra-low power,	Low to moderate	Relatively high	Low power, but

	often in the range of microamps in low-power modes	power, depending on model and features	power consumption due to the wifi and Bluetooth functionalities	not as efficient as MSP430
Wi-Fi	Not available	External modules required	Built-in Wi-Fi	External modules required
Bluetooth	Not available	External modules required	Built-in Bluetooth	External modules required
UART/SPI/I2C	Available	Available	Available	Available
GPIO pins	10 to 80 pins	30 to 100+	34 (digital) 16 (analog-capable ADC)	14 (digital), 6 analog (ADC)
Analog Input	8 to 24 channels	Up to 24 Channels	18 channels (12-bit ADC)	6 channels (10-bit ADC)
IDE	Code Composer Code (TI)	STM32CubeIDE, Keil, IAR	ESP-IDF, Arduino IDE, PlatformIO	Arduino IDE
Programming Languages	C/C++	C/C++, Assembly	C/C++, Python (MicroPython), Arduino	Arduino C, C++
Sleep Modes	Multiple deep sleep modes with uA-level current draw	Low-power sleep modes available	Sleep and deep sleep modes, but higher power consumption due to peripherals	Basic sleep mode available
Price Range ($USD)	$2-$10+	$2 to $20+	$4 to $10+	$5 to $30
Applications	Low-power applications, sensors, wearables, medical devices	Industrial, real-time control, robotics, consumer electronics	IoT applications, home automation, wearables, wireless communication	Education, prototyping, basic DIY electronics

Below is a scoring table that was used to help determine why the ESP32 was chosen as the best board for this project, considering multiple parameters.

Microcontroller Selection

Table 3.3.1.2: Scoring table for determining final Microcontroller.

Score	Costs ($USD)	Feasibility	Functionality	Power Efficiency
3	ESP	ESP, Arduino	ESP, MSP, STM	MSP
2	STM, MSP	MSP		Arduino, STM
1	Arduino	STM	Arduino	ESP

As made clear by the scoring table above, the ESP32 comes out on top as the best option, the microcontroller best suited for this project's needs. While the MSP430 presents itself as a very competitive option, it does not have on board bluetooth and Wifi, which ultimately increases the costs of the project.

3.5.4 - Indicator Subsystem Display Options

3.5.4.1 - Capacitive Touch Screen Display

There are two capacitive touch screen options being considered, the raspberry pi 4 and the ESP-WROOM-32 capacitive touch screens. While both are great options, they have some characteristics that are specifically well suited for their respective associated Microcontroller models, such as the ESP32 and the Raspberry Pi 4. Both displays are perfectly capable of working with any of the given microcontroller options above, but since the Raspberry Pi 4 7-inch display and 10.1-inch display have arguably better graphics and better sizes than the ESP32s 2.4 to 3.5-inch screens, it is safe to say that the Raspberry Pi-4 screen options are the better choice. The screen needs to be compatible with the microcontroller of our choice, either the Raspberry Pi 4, or the ESP32, which according the display documentation, the Raspberry Pi 4 consumes more power but provides us with better graphics library support for more GUI variation and functionality on a large real-estate board.

3.5.4.2 - LED Display

An LED Display is also another feasible option, since it can display all the information needed on the system status indicator board. A graphical user interface will need to be developed with this option as well, just like the capacitive touch screen display. The major difference between this and the capacitive touch screen option is that it cannot be used without an external controller, like a mouse or keyboard, to respond to user input, which increases the overall costs.

LED displays also have very good image quality, which although is a nice feature to have for any display, this project does not necessarily require the best image quality. With increased image quality, the costs of the system status indicator subsystem will likely increase. This project calls for a more durable, practical display solution that can display all the system statuses.

Table 3.5.4.2.1: Characteristics of LED and Capacitive touch screen displays.

Features:	**Capacitive touch screen**	**LED display**

Costs	Varies	Varies + external input system costs
Size	Varies	Varies
Touch Screen	Available	Not Available
purpose	Displays and has touch input capability	Displays images
Durability	Mostly durable through most conditions, except performance is impacted when there is high moisture or when gloves are used.	Very durable through most conditions, can be suspect to burn in if images are displayed for extended periods of time.
Brightness and visibilty	Great for indoor and outdoor use.	Great for indoor and outdoor use.

Below is a scoring table to determine which option better serves the desired outcome of this project.

3.5.4.3 - Display Selection

Table 3.5.4.3.1: Selection of the system status display.

Score	**Costs ($USD)**	**Ease of Use**	**Ease of Assembly (Development time)**
3	LED, CAP	CAP	CAP
2		LED	
1			LED
0			

The conclusion is that the capacitive touch screen will be used for this project, as it allows the user to change pages, scroll, check the system status of the vehicle more intuitively than the LED Display would allow. It is also more intuitive for the user, and has a faster and more feasible development time, since many libraries and development tools have been developed to create graphical user interfaces without much code development. Having a capacitive touch screen saves more development costs and time, while also being easier for the user to use.

3.5.5 - PCB design

With regards to the design of our PCB, we needed to design the proper schematics for the design, considering the MCU we selected, the breakout boards needed to create the proper

functionality of the board to work with both the car and the capacitive touch screen, and to fulfill our senior design requirement of doing work on a custom PCB. The software used to design the custom PCB will either be created using Fusion 360, Eagle, or KiCad. Essentially, all of the following software could work optimally for creating a custom PCB for the status indicator module, but some do not work on a Apple macbook pro computer such as KiCad, which presents an issue in development time since a windows computer would need to be outsourced for development, increasing the costs of this status indicator module development, and decreasing available time to create the custom PCB.

3.5.6 - Indicator Subsystem peripherals

For the System Indicator subsystem, we have a variety of peripherals that will be used to implement the main features, such as gauge the autonomous vehicle battery and check the connectivity of the indicator subsystem and vehicle. The system indicator should also be able to display all of these features on a capacitive touch screen display, so that the user can touch the buttons on the screen to display details on the system status for the vehicle. In order to actually implement most of these peripherals, the following categories of peripherals must be assessed. There are Battery, Battery Gauge, Wifi and Bluetooth, Graphical User Interface (The screen display), Speedometer, and a Capacitive touch screen, all enclosed within a hard casing for the components to stay within. In order to integrate all of the peripherals, Fusion 360 has been chosen as the best software to create the chassis of the system status indicator casing, the PCB, and integration with the capacitive touch display. Many of these peripherals have been chosen on the basis of reducing power consumption, while increasing the efficiency and awareness of the hardware and software maintenance of the autonomous vehicle. For example, the battery gauge will be used to monitor the status of both the vehicle battery and the system status indicator battery, knowing this will help increase the lifespan of the car battery, and make the user more aware of when it is time to replace the battery or charge the vehicle, so they can prepare accordingly.

The Speed monitor will aid in understanding vehicle performance in real time, along with the battery gauge, which will give the user a better understanding of how the machine learning algorithm for the autonomous vehicle is performing. The battery temperature and voltage, also measured by the battery gauge, will give the user a better understanding of how energy efficient and healthy the vehicle is. The battery gauge is also equipped with a safety mechanism, able to give a warning notification to the user via the alert pin, if the battery reaches a critical level of discharge. The battery gauge aids in sustaining the lifespan and safety of the battery, making it an excellent peripheral to have on the indicator subsystem. The bluetooth module and LEDs that are integrated onto the system status indicator, are able to give instant feedback on the connectivity of the board to the vehicle's microcontroller, ensuring that it is enabled. Along with the bluetooth, the battery used by the indicator subsystem must be supported by a charger module and overcurrent protection to ensure the batteries are safe and rechargeable, allowing the system status indicator to operate for many hours without battery replacement. Below, the peripherals used for this indicator subsystem are explained in depth, and why each specific model for a peripheral was chosen over other robust models with similar technical specifications.

Battery gauge

For this component of the status indicator subsystem, we need to measure the battery of both the vehicle and the system display as well. In order to do this, we need a component called a battery gauge. It is an integrated circuit designed to monitor the state of a battery and provides critical information about the health and status of the battery, which enhances the overall performance of the vehicle. The main goal of being able to measure the battery in the vehicle from the system status indicator, is to increase the efficiency in long term vehicle performance, and decrease the costs of the vehicle maintenance. In order to select a good battery gauge, we consider the wiring, charging circuits, monitoring, and safety mechanisms involved for reliable functionality.

One of the most efficient and accurate battery gauges designed for lithium batteries is the MAX17261, which measures State of Charge, voltage, current, and temperature of the battery in real time. The Gauge needs a charging circuit such as a TP4056 or LiPo charger depending on the battery that is being used for the system status indicator and the autonomous vehicle. Just like any battery gauge, it must be connected directly across the battery terminals, while the charger must connect to the battery terminals in parallel. The MAX17261 uses the ModelGauge m5 algorithm to measure the voltage, State of Charge, and temperature, which can and will be recorded on the status indicator subsystem. It also uses I2C interface with the ESP32 and Raspberry Pi Zero w, so that data can be sent between the two boards, and displayed onto the capacitive touch screen. The MAX17261 also boasts safety mechanisms with alert capabilities and temperature monitoring, ensuring that the battery can sustain a longer lifespan. There is an integrated alert pin that allows the system to send a notification to the user if there is a low battery condition or faults in the battery, overall improving the safety of the vehicle and user. Another notable feature is the 2.5V to 4.5V voltage range, which is able to support single cell lithium ion batteries that are used for both the vehicle and the indicator board. This, mixed with the simple integration setup of the MAX17261, avoiding any complicated setup or calibration, makes this specific model of battery gauges an optimal choice. Below is a table showing the various other battery gauge options considered for the system status indicator that also were optimal, but failed to outduel the MAX17261 overall.

Table 3.5.6.1: Microcontroller Power related Features

	TI BQ27542-G1	MAX17261	LTC2943	INA219
Voltage Range	2.5V - 4.5V	2.5V - 4.5V	Up to 60V	0V - 26V
Current Monitoring	Yes	Yes	Yes	Yes
SoC Measurement	Yes	Yes	Yes	Indirect (via current reading)
Interface	I2C, HDQ	I2C	I2C	I2C

Power Consumption	Low (Standby mode)	< 5uA	Moderate (~100 uA)	Low (~20 uA)
Unique Features	Impedance-based SoC & SoH tracking, alerts	ModelGauge m5, no calibration required, compact size	High-voltage support, built-in ADC	Power consumption tracking, high precision

The specifications in which we decide to choose which Battery gauge model to go with are based on whether it will work best with the system status indicator power, size, and peripheral needs. Ultimately, below is the table showing why the MAX17261 is the best choice for the system status indicator. It has the most accurate SoC and voltage monitoring of all the other battery gauges, and boasts ultra-low power consumption and ease of integration with the ESP32. Considering the high level power consumption of the system status indicator, having a peripheral that contributes to lower power consumption is a must have.

Table 3.5.6.2: Battery Gauge Comparison

	TI BQ27542-G1	MAX17261	LTC2943	INA219
Works best with	Smartphones, IoT	Wearables, IoT	Industrial, Automotive	General Power Monitoring
Key Features	Accurate SoC/SoH	No calibration is required	High voltage support	Power Measurement
Interface supported	I2C/ HDQ	I2C	I2C	I2C
Cons	More complex to set up initially	Limited customizations	Higher than normal power consumption	No SoC estimation

While the other battery gauge peripherals have the required interface needed, only two of them were optimal for IoT, and only the MAX17261 had no calibration requirements, all while having very low power consumption, cheap costs, and being relatively compact in size. All of these factors make it an obvious choice to add onto the esp32 as peripheral for the system status indicator.

Battery & Charger module

The system status indicator subsystem needs a battery source to power all of the components. In order to do this, batteries with a sufficient amount of power output for the system indicator are required. There are many different types of batteries that could

theoretically work for this project, but the best option is to find a battery that can support the high power consumption of the esp32 and the battery gauge, the peripherals in the system, and be rechargeable, as the system indicator will most likely be hand held and able to be used mobile without the need of being plugged into a wall charger. In order to implement the batteries into the design of the system indicator, it needs to be inside of a battery holder to promote ease of assembly, connected in parallel over series in order to promote higher capacity.

Using the TP4056 charging module, which manages the charging for the Li-ion and LiPo batteries that will potentially be used, the system status indicator will be safely and efficiently charged. It also has features such as overcharge protection, automatic cutoff, and status indication which aid in preventing battery damage, ensuring that the batteries can have as long a lifespan as possible. This module is also very compact and affordable, making it the most optimal option for holstering the batteries that we choose for the system status indicator.

Table 3.5.6.3: Types of Battery Comparison

	Li-ion (18650)	**LiPo (Flat Pack)**	**LiFePO4 (Lithium Iron Phosphate)**	**NiMH (Nickel-Metal)**	**Alkaline (Disposable)**
Voltage Range	3.0V - 4.2V	3.7V	3.2V - 3.6V	1.2V (per cell)	1.5V (per cell)
Energy Density	~200 - 300 Wh/kg	~150 - 250 Wh/kg	~90 - 160 Wh/kg	~60 - 120 Wh/kg	~100 Wh/kg
Cycle Life	300 - 500 cycles	300 - 400 cycles	1000+ cycles	500-1000 cycles	Not rechargeable
Weight	Moderate	Lightweight	Heavy	Heavy	Lightweight
Size	Cylindrical	Flat	Cylindrical/Prismatic	Cylindrical	Cylindrical
Cost	Low	Moderate	High	Low	Very Low

Considering the following characteristics of the batteries above, it seems there are pros and cons to each of the choices. Below is a table that describes the best model for each type of battery, why that battery works best for the system status indicator, only if it actually does work, and its specifications compared to other models. After careful consideration of all the various types of batteries that could be used, it seems that 18650 Li-ion cells are the best choice as the primary power source for the system status indicator due its higher capacity and long runtime, making it ideal for a mobile handheld device such as the system status indicator.

Table 3.5.6.4: Battery Comparison

	Li-ion (18650)	**LiPo (Flat Pack)**	**LiFePO4 (Lithium Iron Phosphate)**	**NiMH (Nickel-Metal)**	**Alkaline (Disposable)**
Best For	Long runtime and high capacity	Lightweight designs like drones or robots	Applications needing a high cycle life	Budget-conscious projects	One-time use or backup power
Pros	High Capacity, widely available, cost-effective, reliable	Lightweight thin form factor, customizable sizes	Very long cycle life, stable and safe chemistry, high thermal stability	Affordable, no memory effect, good cycle life	Readily available, inexpensive
Cons	Bulky compared to LiPo, requires protection circuitry	More delicate, requires careful handling	Lower energy density, heavier, more expensive	Heavier and less energy dense compared to lithium-based batteries	Not rechargeable, unsuitable for high-power continuous usage
Capacity Range	2000 - 3500 mAh	500 - 5000 mAh	1000 - 3000 mAh	1000 - 2500 mAh	Varies

While the Li-ion cells are the best option in this case, the LiFePO4 batteries are also a solid option for a potential backup power source, since they are lightweight and compact, having high reliability and a long battery life. The LiPo batteries would also make a good primary power source option as well, but they are just slightly more delicate and require more careful handling, which should not be a major issue for the system status indicator, as it will be a lightweight handheld device.

While the batteries are integrated onto the ESP32, the power is managed using a DC-DC converter, any of which can be a boost converter if the battery voltage is lower than the system status indicator power input, or a buck converter if the voltage is higher. In that case, we could use the LM2596 for the buck converter and the MT3608 for the boost converter. Using the MAX17261 battery gauge mentioned previously, we can monitor the battery data in real time, integrate this data onto the status indicator board, and display the information on the screen.

Speed Sensors and Data transmission peripherals

The system status indicator should also have the ability to sense the speed at which the car is going in real time, and record the average speeds of the car during the beginning of any

race. With the speed sensors, or speed tracker, the speed of the car is tracked and calculated based on a speed tracking algorithm. This data is then transferred from the esp32 connected to the vehicle, onto the ESP32 connected to the system status indicator, where it is displayed for the user. There are many different options that can be used to track the speed of the vehicle, and various different methods to do so whether through GPS, tracking the wheel speed, or tracking the distance change on the vehicle in time.

One of the first options is the optical encoder, with the best model being the US Digital E4T, which is very ideal for any applications requiring high accuracy and resolution. This device is normally mounted directly on the wheels or a motor shaft, and counts the number of rotations or movements, hence providing the associated speed date to the ESP32. It has relatively compact size and good accuracy, making it a solid candidate for tracking the speed of the small scale autonomous vehicle in real time with the most precision. The only drawback is that it requires an ample amount of alignment and installation due to its high sensitivity nature.

The Hall Effect Sensor, with the best model being the Allegro A1101, is relatively much lower in costs than the other options, measuring rotational speed using magnetic fields. It is highly durable, meaning that it can last for long periods of time without requiring replacement, since there is no physical contact required with any of the moving parts of the vehicle. Since it does not touch any physical parts ever, it makes this a reliable sensor in environments where there is dirt, dust, vibrations, and many other kinds of debris during a race. It is a simple, effective, and affordable choice for a speed sensor, one that would be a great addition to the on board peripherals of the vehicle.

The Doppler Radar Sensor, specifically the model HB100, is a type of doppler radar sensor which measures the speed of any object relative to the sensor's position without relying on any physical contact. This is very useful for situations that require detecting motion of a vehicle in real time rather than just the rotations of the wheel, providing more accurate reading. While it boasts more accuracy than most other models, it consumes much more power and is relatively more expensive in comparison to other much simpler sensors. They are more well suited for detecting motion in an open environment, like the outdoors, which the small scale autonomous racing vehicle will not be exposed to, since the track for the vehicle is indoors.

The Ultrasonic Sensor option, HC-SR04, is very common for cases in which the measurement of distance must be taken, but they can also aid in tracking the speed. While the distance of the vehicle changes over time, the sensor can help estimate the speed by recording the changes in distance and position of the vehicle over time. This type of sensor is very affordable, and very simple to integrate, but the accuracy is not great, and they are not the best with rapid motion tracking. The small scale autonomous vehicle is also expected to be racing on a track at a high speed, which means that the HC-SR04 cannot be sufficient, since it measures only the forward distance of the vehicle relative to its position. It must be able to measure the speed of the vehicle without having a reading of the relative distance of the vehicle.

The Inertial Measurement Unit, like the MPU6050, otherwise known as an IMU, is another speed tracker that combines accelerometers and gyroscopes to track the speed and

orientation of a vehicle. This is a much more advanced and sophisticated option in comparison to the other sensors, boasting greater accuracy and versatility, creating more valuable data on linear and rotational motion of the vehicle. With more accurate readings, the MPU6050 also requires advanced filtering via Kalman or complementary filters to reduce the noise and drift to make the integration more complex. The sensor works best for situations in which precision orientation and motion tracking are critical measurements.

Lastly, the GPS Module, like the u-blox NEO-M8N, offers precise speed measurements for the vehicle, but mainly caters to outdoor environments. The GPS works by tracking the changes in the global position of the vehicle, making it highly effective for the outdoors. Since the small scale autonomous vehicle is indoors all of the time it is in use, there is no need for a global positioning tracking. The GPS module also consumes much more power than the other sensor models, and is overall too much for the vehicle's tracking needs.

Below is a table comparing the technical details and specifications of each model that has been considered for this project, along with their costs and sensor type. As seen, the cheapest options belong to the Hall Effect Sensor and the Ultrasonic sensor, although the difference between these two options lies in their functionality. The Hall Effect is a generally better suited sensor for tracking the speed of a moving vehicle than the Ultrasonic sensor is, since it tracks the rotations of the wheels with a magnet vs just tracking the distance ahead of the car and recording slow measurements of the cars position.

Table 3.5.6.5: Positioning Sensors Comparison

Types	Model	Technology	Power Consumption	Range	Costs
Optical Encoder	US Digital E4T	Incremental Optical	Low (<20 mA)	Limited to the wheels	$$ (moderate)
Hall Effect Sensor	Allegro A1101	Magnetic Field Sensing	Very Low (<10 mA)	1-5 mm from the origin magnet	$ (low)
Doppler Radar Sensor	HB100	Doppler Radar	Moderate (50 mA)	2-10 meters	$$$ (high)
Ultrasonic Sensor	HC-SR04	Ultrasonic Waves	Low (15 mA)	2cm - 4m	$ (low)
Inertial Measurement Unit	MPU6050	Accelerometer/Gyroscope	Moderate (50 mA)	Limited by calibration	$$ (moderate)
GPS Module	U-blox NEO-M8N	Satellite Positioning	High (100 mA)	Global coverage	$$$$ (very high)

Below is a comparison table showing the different speed sensor options and the more qualitative aspects of each that make them a solid choice. While the Optical Encoder boasts the highest accuracy for speed measurement out of all the potential options, this is not a deal breaker for the system status indicator purposes. Other important factors such as Ease of integration and the key features, along with the costs of the component matter much more than the output type and range for this specific project, as the speed, motion, acceleration of the vehicle are only being tracked.

As the qualitative below demonstrates, the US Digital E4T Optical Encoder is the best option to choose due to its high accuracy, compact design, and well suited design for wheel mounted applications. It gives real time motion data on the amount of rotations, allowing the indicator system to display the most precise speed calculations possible, improving the understanding of the vehicles autonomous navigation. The IMUs and the GPS module, and optical encoder are all unaffected by environmental factors such as satellite availability and vibrations in the surroundings, making them highly reliable and durable options. The Optical Encoder requires careful installation procedures, but the accuracy and reliability are worth the effort for the small scale autonomous racing vehicles needed for precision control and monitoring.

Table 3.5.6.6: Speed Sensor Comparison

Types	**Output Type**	**Accuracy**	**Ease of System Integration**	**Key Features**
Optical Encoder	Digital Pulse	High (0.1 degrees)	Moderate	Great resolution, compact
Hall Effect Sensor	Digital Pulse	Medium (0.5 %)	Easy	No physical contact required, very durable
Doppler Radar Sensor	Analog/ Digital	Medium (5%)	Moderate	Calculates speed without dependence of wheels
Ultrasonic Sensor	Digital Pulse	Low (10%)	Easy	Very affordable, easy to integrate
Inertial Measurement Unit	I2C/ SPI Digital Data	High (0.1%)	Hard	Tracks motion dn orientation at the same time

GPS Module	UART/ SPI Digital Data	Low (2.5m)	Moderate	More precise for outdoors

Wifi/Bluetooth setup and peripherals

In order for the system status indicator to connect with the ESP32 module on the vehicle that will be responsible for measuring the battery level, wifi connectivity, and speed of the vehicle, there must be a connection in order to transfer the data. Luckily the ESP32 comes equipped with a dual-mode Wi-Fi and Bluetooth modules, enabling a cost effective solution to reducing the number of peripherals needed for the board, hence making the ESP32 a robust solution for this IoT based system status indicator. In order to set up the Wi-Fi and bluetooth, the code needs to be set up on the Arduino IDE, and downloaded onto the ESP32.

For the System status indicator, there will be LED lights that not only go off for the Wifi/Bluetooth status monitoring, but also for the charging status of the vehicle. The difference will be that the LED for the Wifi/Bluetooth will be a blue light that will blink if there is no bluetooth connection available, and it will stay blue if there is a connection. On the Display screen, there will also be a status symbol for the bluetooth, showing whether it is connected to the vehicle ESP32 or not.

The connectivity for the peer-to-peer communication will also be measured along with the Wi-Fi and bluetooth, since there will be an ability on the indicator subsystem to switch between the two protocols depending on the situation at hand. There are some cases in which one protocol may be better suited for the ESP32 communication over another, giving the user the flexibility to test and choose the one that best suits their needs.

Below is a comparison chart displaying the different characteristics of the Wi-Fi and Bluetooth options that are housed on the ESP32. As shown, both are great options for establishing connectivity between the ESP32s and a potential host, but they work differently and serve different purposes. Wi-Fi is more suited for clients to host connectivity, while bluetooth is more suited for wireless communication for streaming, data exchange, and pairing up two devices.

Table 3.5.6.7: Bluetooth vs Wi-Fi

	Wi-Fi	**Bluetooth**
Range	50m indoors / 100m outdoors	10 - 30m (BLE) / 100m (Classic)
Speed	High (150Mbps)	Low (1- 3 Mbps for Classic)
Power Consumption	Moderate	Low (BLE) / Moderate (Classic)

Use case	Internet connectivity	Device - to - device Communication

Since there will be two ESP32s connecting and communicating with each other by sending data back and forth, with the indicator subsystem requesting data from the ESP32 recording data from the vehicle, there needs to be a protocol in which to do so. Luckily ESP32 has a variety of different protocols to connect to each other via wifi, bluetooth, P2P, and even longer protocols for very long range distance communication.

Using ESP-NOW, a protocol which enables any ESP32 or ESP8266 device to connect directly to each other, peer-to-peer, without a wifi network needing to be used. It uses a 2.4 GHz frequency since Wi-Fi is a much lighter connection than a wired one, and this also allows for an almost instantaneous data transmission with very little overhead. This is great for many applications where low latency is needed, such as remote controls, and sensor data collections, such as the ones being used for the indicator subsystem.

The power efficiency is also fantastic, as it boasts a significant advantage for battery powered devices, avoiding the overhead of maintaining a constant Wi-Fi connection. The main issue is that it is quite short ranged, so it cannot go beyond 200 meters without losing connection, and is generally not great for transferring large amounts of data since it is designed for compact messages.

Wi-Fi is another reliable option for data transfer and connection between the two ESP32 Modules, being very versatile and high speed with the ability to connect with a singular network with one server. The ESP32 can connect to a network or they can create their own network, which is a very valuable trait to have for the indicator boards that need to transfer large amounts of data very fast without relying on the need of an external source of connection.

Table 3.5.6.8: Communication Protocol for ESP32

Protocol	**Range**	**Speed**	**Power Consumption**	**Ease of Integration**	**Use Case**	**Features**
ESP-NOW	200 m LOS	2 Mbps	Low	Moderate	P2P Communication	Lightweight communication, no Wi-Fi network needed
Wi-Fi (TCP/IP)	50m indoors, 100m LOS	11 Mbps	High	Easy	Reliable Data transfer, Large	Needs a Wi-Fi network or an AP setup to function
Bluetooth	10m	2.1	Moderate	Moderate	Audio,	Standard

Classic	indoors, 100m LOS	Mbps			moderate data transfer	protocol, supports many legacy devices
Bluetooth Low Energy (BLE)	30m indoors, 100m LOS	1 Mbps	Very Low	Easy	Low power IoT devices	Very efficient for small data packets
LoRa	5-15km outside	37.5 kbps	Very Low	Moderate to Hard	Longe Range communication	Great for rural or distant sensors

Graphical User Interface

To program the graphical user interface, the graphic display chosen needs to connect with the ESP32 to run a graphics library in order to display the functions we would like to see on the System status indicator. Considering that the ESP32 does not have as great of a graphics compared to the raspberry pi Zero w, this microcontroller has been considered for the further use and development of a quality graphics display that can properly display all of the information needed from the readings taken by the ESP32. The Raspberry pi zero w has the best built in graphics of all the microcontrollers, making it ideal for the display screen and best for housing the data taken from the ESP32.

For programming the majority of the GUI, the LVGL library seems to be the most robust option for working with the ESP32, as it is lightweight, has a variety of graphics options for a dynamic and intuitive user experience, and is relatively moderate to create. When compared to a much more powerful graphics processor housed inside of the Raspberry Pi zero w, the ESP32 comes short in terms of graphics quality, but it excels with its ability to make graphics that are easy to integrate with the recorded data coming from the other ESP32 onboard the vehicle.

The LVGL library is a great library for the ESP32 graphics display purposes, as it has many resources and good documentation for programming. It also has good performance on the ESP32, being optimized for most embedded systems and runs well with limited resources. The amount of extensions and customizations available on the LVGL library is second to none, having many widget options, various themes, and modern animations, helping to create a unique and modern user experience on the Indicator subsystem.

Table 3.5.6.9: ESP32 Graphic Library

	LVGL	**TFT_eSPI**	**Adafruit GFX**	**uGFX**	**GUIslice**
Ease of use	Moderate	Easy	Very Easy	Moderate	Easy

Performance	High	Very High	Moderate	High	Moderate
Key Features	Moderate widgets, highly scalable	Lightweight, optimized for drawing and fast rendering	Simple graphics primitives, great for beginners	Cross platform functionality and support, modular design, and widgets	Touch support, great for basic GUI designs but not as flexible
Memory usage	Low - Medium	Low	Low	Medium	Low
Customizable	Very High	Limited	Low	High	Moderate

3.4 - Power Management System

3.4.1 - Technology Comparison

For an autonomous racing vehicle to function properly, the power management system must successfully distribute electricity from the batteries or DC jack input to each powered component. To keep track of our vehicle's status, we have decided to implement a power management system that not only distributes power, but also monitors the power being provided to each component and reports power statistics to a remote indicator board. This will ensure that the team can verify proper power conditions for the vehicle to hopefully mitigate any power-related problems. To accomplish this, research was performed on the technology needed for this task. The fundamental requirements for this PCB include: power distribution/conditioning, power monitoring, data analysis, and wireless communication. Because many modern microcontrollers already include wireless communication, the data analysis and wireless communication hardware components will be merged into the same research section. Stretch goals may include: remote control of the power system, such as remote selection of the power input source from the indicator subsystem; "power on/power off" buzzer, and sending data from the Jetson Xavier NX that will allow the System status indicator module to show a mini-map with the vehicle's location on the screen.

Technical Constraints

The power management system on our autonomous racing vehicle will have many design constraints due to the high-intensity nature of autonomous racing. These design constraints impact the hardware component selection of the power management system PCB, size of the PCB, location of the PCB, and other decisions. The importance of realizing these constraints and adjusting to them is very high, and failure to account for these constraints may result in the PCB not fitting, overheating, not being able to communicate wirelessly, and overdrawing power from the battery.

One big constraint placed on the power management system PCB is the power

consumption. Because the vehicle will be driving very fast and doing intense computations at the same time, a lot of power must be reserved for the electric motors that power the drivetrain and the Jetson Xavier NX that performs the autonomous driving. In order to provide as much electricity to these components as possible, the power consumption of the power management system itself must be minimized. To accomplish this, every component selected for the PCB was chosen with low power and high efficiency in mind. In section 3.1.1, the research performed on the type of microcontroller and power regulation to use is a direct result of this design constraint. Instead of using voltage divider circuits with passive resistor components or linear voltage regulators, switching regulators like buck-boost regulators were chosen instead. This is because of their incomparable efficiency. While voltage divider circuits and linear voltage regulators dissipate excess energy in the form of heat, switching regulators store excess energy to be used at a later time, and switch off the power input to conserve energy. This also helps with another design constraint, thermal management.

If the PCB exceeds certain temperatures, components may begin to fail or even catch fire. A great example of this is the buck-boost converter we are deciding to implement. The thermal shutdown temperature is 165 °C, so if the PCB temperature gets to this value, then the power will turn off. Various methods can be used to combat these high temperatures, which are mostly caused by the excessive current draw of all the components together. On top of using high efficiency power regulators, adding heat sinks can dissipate heat much better by providing more surface area for the heat to escape into the environment from. Spacing out the components so that the hottest components aren't all directly next to each other can also help, especially when paired with entering low power modes when able and turning off certain components when they are not in use. When a component enters a low power mode or turns off when not in use, the current draw is either zero or very close to it. This low current draw is optimal for mitigating high temperature sources on the PCB because current generates heat. Another important constraint is the size of the PCB. The vehicle has a dedicated location for the PCB to be placed, with specific measurements provided for the screw holes. These screw holes will be in a rectangular formation. The screw holes must be located 44 mm apart lengthwise, with 3 mm of excess space allowed between each screw hole and the edge of the PCB for a total width of 50 mm. Lengthwise, the PCB will be a total of 100 mm long with the screw holes also being 3 mm away from the edge, allowing 94 mm of distance between the screw holes.

Another constraint of the power management system PCB is the wireless communication range. Because the vehicle could be traveling comparatively long distances away from the team, we need to have a wireless communication method that will maintain a good connection even when the vehicle is racing far from us. Some different methods of wireless communication that were considered include infrared, bluetooth, and Wi-Fi. Ultimately we landed on using Wi-Fi, because bluetooth needs to be a decently close distance in order to be effective. Similarly, infrared also needs to be a close distance, and there cannot be any obstructions between the transmitter and the receiver. This would make infrared completely useless for our application because the vehicle will most certainly be blocked by obstructions during racetime. Bluetooth Does not face this issue, but as mentioned, the range of bluetooth is severely lacking. Another issue with using bluetooth communication is the fact that the bandwidth is lower, which means less data can be sent at a time over the

wireless signal. In a situation where a lot of data must be sent at a time, it is important that the speed of the data being sent is high.

Another big constraint that the power management system will face is durability. The autonomous racing vehicle will be traveling at very high speeds, with many large accelerations taking place. The vehicle may also race outside, where environmental factors that can highly impact the performance of the vehicle include moisture or dust exposure. On top of all this, thermal shock and static electricity can destroy the PCB and prevent the vehicle from running. Luckily, there is a solution that can help with all of these constraints: Conformal PCB Coating. Conformal coating is a polymeric film coating that is sprayed over a completed PCB and acts as a layer of protection for the components onboard the PCB. There are five main types of conformal coating, differing in their composition. acrylic, silicone, polyurethane, epoxy, and parylene-based are used primarily, and each has their own advantages and disadvantages. For our specific application, silicone-based conformal coating seems to be the best option because of its elasticity - which helps the most with high vibration scenarios - and its high range of thermal protection. Having a high range of thermal protection is necessary for our application because the temperature of the PCB will be getting very hot at times due to the power demand. This is also one of the less expensive options, suitable for non-commercial applications such as ours.

Power Conditioning

The path of electricity in our autonomous racing vehicle must start at the LiPo batteries or a 19V DC jack on board and end at each device that will require power. The input voltage from the batteries will vary between 9V and 12.6V, with a nominal value of 11.1V. The input from the DC jack will be 19V. The fact that there will be multiple possible power inputs necessitates the use of a switch to toggle the power input source. The various output voltages for peripherals of the power management system include 19V, two 5V, two 12V, and a 12V 4-pin header for the LIDAR sensor. The power management system will also need power directed to the microcontroller and the multiple power monitors used on the PCB, as well as the wireless transmitter used for the system status indicator module communication - if a separate transmitter is to be used. To accomplish this, effective power converters are needed to convert the input voltage to the appropriate output voltages. After doing research on the types of power converters and which would be the most ideal for our project, a conclusion was reached. The top three styles of power converters that were investigated included simple voltage dividers using resistors, linear regulators, and switching (buck/boost) regulators. Throughout this document, the various regulators used for adjusting the voltage may be referred to as not just voltage/power regulators, but also voltage/power converters, voltage/power controllers, or voltage/power translators. It should be noted that all of these naming conventions are in reference to the same type of technology aimed at either increasing or decreasing one voltage value to another. For this section, the input type primarily discussed will be in reference to the 11.1V nominal voltage from the LiPo batteries, since this is the power source that will be used during use of the autonomous racing vehicle and it is a smaller voltage value than the 19V jack input, meaning it will most likely require more current draw. When researching various power conversion methods, the 19V jack input source is taken into account as well.

Using resistors as a voltage divider has the main advantage of being simple to design. With very little math and inexpensive components, this seemed like a good potential option for power distribution. This method could only be used for any voltage output that is less than the 11.1V nominal input for the system, so this could potentially work for the two 5V outputs. Upon further investigation, this method seemed less ideal because of its lack of efficiency and stability, as well as invariability for multiple power sources. The resistors must dissipate the current drawn from the power source to generate an appropriate divided voltage as the output, and this dissipated current is converted to heat. This makes it extremely inefficient. In a setting where efficiency is very important because a limited battery means a limited drive time, this is a large drawback. Because the voltage divider is completely dependent on the voltage input – which varies between 9V and 12.6V for just the batteries – the voltage divider circuit will also vary linearly with the input. The resistors used will have constant values, meaning there's no way to adjust the ratio to allow for non-linear voltage regulation. This means that the output will be very unstable, which is not desirable for the components being connected because this could give inaccurate readings due to brownout or even break the connected component altogether. These two major factors ultimately pushed us to look for alternative options for power regulation.

Another method that was researched for power conversion was a linear dropout regulator. A linear dropout regulator solves one big problem that the voltage divider circuit faces: greater stability in output. Because the linear regulator acts as a sort of variable resistor, the output-to-input voltage graph will be closer to a horizontal line than the voltage divider graph. This indicates that the output voltage will be less affected by the input voltage, which is very desirable for a project that relies on a battery with a voltage supply range of about 3.6V. Although this is a big improvement compared to the voltage divider, the linear regulator still lacks in the efficiency department. Any excess power from the input that does not get used by the output is still dissipated as heat, which not only means that electricity is being wasted, but also that the vehicle will begin to heat up rapidly. A linear regulator gets more inefficient as the voltage difference between the input and output increases, so the 5V outputs will be incredibly inefficient using this voltage regulation technique compared to our other voltage outputs. This technique has another drawback that the voltage divider circuit also faced: the output voltage must be lower than the input voltage. That means this will not work on the 19V output and, in cases where the battery supply voltage drops below 12V, the 12V output. For these reasons, the linear regulator was ultimately denied as a viable option.

The last option for voltage regulation is a switching regulator. A switching regulator uses components like capacitors and inductors to store energy for later use and can "turn off" the power input. This means that for buck converters, which convert a higher voltage to a lower voltage, the efficiency is extremely high. The buck converter can store excess energy from the input, turn the input off and dissipate the stored energy into the output instead of heat, then turn the input back on and start over. This not only mitigates the inefficiency problem faced by the previous two regulation schemes, but also helps to reduce heat produced by the regulator. Because we will have two options for the power input, 11.1V from the batteries and 19V from the power jack, a buck-boost converter would be best. This type of converter can both increase the voltage or decrease the voltage from the input to the output, ideal for scenarios like ours where some peripherals require voltages between

the range of 9V and 19V - the range at which our input power will vary between. This is a great quality to have for our voltage regulator because it will reduce the number of components and even some of the complexity of our circuit, because the switch that toggles between the different power sources for the power management system would otherwise have to toggle not only the power sources, but also the type of regulator being used for higher voltages vs lower voltages. Instead, either power source can go through the same power regulator when selected. Downsides to this type of power regulation include complexity, price, and noise generation. Switching regulators typically require a more complex circuit schematic, with multiple components needing to be connected to the regulator for proper support. This inevitably leads to a more expensive PCB layout. The single switching regulator IC is also typically more expensive than the linear regulator IC, which adds to the price increase. Due to the switching properties of these converters, noise is also introduced to the output. This noise that is produced can be attenuated using passive components like capacitors, and these capacitors will also add to the overall PCB price. Although the switching regulator has these drawbacks, it still seems to be the best option for power conversion because the advantages far outweigh the disadvantages.

Power Monitoring

Monitoring the power of the autonomous racing vehicle plays a vital role in the diagnostics and runtime of the vehicle. If a component is not working correctly and the cause is unknown, an important metric to know in order to narrow down the problem is the power being provided to the component. On a bigger scale, if the entire vehicle is not operating as it should, the first thing to check will be the battery power. The vehicle will not be able to function if the central source of electricity – the batteries – are dead. For these two reasons, monitoring the battery state of charge (SOC) and the power statistics for the connected components will be necessary tasks for the power management system to perform.

To monitor the batteries' state of charge – and therefore display a battery percentage on the indicator board - we can simply measure the voltage output from the batteries, then compare this to a predetermined chart of battery voltages between a fully charged battery and a fully depleted battery. After digging around for a chart showing battery voltage vs. battery charge of our specific batteries, I realized there was not a chart provided by the manufacturer. Because of this, we will have to perform our own testing on the batteries to get an accurate estimate on how the voltage changes according to percentage of battery use. In the meantime, a linear correlation can be used to approximate the battery percentage based on voltage of the battery. For monitoring the power consumption of the outputs, a power monitor can be used to measure the voltage and current of each output. The voltage is easy to measure because a power monitor has an ADC that can read analog voltages, but the current is typically measured by using a very low resistance shunt resistor (typically less than 1 ohm) in series with the output line and measuring the voltage difference across it. By using a simple Ohm's Law equation, V/R=I, the current can be calculated. The component selected to perform the power monitoring for both the input and output is the LTC2945. This component was selected because our team already has prior experience with this component, so learning overhead is mitigated. This component is also low power, easy to interface with (I2C communication), and has a high accuracy, perfect for both input

and output. The initial idea for power monitoring involved two separate ICs for the input and output because the input would require more precision to give a more accurate battery percentage. This idea was ultimately aborted because the use of a wider variety of components on the same circuit board leads to higher prices and more learning overhead, since we would have to learn how to interface with two power monitors vs. only one.

Data Analysis

After power monitoring is performed and quantities are obtained for the power statistics, the power management system must do something with these values. The end goal is to be able to read the values on a remote system status indicator module, but before this, the values must be translated to human-readable numbers and transmitted wirelessly to the system status indicator module. To accomplish this, some sort of central brain is required to receive the values obtained by the power monitors and make sense of them. This central brain will also be responsible for sending out the commands that tell the monitors when to sample. For our central brain of the power management system, we will be using a microcontroller. The idea of using an FPGA also crossed our minds, but microcontrollers are less expensive and have a lower learning overhead for our team.

3.4.2 - Part Comparison

Microcontroller Comparison

Choosing the correct microcontroller for the power management system onboard our autonomous racing vehicle is a critical aspect to the power delivery for all the other components on the vehicle. An effective microcontroller must have a fast enough clock rate to monitor the power delivery to each connected component and send the information to the system status indicator module, as well as have Wi-Fi connectivity readily available to send information. This microcontroller must also be fairly easy to configure, as our time frame for this project does not allow for extensive periods of learning embedded programming from scratch. A big enough memory is also necessary to store any information read from the sensors. The last important quality our chosen microcontroller should have is ease of programming, meaning that the workload on the software side of the power management system should be limited. This can include the support of multiple programming languages, like embedded C, MicroPython, C++, etc.; Hardware Abstraction Layers to simplify the use of peripherals, like I2C, UART, etc.; or user-friendly Integrated Development Environments (IDE), like Arduino IDE. This will allow more time and effort to be spent on the hardware side of the power management system circuit board.

Table 3.4.2.1: Table of Microcontroller Comparisons

Microcontroller	Wireless Support	Multi-language support	Clock Rate	Price	Learning Overhead
MSP430FR6989	X	X	16 MHz	$10.05	Low

Raspberry Pi Zero W	√	√	1 GHz	$15.00	Medium
Arduino UNO R3	X	X	16MHz	$27.60	Low
Arduino RP2040 Connect	√	√	133 MHz	$29.40	Medium
ESP32	√	√	240 MHz	$1.00	Low

The microcontroller that we selected for the power management system onboard the autonomous racing vehicle is the ESP32. The different potential options for the microcontroller included the MSP430FR6989 from TI, Raspberry Pi Zero W microprocessor, Arduino UNO R3, and Arduino RP2040 Connect. All these microcontrollers/microprocessors had their pros and cons, but the ESP32 seemed like the best option for multiple reasons.

The MSP430FR6989 is a familiar microcontroller that many members of our group already have previous experience with, so this microcontroller started at the top of the list as the best contender for the power management system's microcontroller. With I2C connectivity, multiple low-power modes, and bitwise manipulability, this microcontroller seemed like a strong candidate. After realizing that the clock rate of the MSP430FR6989 is slower than some of the other microcontrollers (up to 16 MHz), it doesn't have onboard Wi-Fi connectivity or Bluetooth connectivity, and that it is limited to embedded C as the programming language, we decided to look at other options for the microcontroller to use.

The next option we selected was the Raspberry Pi Zero W. This microprocessor is a very powerful board that has fast Wi-Fi and Bluetooth capabilities, interchangeable storage (depends on size of SD card, up to 1TB), fast clock speeds (1 GHz), and a good amount of RAM (512 MB SDRAM). As good as this microprocessor is, it was a bit too much. This microprocessor proved to be out of our price range, and a much cheaper microcontroller that may not have all the same fancy features as the Raspberry Pi can still get the job done. Another downside to this microprocessor is the chip shortage caused by the large demand for them. This caused the prices to increase and the lead time to go up, so ordering one may prove to be unreliable.

Another possible microcontroller to use is the Arduino UNO R3, which uses the ATMEGA328P core. This microcontroller is an excellent option for low-power applications, and it is very easy to program and use because it utilizes the Arduino IDE. Unfortunately, this board is also somewhat expensive for less features than other microcontrollers, and the clock speed is lower than others as well (up to 16 MHz). The core of this microcontroller is also quite large compared to other options, which is most likely due to the fact that it was released in 2010. For these reasons, we decided against the Arduino UNO R3.

The Arduino RP2040 Connect was another top choice for our microcontroller. With Wi-Fi connectivity, a 133 MHz clock speed, and a dual core 32-bit Raspberry Pi microcontroller, this component is a fast and easy-to-configure embedded device. It is also incredibly efficient with very low power consumption. Like the Raspberry Pi discussed previously, however, this device also proved to be a bit overkill and expensive. At around $30 per unit, this price is just too high for what we need. Because we're spending more on the power monitor ICs to use switching power regulators, we should ideally find a microcontroller that's low cost but still powerful enough to accomplish what we need. With a Raspberry Pi RP2B2 core, this microcontroller is also experiencing a shortage, like most other Raspberry Pi chips. This microcontroller also has some learning overhead because previous experience with it in our group is very limited.

The ESP32 is the final microcontroller that we landed on using for the power management system. This microcontroller is inexpensive at around $1 per chip, easy to find & order on a multitude of electrical component websites, easy to use with many more peripherals and interfaces than its competitors, and has built-in Wi-fi and Bluetooth connectivity. The clock speed can reach 240 MHz as well, which gives it another large advantage over similar microcontrollers. While this chip does consume more power than the RP2040 Connect, it is worth it for the reduced price and ease-of-use. Another advantage to this microcontroller is the fact that many members of our group already have experience with it, which reduces the learning overhead associated with it. After doing much research on this device, I learned that it does not even require a Wi-fi connection with a router if it is connected directly to another ESP32, which makes it a great candidate if the indicator board also uses an ESP32.

Voltage Regulator Comparison

The decision of the specific voltage regulator to use in the power management system is vital to the performance and health of the overall system for the autonomous racing vehicle. Because the chosen technology for the voltage regulator is a switching-style voltage regulator – specifically, a buck-boost controller - we looked at various parts for this type of DC-to-DC regulator. The differing power input sources being from the 11.1V batteries or the 19V power jack means that the ideal type of power regulator will need to handle this wide of a range of voltage input, as well as handle approximately 3A of current. This is a lot of power for a voltage regulator, so it must be very efficient to mitigate heat being generated. The specific buck-boost converter selected is the LM5176 4-switch buck-boost controller. This component was selected for a variety of reasons, most of which have to do with its high efficiency, high power capability, and minimal switching noise attributes.

The two main types of buck-boost converters are 2-switch and 4-switch converters. While a 2-switch converter is easier and less expensive to manufacture, it is less efficient than a 4-switch converter, which is what the LM5176 features. A 2-switch converter can only function in buck-boost mode, making it un-ideal for situations where only one of the two (buck or boost) is required. Two of the regulators we looked at using are the LM2585 and XL6009, both of which are 2-switch regulators. Although these are less expensive, the deficiency caused by the fact that they are 2-switch regulators cannot be ignored. They also use diodes, which have a voltage drop across them. On the other hand, a 4-switch converter can be in each individual mode, either buck or boost. This increases the efficiency of the

converter. Commonly called “synchronous rectification” (or active rectification), 4-switch converters also replace the two diodes used in 2-switch converters with synchronized MOSFETs, which is where they get the name “4-switch converters.” This replacement decreases the voltage drops that would normally appear across the diodes, which also increases efficiency. Other advantages to 4-switch converters include bidirectional power flow (storing unused electricity back into source) and reduced component stress.

The LM5176 buck-boost controller also uses current-mode control where the inductor current is sensed and adjusted by the controller and used as feedback to the system so it can adjust the duty cycle of the FET switches. This provides a faster response over regular voltage sensing, providing a more stable output and a faster transient response to the input voltage changing or surging.

Other benefits to the LM5176 converter include **programmable soft-start functionality**, **cycle-by-cycle current limiting**, **input undervoltage lockout (UVLO)**, **output overvoltage protection (OVP)**, and **thermal shutdown**. All these features are very desirable for our system, which uses many expensive components that need stable power input.

A buck-boost converter that has a programmable soft-start functionality protects components from fast increases in current and protects power supplies from voltage dips. This works by ramping up the output voltage gradually rather than an immediate voltage increase and is manually set by adjusting an external capacitor’s value. This feature also prevents voltage from overshooting at the output as the voltage initially rises quickly.

Cycle-by-cycle current limiting is another means of protection that continuously monitors the current flowing through the inductor of the converter and restricts it accordingly. With a pre-set threshold for the current, the converter is able to turn off the current to the inductor if the current threshold is passed. This protects not only the inductor, but the MOSFETs and diodes as well. This feature has a quick response time, ensuring the protection of all components in the converter and connected to the converter.

Input undervoltage lockout does exactly what it sounds like. If the input voltage drops below a specified threshold, the converter is disabled and prevents any power at the output. Because of the basic premise of how a voltage converter works - the lower the input voltage is compared to the output voltage, the more current must be drawn from the input – as the input voltage approaches zero, the current draw approaches infinity. This is very bad in an electrical system, because more current means more heat. Very low voltages from the input can also lead to instability in the system. To prevent this from occurring, if the converter senses the input voltage dip below a specified threshold (undervoltage lockout level), it immediately shuts off the converter. After shutoff, the UVLO feature also typically includes hysteresis, which means the voltage must rise above a value slightly higher than the threshold value before turning on again. This prevents cases where the converter constantly turns on and off quickly if the input voltage is at or near the undervoltage lockout level.

Output overvoltage protection is another self-explanatory feature of the LM5176. If the

output voltage exceeds a specified threshold, the converter disables power delivery to the output. This protects any connected devices from experiencing high voltage values that can damage circuitry. Although there are many other overvoltage protection features in the LM5176 buck-boost converter, this feature is important because overvoltage at the output can be caused by a sudden disconnect of one of the loads as the inductor suddenly outputs excessively stored energy, failure of one of the components (like a shorted MOSFET), or even a malfunction in the feedback loop that monitors the power conversion in the LM5176.

Last but certainly not least, the thermal shutdown feature of the LM5176 ensures that the device shuts down if the temperature exceeds a specified temperature value. With so much current flowing through the converter, the device is guaranteed to heat up – quite a bit, in fact. With so much heat being generated, it is vital that the components inside are resistant to such high temperatures and are protected in the event that the temperature exceeds these high temperatures. With an integrated thermal sensor near components that are particularly sensitive, the controller is able to monitor its internal temperature and shutdown if it is too hot to allow itself to cool down before starting up again.

For all these reasons and more (especially the fact that our team has had previous experience with this component prior to the project), the LM5176 was chosen as the buck-boost controller to be used in the power management system. As can be seen in table 3.4.2.2 below, none of the other regulators we looked at have as many features as the LM5176. The only other 4-switch buck-boost regulator - the LT3958 - has not only less features with only over-voltage protection and thermal shutdown, but it is also more expensive.

Table 3.4.2.2: Comparison of Voltage Regulators

Property	**LM5176**	**LM2585**	**LT3958**	**XL6009**
Input Voltage	3.5V-42V	8V-40V	5V-58V	5V-32V
Current Limit	<5A	<3A	<4A	<2.5A
Switching Type	4-Switch (Buck-Boost)	2-Switch (Boost)	4-Switch (Buck-Boost)	2-Switch (Buck-Boost)
Efficiency	High (>95%)	Medium (85%)	High (>92%)	Medium (85%)
Protections	UVLO, OVP, Thermal Shutdown	Basic OVP	OVP, Thermal Shutdown	Basic OVP
Price	Medium	Low	High	Low

Power Monitoring Sensor Comparison

Choosing the correct power monitor for the power management system is another critical decision. To calculate the power of a system, the most used method involves measuring the voltage and current at the output of the power supply and multiplying the values in the

equation P = V*I. Measuring the voltage is fairly simple, as it only requires an analog-to-digital converter connected to the test point where the voltage is wished to be known. Measuring current, however, is a little more tricky. For this, we need to use a current monitor, which typically doubles as a power monitor since both voltage and current – and therefore power – are usually measured with the same monitor. Henceforth, the names "current monitor" and "power monitor" shall be used interchangeably.

The current monitor needed should be accurate, durable, easy to communicate with, robust, and able to withstand a large temperature range. Besides all of these, the most important factor in selecting a power monitor is price. This project is not a professional project in the sense that we will be selling this to a company or consumers. we will be using this vehicle for our own purposes, which includes racing it in a competition. For these reasons, the power monitor our team landed on is the INA219BIDR. This IC uses a current shunt method for measuring current in a wire. This method involves placing a very low but known value resistor named a "shunt resistor" in the path of electricity that the current will flow through.

By measuring the voltage drop across the shunt resistor with an analog-to-digital converter and using a variation of Ohm's law (I = V/R), the current through the shunt resistor can be calculated. Once calculated, the current is simply multiplied by the voltage measured in the equation discussed previously to give us a power value. This value will then be sent to the microcontroller where further statistical analysis can take place. For this communication, I2C or another similar communication method (like SMBus) would be ideal because we will require multiple sensors, and both I2C and SMBus allow for multiple devices to be connected to the same data bus.

This shows why a current monitor with I2C communication is preferable. The INA219 comes in two different grades, A and B. The difference between these two grades lies in the accuracy and precision specifications, with grade B being the best of the two. For our project we have selected grade B because of this.

The top other contenders for the current sensor we plan on using are also very robust and have some attractive features, including wider input voltage ranges and smaller package sizes. Although a smaller package size may seem ideal, since it would take up less space on the circuit board, it would actually be less ideal for our project.

The reason for this is that we don't have our own selective solder machine, we will have to place these components ourselves and even hand solder a few. A smaller component means a more difficult to install component, which automatically shoots down the MAX40080. This chip has many features that seem ideal, however the fact that it is a WLP package means that it would be very hard to work with due to the small size.

Another factor that is important to look at with these ICs is resolution. An analog-to-digital converter, or ADC, uses multiple capacitors that charge up and provide a reading to measure voltage. With more capacitors, an ADC can obtain a higher resolution of the current being measured. However, introducing more capacitors also brings the downside of having a lower sample rate because it takes time for all of the capacitors to charge up. This downside is not a huge concern for us because we only need the sensors to monitor

the output for human knowledge, so even a slow ADC will work fine.

Because of this - and a large price tag - the LTC2945 was decided against for our project. The INA219BIDR is the least expensive option for the project, so this also shuts down the INA226. Even though the INA226 is the second best option with all other features outperforming the INA219BIDR, the price is the biggest concern, so we are willing to accept that it may not do as good a job as the INA226.

The power management system shall have approximately 6 of these sensors, although this number is liable to change depending on testing performed after the power management system has been installed on the vehicle. Because each sensor will consume some power from the batteries, albeit a small amount, the number of sensors installed may impact battery life of the vehicle. Ideally, every input and every output should have a sensor to monitor the power through it.

This includes the inputs from the DC jack input and the batteries, as well as the outputs to the 12V-powered Jetson Xavier NX, 12V-powered Hokuyo UST-10LX rangefinder, 12V-powered Electronic speed controller, and 3.3V for the ESP32 microcontroller. Some of these may have to go without a power monitor, for instance the ESP32 since it has a built-in voltage regulator.

Table 3.4.2.3: Comparison of Current Monitors

Property	INA219BIDR	LTC2945	MAX40080	INA226
Input Voltage	0V-26V	0V-80V	0V-36V	0V-36V
Current Range	3.2A (programmable)	Wide	50mV shunt voltage	81.92mV shunt voltage
Package	SOIC-8	MSOP-16	WLP-10	MSOP-10
Voltage Accuracy	±1%	±0.4%	±0.1%	±0.1%
Resolution	High (16-bit ADC)	Medium (12-bit ADC)	High (12-bit ADC w/ high speed)	High (16-bit ADC)
Price	Low	High	Moderate	Moderate

3.5 - Motor Controller (Hardware)

Motor controllers are critical components in any robotics or vehicle system that involves motorized motion. They bridge the gap between the control logic of a system, typically executed on a computer or high-level processor, and the physical operation of motors. These controllers manage the power supplied to the motors and interpret commands to

achieve precise movements. The choice of motor controller hardware significantly affects system capabilities, ease of integration, and overall performance.

3.5.1 - Technology Comparison

When evaluating hardware options for motor control, two primary approaches emerge: general-purpose microcontrollers and autopilots. Each approach has distinct strengths and trade-offs depending on the requirements of the system. Below are some comparisons of those two.

Table 3.5.1.1: Comparison of hardware approaches to motor control

Controller hardware	Native support for vehicle systems	ROS Integration
microcontroller	X	X
autopilot	√	√

Because motors require a continuous PWM signal to operate, controlling them from a microprocessor such as a Raspberry Pi or Nvidia Jetson is infeasible [83]. This must be achieved through a microcontroller that can emit a constant signal, and our two alternatives are a general purpose microcontroller or an autopilot. Autopilots are purpose-built systems which contain a microcontroller in addition to internal sensors like IMUs and are designed to connect with additional RC vehicle hardware [84]. In addition, autopilots are generally built to run MAVLink-based firmware, either ArduPilot or PX4. Both of these systems provide a high level interface for passing commands to the vehicle as a whole and reading out system information which would otherwise be difficult to extract. The firmware running on them provides additional helpful features as well, such as automatically fusing its own IMU data with readings from an external localization source (e.g. SLAM or a GPS) in order to achieve accurate positioning. For this reason, we chose to use an autopilot instead of a general purpose microcontroller.

3.5.2 - Autopilot Comparison

Autopilots are specialized systems designed to manage the complex control, navigation, and communication tasks required in autonomous vehicles and drones. They integrate onboard sensors, motor controllers, and communication interfaces into a cohesive platform capable of executing high-level commands. Choosing the right autopilot system is crucial for ensuring reliability, accuracy, and compatibility with other components of the system. Below is a table summarizing some of our options for autopilot and their features.

Table 3.5.2.1: Comparison of autopilots

Autopilot	Number of IMUs	Microcontroller	PWM Out	USB Port	Price
HolyBro	2	32 Bit Arm® Cortex®-	16	Yes	$212

PixHawk 6c [85]		M7, 480MHz, 2MB memory, 1MB SRAM			
ModalAI Flight Core v1 [86]	3	216MHz, 32-bit ARM M7 STM32F765II	8	No	$200
Sky-Drones AIRLink [87]	3	STM32F7, ARM Cortex M7 with FPU, 216 MHz, 2MB Flash, 512 kB RAM	16	Yes	$2,790

When selecting an autopilot, the key considerations were cost, processing ability, and ease of debugging. Small, stripped down autopilots like the ModalAI Flight Core V1 are only the size of a quarter, but as a result have reduced processing capability and do not come with USB ports. This makes it difficult to connect them to a computer, and by extension difficult to calibrate the connected sensors and change other parameters. On the other end of the spectrum, there are systems like the Sky-Drones AIRLink which constitute an integrated autopilot and gpu-powered companion computer. For the price, we determined that having the companion computer built in wasn't worth it, especially since that makes it impossible to update the computer hardware later on without replacing the autopilot as well. In the middle, and most widely supported, is the PixHawk 6C, with a 480MHz microcontroller and a price of only $212. Its form factor is slightly larger than the Flight Core v1, but only by a couple of inches. Furthermore, it includes a USB port that allows it to be connected directly to a computer, where the parameters of its firmware can easily be set. Moreover, our sponsor had already purchased a number of these flight controllers for a previous project, so they were the obvious choice.

3.6 - Software Architecture (Communication)

Effective communication between system components is a cornerstone of robotic software architecture. Robots operate in dynamic and often unpredictable environments, requiring constant interaction between sensors, actuators, and processing units. This interaction must happen seamlessly, enabling real-time responsiveness and coordination. The architecture responsible for this communication must efficiently handle asynchronous data streams from sensors, synchronize control signals to actuators, and facilitate the exchange of information across various subsystems.

3.6.1 - Technology Comparison

Robots are inherently asynchronous systems. While all parts of the robot must operate in unison and at a high level a cycle for how the robot acts can be defined, each sensor is collecting a continuous stream irrespective of what's going on around it and each motor requires a constant PWM signal in order to maintain steady motion. As a result, an asynchronous framework must exist encompassing all of these components so their continuous tasks can occur simultaneously. The most common solution to this is ROS, the Robot Operating System. For reasons detailed below, we chose ROS for this project as our

primary software architecture. On the far opposite side from using an established platform like ROS (or MAVLink), we could use the client libraries for all of our sensors and simply write asynchronous code ourselves using a language like Python that supports subprocesses. Somewhere in the middle, there are message passing frameworks like RabbitMQ or Redis that support a general-purpose publish/subscribe architecture and work well with languages like Python.

Table 3.6.1.1: Comparison of asynchronous system implementation approaches

	Compatible with robot hardware out of the box	**Compatible with simulation software out of the box**	**Work required**	**Difficulty to debug**	**Customizability**
ROS [88]	√	√	Moderate	Low	Universal
MAVLink [89]	~	~	Low	High	Not supported
Pure Python	~	X	Extreme	Extreme	Universal
Python + Message Broker (Redis, RabbitMQ, etc.)	~	X	High	High	Universal

Pure Python/Javascript

This is the most straightforward approach, in terms of requirement to learn new technologies. However, as a direct result, it is by far the most complicated. Given that many pieces of robot hardware do provide Python SDKs (e.g. Zed camera [90]), it would technically be possible to implement an asynchronous framework for robot control entirely with Python.

However, this would require managing the inconsistencies in return type between the unrelated SDKs of the sensors being used, and writing new code to enable the motors to be able to read and act on the movement commands returned by the program. Given that frameworks already exist that can do this exact thing extremely well, converting sensor readings to consistent data types and routing motion commands to their correct controllers, it doesn't make sense for use to attempt to re-do that work ourselves. Further, testing would be very difficult because we would have to write our own code to connect our platform to common simulators and visualization tools, both capabilities that would come out of the box with an established framework.

Python + Message Broker

This option carries with it most of the drawbacks of the pure python option, but does make debugging and development easier by providing an interface for cleanly defining consistent

message types and efficiently handling storage of queued messages as they come in.

MAVLink

While ROS is by far the most common framework for developing robots, MAVLink is a very common communication protocol used for FPV drones and RC cars. It consists of messages for pieces of information common to RC vehicles, such as changing the control mode, receiving motion commands, arming motors, and checking battery charge. It is primarily used as the communication protocol between a base station or companion computer (a computer connected wirelessly to the vehicle or mounted on it, respectively), in either case emitted by a MAVLink framework.

The two MAVLink frameworks are ArduPilot [91] and PX4 [92], each with similar features. This protocol provides a very useful high-level view of what is happening on the hardware of RC vehicles, unifying the operation of many of the disparate systems they are composed of. However, because it is so heavily tailored to RC vehicles, it cannot be used for anything else. Programs can be written that receive and emit MAVLink messages, but out of the box solutions for general robotics tasks like mapping and planning do not exist or see the same level of support that they do in ROS.

A saving grace in this respect is the existence of an official bridge between MAVLink and ROS [93]. Because of the advantages of MAVLink over ROS in hardware level control and the advantages of ROS over MAVLink in terms of being a general framework for robotics, we use ROS as our core framework while using an autopilot built on MAVLink to convert motion commands into PWM signals, and use the ROS-MAVLink bridge as an intermediary between the companion computer running ROS and the autopilot.

ROS

Nearly all modern robots are controlled using either ROS1 or ROS2, a universal message passing framework for robotic systems that allows different components, such as a visualization program (i.e. RViz), a physical sensor, an electronic speed controller, etc. to communicate with one another in real time. Because it is so widespread, every reputable robot parts manufacturer produces well-supported integrations with ROS, along with all major simulation software. Instead of writing complex message passing or integration code manually, ROS enables us to simply plug a camera in and write a single line of code to begin listening for RGB images; because of the static list of message types, regardless of who produced the camera our code can expect the same image format every time. This allows us to massively speed up development over any pure Python or pure MAVLink-based solution.

However, ROS is designed solely for Ubuntu, and as such doesn't run on microprocessors like Arduino or on embedded hardware like an autopilot. As such, while it can rapidly emit movement commands, it has no good way of emitting a constant PWM signal of the kind necessary for continuous motor control. As stated in the MAVLink section, this is why we used ROS as our core architecture and MAVLink on an autopilot alongside.

3.6.2 - ROS Distro Comparison

The Robot Operating System (ROS) has become the de facto standard for developing robotic systems due to its modularity, extensive library support, and vibrant community. However, ROS is not a single static entity; it is released in distinct versions or "distributions" (distros), each tailored to support specific features, compatibility with underlying software, and hardware advancements.

These distros evolve over time, with new versions introducing enhanced functionality, improved performance, and extended hardware support, while older ones reach their end of life. Choosing the appropriate ROS distribution is a critical decision that impacts the project's development environment, compatibility with other software libraries, and long-term maintenance. Factors such as operating system requirements, community support, stability, and access to the latest features must be carefully weighed.

The table below provides a comparison of available ROS distros, highlighting their strengths and limitations to guide the selection process for the project

Table 3.6.2.1: Comparison of ROS distributions

ROS Distro	**Hasn't reached EOL**	**Compatible with F1Tenth Gazebo Sim**	**Ubuntu Version**	**Compatible with Gmapping**	**Compatible with Cartographer**
ROS1 Melodic [94]	X	√	18	√	X
ROS1 Noetic [95]	X	X	20	√	X
ROS2 Foxy [96]	X	X	22	~	√
ROS2 Jazzy [97]	√	X	22	~	√

We settled on using ROS2 Jazzy for both simulation and the physical hardware for the sake of easing the sim-to-real transfer and for maintaining the highest level of compatibility with current software. This was required in part because of our tech stack. The companion computers we use are Nvidia Jetsons, which run custom distributions of Ubuntu provided by Nvidia when the computer is first set up. Each of these custom distributions is built on a specific version of Ubuntu and is customized to that set of hardware. As such, upgrading an Nvidia Jetson to a newer version of Ubuntu is not well supported and, while possible, highly error prone.

The distribution of Ubuntu our Jetsons come with is based on Ubuntu 22, so for the cars using those computers we were forced to use ROS2. Additionally, since we are using Cartographer (see Mapping - Slam Package Comparison), ROS2 was an obvious choice for the simulator since there are no currently maintained Cartographer releases that support ROS 1.

3.7 - Mapping

Mapping is a fundamental component of robotic navigation, enabling robots to perceive and understand their environment. By constructing a representation of the surrounding space, robots can plan paths, avoid obstacles, and localize themselves effectively.

Mapping techniques vary widely, ranging from simple 2D grid maps to complex 3D representations that capture the intricate details of real-world environments. The choice of mapping strategy depends on factors like the robot's application, sensor capabilities, computational resources, and the level of accuracy required.

3.7.1 - Technology Comparison

Below is a table that outlines the benefits of different mapping techniques.

Table 3.7.1.1: Comparison of mapping techniques

	Enables raceline optimization	**Ease of Implementation**
SLAM	√	~
GPS	X	N/A
No mapping	X	√

In order to optimize the way it travels throughout the track, the car must have access to or be able to construct a representation of the track as a whole that it can place itself within. The family of algorithms known as SLAM algorithms, or simultaneous localization and mapping, achieves this through a number of techniques with the same general purpose: taking new point cloud data and attempting to align it with its current representation, and then correcting the estimates sensor pose to match where the points ended up aligning. Whether this is achieved through a particle filter or by constructing a pose graph, the end result is that as the robot drives around, it constructs a 3D representation of the world and is able to locate where it is within that representation at any point in time.

Because of the wide use of SLAM algorithms, a number of frameworks integrated with ROS exist to take in sensor data and produce such a map, making implementation non-trivial (one must still be careful in managing reference frames as the sensor data must be transformers to the SLAM coordinate system before it can be fused) but still relatively straightforward.

The primary alternative to building a map is to use one already generated; this is typically accomplished by affixing a GPS to the robot and using that data to determine its location on Earth. However, since we will be racing indoors, and the tracks are too small for GPS without RTK to be useful, GPS was not an option for us. The final alternative approach for mapping is to simply not do it at all. Robots are capable of navigating using reactive methods which only depend on the robot’s immediate surroundings.

However, these methods are unable to be optimized and are susceptible to falling for dead ends as the robot has no way to remember what it has seen in previous laps. As a result, we settled on using SLAM as our mapping technology.

3.7.2 - SLAM Package Comparison

Simultaneous Localization and Mapping (SLAM) is a cornerstone of autonomous robotic systems, enabling them to construct a map of an unknown environment while simultaneously determining their position within it. The effectiveness of a SLAM system is highly dependent on the software package used, as it dictates how sensor data is processed, maps are generated, and localization is achieved. SLAM packages vary in their approaches, features, and computational requirements. Some are optimized for 2D environments, while others excel in 3D mapping.

Key factors such as sensor compatibility, accuracy, robustness, scalability, and ease of integration with robotic frameworks like ROS must be considered when selecting a SLAM package. The table below provides a comparative analysis of leading SLAM packages, highlighting their strengths, weaknesses, and suitability for our project.

Table 3.7.2.1: Comparison of SLAM methods

SLAM Package	**Works with live sensor data**	**Pose estimate stable when vehicle isn't moving**	**Clean loop closure**	**Loop closure detection**	**Works with Nav2**
Cartographer [98]	√	√	√	√	X
Slam Toolbox [99]	√	√	√	√	√
Hector [100]	√	√	X	N/A	X
Gmapping [101]	√	~	√	X	X

Below is a visual representation of two of the SLAM Package mentioned above (Hector and Gmapping).

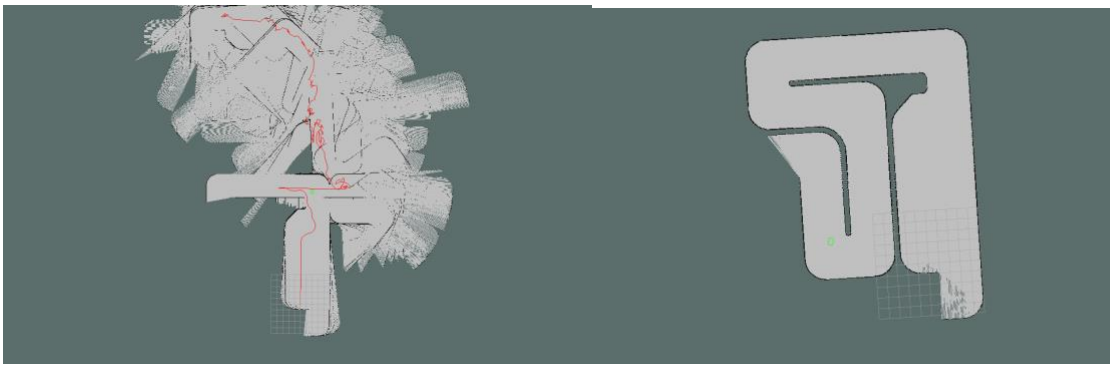


Figure 3.7.2.1: Hector SLAM (left) and Gmapping (right)

There are four major packages available to run simultaneous localization and mapping (SLAM) in ROS. Since the SLAM node selected here is the same one which will be used on the physical car during the race, it must be capable of creating a clean map of the world while properly localizing the vehicle and doing so as fast as new scans become available.

All of these three SLAM implementations function as a ROS node which is spun up at the start of simulation or driving and subscribed to the scan topic published to by the LiDAR or depth camera, the odometry frame, and the base frame. With the information from those three topics, the node computes an occupancy grid of the world surrounding the car, and then localizes the car to the correct spot in that occupancy grid. This position and occupancy grid (the map) is then published to another topic, which can be visualized or used in implementing autonomous planning.

The simulator we originally worked with came pre-configured to use Hector slam, so that was the first SLAM implementation we tested. In the benefits column, Hector slam runs very quickly, correctly fusing the map effectively as fast as the car can drive. However, after the car makes its first turn and begins exploring a wholly unseen part of the map, the position estimate begins to deviate in all trials, leading to a disjoint and unusable map akin to the one displayed earlier in this section.

Notably, a cursory examination of the data collected indicates that the localization failed before the mapping, as the tracked direction of the car abruptly reverses as the car continues straight after stopping. We are currently looking into a solution to this issue, as correcting the incorrect localization will allow us to take advantage of the rapid convergence of scan matching afforded by this implementation.

Gmapping is the most commonly used SLAM implementation in ROS, and as such it was the next one we tested. Because of its popularity, it is installable with apt-get, which greatly simplifies its integration into the software as version compatibility can be determined automatically and we do not have to rebuild our workspace to install it.

Similarly, because of its simple installation at the user level as opposed to the workspace level, it can be swapped in place of Hector SLAM in the mapping roslaunch file without introducing user-specific filepaths. Running the simulator with gmapping resulted in a far cleaner map and perfect odometry, as can be seen in the above graphic.

However, the position estimate of the car was not completely static when the vehicle was not moving, vibrating around the ground truth point slightly instead. While that vibration is unlikely to cause any issues as the problem disappears when the vehicle is moving, the former point is a pressing concern as the car cannot be stopped to allow mapping to complete during the actual race.

Unfortunately, upon further investigation we discovered that it uses a particle filter for scan matching, making it impossible to guarantee detection of loop closure; this is key for us as we need to detect when the first lap has been completed so we can optimize the raceline.

Cartographer is the most common SLAM implementation in ROS1. It is more difficult to install than Gmapping, but because it constructs a pose graph as it runs, it is possible to check for a cycle in the graph and know as a certainty that loop closure has been achieved.

Slam toolbox is the officially supported SLAM package of Nav2, which we will be using as our core navigation framework. As such, we selected slam toolbox for our SLAM implementation. It has the entire feature set of all other mapping implementations here while maintaining a large and active community of users.

3.8 - Control

Control systems are the backbone of robotic functionality, translating high-level commands into precise, actionable movements. Whether it involves steering a vehicle, maintaining stability, or coordinating multiple actuators, control ensures that the robot performs as intended in dynamic environments. Control systems must balance accuracy, responsiveness, and robustness to manage both predictable and unpredictable scenarios.

3.8.1 - Technology Comparison

The choice of control technology plays a critical role in the performance and adaptability of robotic systems. Different approaches to control, ranging from basic PID controllers to advanced model-predictive and adaptive control algorithms, offer unique strengths depending on the application. The table below compares control technologies, examining their capabilities and limitations regarding our project.

Table 3.8.1.1: Comparison of control methods

Control Method	Requires vehicle model	Amenable to multiple inputs and outputs	Handles lateral and velocity control
PID	X	X	√
Stanley	X	X	X
Pure Pursuit	X	X	X
Model Predictive Control	√	√	√

There are three major control techniques used in autonomous systems, namely PID control, Stanley control (specifically designed for autonomous driving), and Model Predictive Control. We plan to use all three over the course of developing our 1/10th scale autonomous racecar, but will only be using model predictive control in the final version. The reason for this is that PID control, and to some extent Stanley control as well, are very simple controllers as neither require a full model of the vehicle's dynamics. This has the benefit of making them very easy to implement, and for that reason we have worked with them in previous projects outside of senior design. Thus, we will be using them as a temporary solution to control the vehicle until the Model Predictive Control implementation is complete and tested.

PID is an incredibly well-known and widely utilized control method that uses a proportional term, an integral term, and a derivative term to smoothly respond to disturbances in some controlled value and guide it back to the desired baseline. PID control is used in everything from factories to control the temperature of furnaces to robots to control motion. It is its simplicity that allows it to be used so generally, and thus it is unsurprising that it can be used for both lateral and velocity control in autonomous driving.

That said, while it can be used for lateral control, it is designed to use constant parameters that make it better suited to linear systems; ie, ensuring a car stays at a constant speed as opposed to smoothly turning the wheels as the desired heading of the car continuously changes. Put another way, the constant parameters that result in a PID controller performing well in keeping a vehicle pointed the right direction on a straight portion of a track may result in erratic movement when the vehicle attempts to turn [102].

This shortcoming is largely addressed by the pure pursuit controller and the Stanley controller, both designed specifically to tackle lateral vehicle control. Both controllers use a generalized model of vehicle dynamics alleviating the need to create an entirely new model to use these controllers with a new vehicle. Pure pursuit controllers look a fixed distance ahead along the trajectory the vehicle is supposed to follow and use the center of the rear axle of the vehicle to calculate the cross-track error. Using a circle that intersects the rear axle of the vehicle and the lookahead point, it uses the kinematic bicycle model to calculate the steering angle necessary to drive to the lookahead point.

Stanley controllers instead use the front axle of the vehicle and add a heading error alongside the cross track error to reward the vehicle for maintaining the same heading as the desired trajectory while still steering towards the lookahead point [103]. This results in vehicle motion that is more consistent with the direction of the trajectory, eliminating oscillations. For this reason, we chose Stanley over pure pursuit as our lateral controller in the first implementation of the car pre-MPC (we will use PID for velocity control).

Unlike Stanley and pure pursuit, and much like PID, MPC is adaptable for a variety of use cases outside of autonomous driving. Unlike PID, however, an entire custom dynamics model and cost function is required. In our case, the cost function for our MPC controller will be deviation from whatever the current trajectory chosen by our local planner is. Based on the model of vehicle dynamics we create for our vehicle, MPC predicts the future states of the vehicle which will result from a certain steering angle, and computes the steering angle such that those predicted future states best align with the provided trajectory. Because

it can be adapted to compute multiple outputs, it is also the only way we can control both steering and velocity jointly in the same mathematical controller [104]. Thus, we choose to use this controller for the final implementation of the car.

3.8.2 - Implementation Comparison

Below is a table outline our different approach to our control algorithm implementation.

Table 3.8.2.1: Comparison of control implementations

	Ease of implementation	**Customizable to vehicle dynamics**
Write our own controllers	Hard	√
Python Control Systems Library [105]	Moderate	~
ROS2 controller library [106]	Easy	~

Our choice in controller implementation primarily came down to selecting between a pre-made control implementation and writing our own controller. Because neither the Python Control Systems Library nor the ROS2 controller library contained support for Stanley or MPC controllers, we decided to write our own. This builds on our past experience with controller technology as well, as some of our team members have had experience with implementing Stanley controllers for RC vehicles. That said, we did use the ROS2 controller library's ackermann steering controller in the simulator in order to map from TwistStamped messages to changes in the joint angles of our URDF model.

3.9 - Planning & Obstacle Avoidance

Planning and obstacle avoidance are vital components of autonomous robotics, ensuring that a robot can navigate its environment efficiently and safely. Planning involves generating an optimal path from the robot's current position to its target destination, considering factors such as distance, terrain, and energy efficiency. Obstacle avoidance complements this by dynamically detecting and maneuvering around obstacles in real-time, enabling the robot to adapt to unexpected changes in its surroundings. Together, planning and obstacle avoidance empower robots to operate in dynamic, unstructured environments with minimal human intervention.

3.9.1 - Technology Comparison

Choosing the right technology for our project involves evaluating multiple alternatives based on their functionality, performance, and compatibility with our project goals. The table below provides a brief comparison of technologies relevant to the topic in regard to our project.

Table 3.9.1.1: Comparison of local planners [107]

Local planner	Fast online	Can compute optimal path
RRT	√	X
Clothoid Sampling	X	√
Graph-based planner	√	√

In autonomous racing, the local planner is the system that creates the trajectory for the controller to follow while avoiding obstacles. In all cases, the local planner attempts to create a trajectory that follows the pre-computed race line while circumventing other cars with minimal deviation. The three main techniques we considered for this are rapidly exploring random trees (RRT), clothoid-based sampling, and a graph-based planner. The first approach, RRT, creates a tree from the current node that randomly explores with new, smaller branches in all directions. This is technically inefficient, unless a heuristic is applied to guide the direction in which branches are grown, since it does grow branches backwards, but the sheer simplicity of the approach overwhelms that fact. An iteration of RRT stops when one of the branches ends up within a certain distance of the target point, and then traverses backward through its parent nodes to reach the initial point. The trajectory taken through the tree is then followed by the vehicle. While this almost always results in a quickly computed trajectory, this method makes it impossible to ensure that the resulting trajectory is optimal, as the branches may take a jagged path to their destination that leads to slow movement, erratic steering, or both. We will likely implement this as an initial method to test out the car before moving onto the more advanced methods, but it won't make it in the final implementation of the car.

A clothoid is a curve whose curvature changes linearly with the distance along it. They can be used to parameterize a spline, which is effectively just a piecewise function that defines a continuous and differentiable. In the sampling based approach, one chooses a grid of unoccupied points in front of the car in the direction they want the car to travel, and then simultaneously optimize splines leading from the car's current position to each of those points. Then, one removes all the splines that intersect with an obstacle, and for the remaining splines choose the route with the least curvature. This allows the car to maximize its speed. However, solving these optimization problems constantly is costly and can be considered a waste of computation. Graph-based planners also use clothoids, but once loop closure is achieved they precompute a graph of possible states and the splines between them all traveling in the direction of the race. Then, one can assign a cost to every spline in that graph based on its curvature, and create a virtual "finish line node" which connects to all of the points along the finish line with a cost of 0. Finally, one can use an efficient algorithm like Dijkstra's algorithm to plot the most efficient trajectory from the closest state in the graph to the finish line. By selecting an appropriate density for points in the grid, even if the car isn't on one of the predicted states choosing the closest point in the grid as the starting point of the trajectory will still result in close to optimal movement as the controller (MPC or Stanley) will treat the situation as the car simply being slightly off course [46].

3.9.2 - Implementation Comparison

Below is a table outlining the different approaches to implementing the planner for our system. Each approach is evaluated based on criteria such as difficulty of implementation, compatibility with ROS2, and suitability for high-speed applications such as racing.

Table 3.9.2.1: Comparison of planner implementations

	Difficulty to implement	**ROS2 compatible**	**Designed for racing**
Write our own planner	Hard	√	N/A
ROS Navigation [108]	Moderate	X	X
Nav2 [109]	Simple	√	X

Developing a custom planner provides full control over the system's behavior, allowing it to be tailored precisely to the needs of racing or other specific applications. However, this approach requires significant development effort and expertise, making it the most challenging to implement.

ROS Navigation is a well-established framework for path planning and obstacle avoidance in ROS1 systems. While it offers ease of use and community support, it is not compatible with ROS2 and lacks optimizations for high-speed applications like racing.

Nav2 is the ROS2 equivalent of ROS Navigation and offers similar functionality with improved compatibility for ROS2 systems. However, Nav2 extends well beyond the functionality present in ROS Navigation, and is both highly extensible and reconfigurable. While not solely designed for racing, this can be seen as an advantage as its general purpose nature makes it very easy to adapt for our specific use case while remaining structured and very well documented.

Critically, it inherently supports using behavior trees to direct the flow of control over the life cycle of the application. If this were not the case, we would have to write the complicated callback logic ourselves to ensure that the robot’s objective is being updated to match what it should be doing at a certain point of the race. In comparison, a behavior tree is a well known data structure that can be visualized by a number of different tools and easily debugged.

This makes the Nav2 framework more than a simple planner, but a framework with which we can implement the whole of the car’s control logic. It also allows users to add custom layers to cost maps, which will make it easy to direct the car during the exploration stage as we can give additional weight to unexplored points in the direction of the track and also allow us to visualize that data in Rviz2.

By analyzing these options, Nav2 was chosen for this project due to its balance of implementation difficulty and ROS2 compatibility, despite its limitations in racing-specific optimizations.

3.10 - Testing

Testing is a critical phase in the development of any robotic system, ensuring that software and hardware components function as intended under realistic conditions. It provides an opportunity to validate algorithms, identify bugs, and refine system performance. Effective testing balances the need for accuracy with development efficiency, often requiring a mix of real-world and simulated environments.

3.10.1 - Technique Comparison

Testing techniques in robotics vary in complexity, efficiency, and relevance to real-world operations. The two main types of testing are simulation-based testing and hardware-based testing. The table below compares these techniques to highlight their trade-offs.

Table 3.10.1.1: Comparison of testing techniques

	Ease of setup	**Speed of iteration**
Simulation	Moderate	High
Hardware	Hard	Low

We decided to do the majority of our testing in simulation. While pure hardware testing does eliminate any potential issues with sim2real transfer, we decided that it was against our best interests to wait on the construction of the vehicle before beginning development of the software. Furthermore, testing in simulation gives us the ability to iterate extremely rapidly, as the cars don't have to be physically reset to a starting point for changes in the code to be tested.

One of our team members has participated in a past robotics project that did almost entirely hardware-based testing, and a large portion of time ended up being devoted to messing with SSH connections, resetting trials, and managing battery life that could have been spent making direct code changes.

3.10.2 - Simulator Comparison

Simulation tools are indispensable for robotic testing, providing virtual environments to model the behavior of robotic systems under various conditions. Different simulators vary in their capabilities, such as support for realistic physics, compatibility with specific hardware scales, and operating system support. Selecting the right simulator involves balancing factors like accuracy, ease of use, and system compatibility. The table below compares available simulation tools, evaluating their features and suitability for our project to guide our choice of an appropriate platform.

Table 3.10.2.1: Comparison of Simulation Software

Simulator	3D Physics Engine	Amenable to 1/10th scale vehicle dynamics	Runs on Mac
F1Tenth simulator [110]	X	√	X
F1Tenth gym [111]	X	√	√
CARLA [112]	√	X	X
F1Tenth Gazebo [113]	√	√	X
Writing our own	√	√	√

Below is a visual representation of the F1Tenth Gazebo Simulator and the F1Tenth Gym Simulator

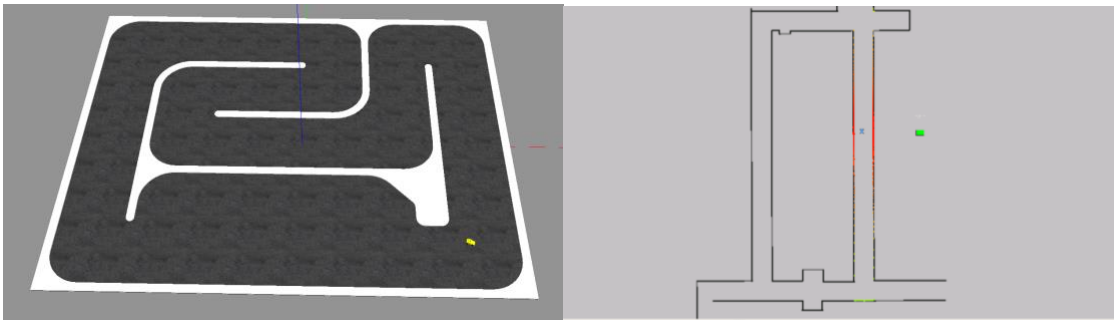


Figure 3.10.2.1: F1Tenth Gazebo Simulator (left) and F1Tenth simulator/gym (right)

We surveyed 4 different simulators for use with our project, and selected based primarily on two characteristics: the realism of the physics engine used and the amenability of the simulation software to the reduced-scale vehicle dynamics we are concerned with given the size of our vehicles.

There are two simulators officially affiliated with the F1Tenth organization, which organizes the races our cars will be participating in. The older of these two simulators, simply called the F1Tenth simulator, is a top down 2D simulator of an F1Tenth racing vehicle on a track. As such, it does not simulate the physics of the racing vehicle in full 3D fidelity. This allows it to run faster, but for the sake of being "quick to the point", we decided to skip using this simulator in favor of starting out with one that we would be able to continue to use throughout the project (ie one that fully simulates the physics of the car and that would allow us to easily transition from simulator to physical hardware when the time comes). Though we didn't end up using it, it did have some beneficial features, including the ability to be swapped out for a real 1/10th scale racing vehicle with no code modifications thanks to the naming convention of the topics it subscribes to [47]. The F1Tenth gym, the other official simulator, is similar to the origins feature-wise but is

dockerized and designed with ROS2 in mind. The dockerization allows it to run on Macs, which is especially tempting because of our team's Mac-heavy distribution of personal computers, but again the lack of 3D physics ruled this simulator out [48].

The most common simulator used in autonomous driving is CARLA, a well supported fully 3D simulator capable of GPU acceleration. Primarily used for production autonomous driving systems and research for systems intended to be applicable to real-world driving, CARLA supports additional features including traffic generation, weather, realistic cityscapes, and multiple agents. It is also compatible with ROS1 and ROS2, and provides an easily extensible framework for designing and texturing new vehicles. However, because it is so thoroughly intended for real-world scenarios, there is a large amount of additional complexity which would make it challenging to adapt to our very simple use case of driving a small car around a featureless and generic racetrack. Further, the API exposed for adding new vehicles requires selecting a base vehicle frame which cannot be scaled down to the degree that we need it to be. As such, we ruled out CARLA as a viable option [49].

The final simulator we tested was an adaptation of the Gazebo simulator specifically for F1Tenth-style vehicles. While this simulator is fairly old, using ROS Melodic and Ubuntu 18 as opposed to a newer stack with a ROS2 distribution and Ubuntu 22+, it checks the boxes we need it to check in that it natively supports 1/10th scale vehicles and has full 3D physics. The reason it is able to offer this level of fidelity is that, unlike CARLA, which uses its own simulation engine purpose built for their use case, this simulator is built on top of Gazebo which is a general-purpose robotic simulator. This has the added benefit of making it very easy for us to add additional maps, though the simulator comes with several pre-made 1/10th scale racing maps and a 1/10th scale vehicle already set up with a LiDAR and depth camera. The simulator also comes with tutorials to assist in setting up SLAM, and the included launch files automatically open RViz in the configuration necessary to see both the sensor readings and the fused map [50]. Unfortunately, because it is based on ROS 1, all of the code we wrote during in-simulator development would have to be substantially modified down the line to work with the physical cars, saddling us with technical debt that would be costly to repay. Furthermore, this simulator only works on Ubuntu, and does not support Cartographer.

Our final decision was to use none of those 4 simulators and create our own. We adapted the ROS Gazebo simulator from ROS 1 to ROS 2 and Gazebo Classic to Gazebo Sim, and ported over the vehicle and race track from the original simulator. From the ground up, we designed our implementation around dockerization allowing it to run seamlessly on any computer, including visualization support using VNC. This custom approach gave us the best of everything, and will allow us to iterate on the software component much more rapidly as well as easing the burden of sim 2 real transfer.

3.11 - PCB Design

Printed Circuit Board (PCB) design is a crucial step in the development of electronic systems, enabling the integration of components into a compact, reliable, and efficient layout. A well-designed PCB not only ensures the electrical functionality of the system but also optimizes physical size, thermal management, and manufacturability. Modern PCB

design involves the use of sophisticated Computer-Aided Design (CAD) tools that support schematic capture, layout design, and simulation.

3.11.1 - CAD Comparison

Software to use for the PCB fabrication included a few different options, but ultimately we landed on one schematic design tool. The top three options for PCB design included Altium Designer, Autodesk Fusion 360, and Autodesk EAGLE. All of these options have advantages and disadvantages, but ultimately, Fusion 360 became the top option.

Autodesk EAGLE was the first option for PCB design because most of our team members have prior experience with it. In Junior Design, the precursor class to Senior Design, EAGLE was used to construct a PCB. The program is very easy to use and understand, and it is also free. This made it a suitable option for our PCB design, but we realized that EAGLE will be discontinued in 2026. Although this has no effect on our current project, the purpose of Senior Design is to give us experience with a real-world project so we can take this knowledge and apply it in our careers. With this in mind, getting comfortable with EAGLE will unfortunately be useless because it will be discontinued soon. It would be more ideal for us to use a software that will still be around for a much longer time so that when we are working in the field, we can still use a software that we know for many more years.

The second option we considered is Altium Designer. This is a very strong tool with a multitude of functionality, and much better routing automation than many other circuit design softwares. Chock full of features, Altium Design has become a giant in the industry, and we considered it because some team members have prior experience. Unfortunately, Altium Designer is not only expensive, but it also has a large learning overhead for the team members that are not experienced with it. With so many features, Altium Designer is hard to adjust to because there are just too many options. With such a complex UI, Altium Designer was ultimately shot down as our circuit design software.

The final option for PCB design that we landed on is Autodesk Fusion 360. This circuit design tool is very easy to use, free, and will be around for many more years to come (unlike EAGLE). With many similarities to EAGLE, the transition from EAGLE to Fusion 360 should be minimal. Some team members also have prior experience with fusion 360, so the learning overhead is decreased even further. Fusion 360 also has tons of support online, from hobbyists and employees alike. For such an easy tool that also costs nothing, this option is the final circuit design software that we decided to use.

Table 3.11.1.1: Comparison of PCB Design Tools

Circuit Design Software	Easy to use	Price	longevity
Altium Designer	X	$355	√

Autodesk EAGLE	√	Free	Support ends in 2026
Autodesk Fusion 360	√	Free	√

4 - Design Standards and Constraints

Designing complex systems requires adherence to established standards and consideration of various constraints. Standards ensure compatibility, safety, and reliability across components, while constraints define the limits within which the system must operate, such as power, size, cost, and environmental conditions. Together, they shape the development process, streamlining integration and reducing potential errors.

4.1 - Standards

Standards play a critical role in the development of any technical system by providing a structured framework for design, implementation, and testing. They promote consistency, simplify integration, and enhance interoperability across components. In this project, standards are especially important in areas such as power management, communication protocols, and safety compliance. By leveraging existing standards, the team can focus on innovation while minimizing risks and ensuring compatibility with industry norms.

4.1.1 - Power Management System

Many standards will have to be followed for the power management system. These standards are vital to proper system integration and also make planning and production a lot easier. This is because standards that are already in place for us to follow mean that we don't need to "invent" or create our own standard, so less time needs to be dedicated to total planning. Two standards that will prove to be very important for us to follow are IEEE 802.11, or Wi-Fi, and the I2C communication standard.

4.1.1.1 - Electrical and Power Standards

IPC-2221: Generic Standard on Printed Board Design

This standard lists requirements to be met when constructing a printed circuit board, and will be applicable to the power management system because the entire system will be located on a PCB. Relevant requirements from this standard include coefficients of thermal expansion, voltage/ground distribution concepts, electrical clearance, conductor routing and spacing, and many more. It will be important to read through this standard thoroughly to ensure the PCB is constructed properly. One such section outlined in this standard that is vital to our project is PCB testability, an important aspect in the creation of any printed circuit board. This means that the PCB must be designed in a way that is easy to test for faults after construction, which can then lead to re-spins and redesigns of the original circuit

board. There are two main types of PCB testing: functional testing and in-circuit testing. Functional testing is essentially testing if the circuit board logic works as intended by subjecting the input to typical values that the system will expect to see during operation. During subjection to these inputs, the output is analyzed to verify that the response is functioning appropriately. In-circuit testing, however, is done to verify that the manufacturing was done correctly. This can involve measuring voltage values to look for short or open circuits, correct resistance or capacitance values, and other values that prove the correctness of the circuit. To aid in this type of testing, test points are placed in various locations around the circuit board during the design to give locations where a multimeter probe can be used to measure these values throughout the circuit. By using both types of testing, the total system should be very well adjusted and verified.

The first type of testing to be done should be in-circuit testing. The reason for this is because functional testing cannot be performed if the circuit itself refuses to work properly, which can be due to incorrect power delivery, circuit shorts or opens, wrong resistor/capacitor values, and a multitude of other reasons. To perform in-circuit testing, the high-level procedure should include verification of power regulator functionality by applying a typical voltage input to the board and measuring the voltage values into and out of each power regulator via designated test points. This will ensure that there are no faults with the power delivery circuits, which is an important first step. If this is not done properly, then the power being provided to all other components to be tested could be incorrect, resulting in bad readings, failures, or even broken components. Once the power delivery circuits are tested, then the testing can "move up the chain" by verifying the voltage values being provided to the MCU and current monitors. If everything seems correct and all the values are proving to be accurate, then functional testing can begin.

For the case of the power management system, the functional testing that can be performed includes using a power supply to power the system at an appropriate voltage level (between 9V and 19V to be accurate to our real-world power supplies). Then, the microcontroller should be checked for functionality. To test for this, the microcontroller can be programmed via JTAG and using a debug session in Arduino IDE to verify values in registers are correct. LEDs can be connected to some GPIO pins of the microcontroller to act as a visual verification method where program functionality is working. From here, the programming functionality testing can be extended to Wi-fi functionality, I2C bus functionality with the current monitors, and total functionality with the indicator subsystem.

Another standard from IPC-2221 that is important to follow is electrical clearance. Electrical clearance is the idea of spacing conductors on a PCB - whether on the same layer, different layers, or between different conductive materials - in an effective way to minimize interference and possible dielectric breakdown, where the voltage difference across a dielectric is large enough to arc through the dielectric and cause a short. When on the same layer, conductor spacing should be maximized whenever possible to avoid any potential issues. Interestingly, when using any type of coating over the circuit (like a conformal coating), the spacing between conductors must increase. This is because the dielectric constant of conformal coating is higher than that of air. A dielectric constant is a value given to all materials that tells how well a material responds to an electric field, and it is

also commonly called relative permittivity. With a lower dielectric constant, a material is better suited as an insulator because it will have lower parasitic capacitance properties than an insulator with a higher dielectric constant. Air has an approximate dielectric constant value of 1, whereas most coatings have dielectric constants between 2 and 8. This makes air a better insulator than conformal coatings, so when a coating is added to a circuit board, it effectively makes the insulation between conductors worse. This is why the spacing must be greater, because increasing the space between the conductors will provide more insulation between the conductors. Another factor taken into consideration in the chart is elevation. Higher elevations require greater spacing between conductors because at higher altitudes, the air pressure decreases. As air pressure decreases, so too does the dielectric constant of the surrounding area. This means that arcing is more likely to happen at higher altitudes, so the conductor spacing is increased to compensate for this. According to the chart below, the best spacing between conductors for our power management board would be at least 0.13mm, since the highest voltage between conductors would be 19V (between 16V-30V) and conformal coating is planned to be used. Luckily we live in Florida, so increasing the spacing due to higher elevations is not a huge concern; If our vehicle competes in a competition located in Colorado, however, this would be something to consider. With our current parameters and a 0.13mm spacing between conductors, the elevation should not matter according to the chart.

Voltage Between Conductors (DC or AC Peaks)	Minimum Spacing						
	Bare Board				Assembly		
	B1	B2	B3	B4	A5	A6	A7
0-15	0.05 mm [0.00197 in]	0.1 mm [0.0039 in]	0.1 mm [0.0039 in]	0.05 mm [0.00197 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]
16-30	0.05 mm [0.00197 in]	0.1 mm [0.0039 in]	0.1 mm [0.0039 in]	0.05 mm [0.00197 in]	0.13 mm [0.00512 in]	0.25 mm [0.00984 in]	0.13 mm [0.00512 in]
31-50	0.1 mm [0.0039 in]	0.6 mm [0.024 in]	0.6 mm [0.024 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.4 mm [0.016 in]	0.13 mm [0.00512 in]
51-100	0.1 mm [0.0039 in]	0.6 mm [0.024 in]	1.5 mm [0.0591 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.5 mm [0.020 in]	0.13 mm [0.00512 in]
101-150	0.2 mm [0.0079 in]	0.6 mm [0.024 in]	3.2 mm [0.126 in]	0.4 mm [0.016 in]	0.4 mm [0.016 in]	0.8 mm [0.031 in]	0.4 mm [0.016 in]
151-170	0.2 mm [0.0079 in]	1.25 mm [0.0492 in]	3.2 mm [0.126 in]	0.4 mm [0.016 in]	0.4 mm [0.016 in]	0.8 mm [0.031 in]	0.4 mm [0.016 in]
171-250	0.2 mm [0.0079 in]	1.25 mm [0.0492 in]	6.4 mm [0.252 in]	0.4 mm [0.016 in]	0.4 mm [0.016 in]	0.8 mm [0.031 in]	0.4 mm [0.016 in]
251-300	0.2 mm [0.0079 in]	1.25 mm [0.0492 in]	12.5 mm [0.4921 in]	0.4 mm [0.016 in]	0.4 mm [0.016 in]	0.8 mm [0.031 in]	0.8 mm [0.031 in]
301-500	0.25 mm [0.00984 in]	2.5 mm [0.0984 in]	12.5 mm [0.4921 in]	0.8 mm [0.031 in]	0.8 mm [0.031 in]	1.5 mm [0.0591 in]	0.8 mm [0.031 in]
> 500 See para. 6.3 for calc.	0.0025 mm /volt	0.005 mm /volt	0.025 mm /volt	0.00305 mm /volt	0.00305 mm /volt	0.00305 mm /volt	0.00305 mm /volt

B1 - Internal Conductors
B2 - External Conductors, uncoated, sea level to 3050 m [10,007 feet]
B3 - External Conductors, uncoated, over 3050 m [10,007 feet]
B4 - External Conductors, with permanent polymer coating (any elevation)
A5 - External Conductors, with conformal coating over assembly (any elevation)
A6 - External Component lead/termination, uncoated, sea level to 3050 m [10,007 feet]
A7 - External Component lead termination, with conformal coating (any elevation)

Figure 4.1.1.1.1: Minimum Electrical Conductor Spacing

IPC-9592: Requirements for Power Conversion Devices for the Computer and Telecommunications Industries

This standard describes the requirements for any type of power conversion system, whether it's ac to dc, dc to dc, or even power supplies. This is applicable to our project because the power management system is primarily power conversions. By converting the power from a battery to multiple outputs, the various standards described in IPC-9592 become applicable. Relevant sections of this standard include input power specifications, output power specifications, conformal coatings, and voltage spacing design requirements. With the help of this standard, we can produce an effective power management system.

UL 2054: Certification of Lithium-ion Battery

This standard describes the safe handling and use of lithium-ion batteries. This is applicable to the power management system onboard the autonomous vehicle because the main source of power comes directly from two lithium-ion batteries. By adhering to the guidelines in this standard, we can ensure the safe and proper usage of the batteries. Relevant topics in this standard include operating temperatures, charging requirements, and enclosure requirements.

IPC-2152: Standard for Determining Current Carrying Capacity in Printed Board Design

This standard is strictly for realizing the relationship between current, conductor sizes, and their effect on temperature. This will be very relevant to the power management system because one of the big concerns is heat dissipation. With so many devices requiring a lot of power at a time, the power management system could easily begin to heat up quite a bit because of the large draw of current. This standard will help us realize the impact of this and mitigate the effects through understanding how current and conductors affect heat generation.

4.1.1.2 - Safety and Protection Standards

IEC 61340-5-1: Protection of Electronic Devices from Electrostatic Phenomena - General Requirements

This standard specifies essential measures for protecting sensitive electronic devices from electrostatic discharge (ESD), a common risk when handling electronics. ESD protection is crucial for sensitive and high-value components, such as the Jetson Xavier, in the autonomous vehicle system. Adhering to IEC 61340-5-1 minimizes the risk of damage from voltage surges caused by ESD. Implementing decoupling capacitors to absorb excess voltage, alongside diodes to block transient spikes from reaching critical circuits, provides robust ESD shielding. This is particularly vital for maintaining the integrity of the power management system.

UL 60950-1: Information Technology Equipment Safety

UL 60950-1 outlines requirements for the safe handling and use of technology that processes data or facilitates communication. This standard applies to the Power Management System due to the microcontroller that processes data and communicates with the remote indicator subsystem via Wi-Fi. Key provisions include safeguards against

thermal overload, fire hazards, and electrical shock—each relevant to the microcontroller's operation within the power management system. Compliance with UL 60950-1 ensures the reliability and safety of the control and monitoring functionalities for the vehicle's power distribution.

4.1.1.3 - Communication Standards

IEEE 802.11: Wi-Fi

The IEEE 802.11 standard defines protocols for wireless transmission, specifying the medium access control (MAC) and physical (PHY) layers. It ensures interoperability across devices on Wi-Fi networks, allowing seamless internet-based communication. For the autonomous vehicle, IEEE 802.11 provides the framework for wireless data transfer between the Power Management System's microcontroller and the status indicator module. By following this standard, the system ensures reliable, standardized communication, allowing the remote module to consistently receive updates on vehicle status.

ISO 11898: Controller Area Network (CAN)

ISO 11898 provides protocols for CAN bus communication, essential in systems like the power management system, which employs I2C communication for monitoring voltage outputs and battery input. This standard details requirements for I2C's two-wire configuration—a data line and a clock signal—allowing multiple devices to communicate over a shared bus. The protocol defines a clear sequence for message transmission: a start bit, device addressing, read/write instruction, acknowledgment by the slave device, and an 8-bit data exchange. Given the fast-paced, high-interference environment of an autonomous racing vehicle, adherence to ISO 11898 ensures robust and reliable data transfer, allowing continuous monitoring and data retrieval from power sensors across the system.

4.1.1.4 - PCB Design Standards

IPC-2221, IPC-6012, IPC-A-600 - PCB Layout and Fabrication

These standards define the essential parameters for high-quality PCB design and fabrication. IPC-2221 establishes general design rules, ensuring that all components and traces meet basic structural requirements. IPC-6012 focuses on the qualification and performance standards for rigid PCBs, detailing the specifications necessary for reliable operation under varied conditions. Finally, IPC-A-600 provides visual acceptance criteria, guiding manufacturers on aspects like soldering quality, plating thickness, and overall board appearance. Compliance with these standards guarantees that the Power Management System PCB meets rigorous performance expectations and quality benchmarks.

Power and Ground Planes

In high-power PCB designs, establishing robust power and ground planes is essential to

minimize electrical noise and prevent voltage drops across the board. Proper design rules are necessary to ensure that power delivery remains consistent even under load changes. This is particularly relevant for powering high-demand components like motors and processors. Following best practices for power and ground plane layout reduces noise interference, enhances signal stability, and maintains overall reliability across the board's power distribution network.

IPC-2152 - Trace Width and Clearance

IPC-2152 provides specific guidelines for selecting trace widths and clearances on PCB layouts, especially in high-current paths. Adhering to this standard is critical for the paths supplying power to high-demand components, such as the electric motor. IPC-2152 ensures that traces can handle current loads without excessive heat or potential damage, safeguarding against issues like voltage drops or overheating. By following this standard, we ensure that the PCB's high-current sections maintain integrity and performance, even under continuous operation.

Component Placement and Signal Integrity

Proper placement of components on the PCB is vital to achieving optimal signal integrity, especially in designs involving high-frequency or high-power parts. Placing power monitors, filters, and other sensitive components strategically minimizes interference and crosstalk, reducing potential power losses. By adhering to layout guidelines, we avoid signal degradation, enhance overall circuit stability, and ensure that critical data readings, like those from power monitors, remain accurate.

4.1.1.5 - Reliability Standards and Testing

IPC-9701, IPC-TM-650 - Quality and Reliability Testing

Reliability testing standards are essential to confirm that the PCB can withstand operational stresses over time. IPC-9701 specifies testing protocols for surface-mount assemblies, focusing on assessments like thermal cycling and stress testing to ensure resilience under environmental fluctuations. Additionally, IPC-TM-650 provides a comprehensive suite of test methods to evaluate material properties, helping to verify that all components and connections meet stringent quality standards. Following these standards ensures the power management system PCB can endure the physical and thermal stresses encountered in an autonomous racing vehicle, maintaining performance and safety across varied conditions.

4.1.2 - Software Stack

REP 103

The REP 103 standard defines the units of measurement used across ROS packages, as well as conventions for using coordinate frames [114]. Individual robots frequently have numerous coordinate frames relating to different features on the robot, such as sensors, wheels, and grippers. ROS 2 uses the TF2 package to handle both static and dynamic transformations between coordinate frames, and as a result it follows that the frames must

have matching conventions for this to work effectively. Given that a vast number of ROS packages use information tied to coordinate frames–be it the SDK associated with a LIDAR sensor publishing its scan in the frame associated with the location of the sensor on the robot or a navigation package tasked with using that data to create a map and localize the robot–a standard was necessary to ensure all of these packages can work together reliably and in ways that can be predicted. The same goes for units of measure; as ROS is built on publishing and subscribing to informational messages broadcast on topics, it is crucial to ensure everyone is on the same page regarding how to measure that information. If some packages used imperial units and others used SI, that would both present an opportunity for unexpected bugs and require additional computation to continuously convert the units. For these reasons, REP 103 selects SI units as the measures underlying ROS and all packages associated with it (though, per the standard itself, exceptions may be made in cases where these units don't make sense; the provided example is packages for satellites that measure vast distances which are impractical to specify in meters). With regards to coordinate frames, x is used for forward, y for left, and z for up in most cases.

REP 105

The REP 105 standard goes along with the REP 103 standard in defining conventions for using coordinate frames, however instead of dealing with the measures itself it deals with naming conventions [115]. While not as fundamental, it is equally important; regardless of a robot's structure there are certain coordinate frames that are crucial nearly universally. These are the base_link, odom, and map frames. The first of these, base_link, is the highest level frame attached to the robot itself. It provides the parent node in the transform tree which all other nodes defining frames for specific parts of the robot are children of. Having this consistent parent is essential in particular for mapping, as it provides a meaning to "with respect to the robot"; in other words, the position of sensor data as well as objects external to the robot with respect to the robot as a whole is determined using this frame. The odom frame is used for dead reckoning, with the origin being the position where the robot was powered on.

While dead reckoning will always accrue errors over time, this frame is still important as it provides a continuous record of the robot's position. This is in contrast to the map frame, which is intended to be the fixed frame that defines the robot's actual position with respect to objects in the world. SLAM packages can use the odometry information in the odom frame along with sensor information transformed to the base_link frame to localize the robot in this frame. Since the odom frame already provides a continuous record of the robot's position, relocalization can correct the robot's position in the map frame in discrete jumps without worrying about potential consequences.

Finally, REP 105 defines the relationship between these frames in the transform tree; namely, that the map frame is the parent of the odom frame, which is in turn the parent of the base_link frame. Aside from these conventions which are directly relevant to our project, it defines conventions for aligning the generated map with a structured environment like position on Earth; we don't need to worry about that since our robot is not using a GPS and does not need to know what country its in so long as it knows where it is in the racetrack.

REP 138

REP 138 is the standard which defines the conventions for using LiDAR sensors in ROS [116]. Since LiDARs are such a fundamental sensor in robotics, this standard arose due to frustration with the differences in different sensors using different topics to output the same data. This made it challenging to create mapping packages that could work out of the box with any such sensor, again requiring unwarranted additional care and work on the part of end users. REP 138 declares that the topic on which to broadcast laser scans is, unsurprisingly, "scan", with "echoes" being used for LiDARs which can receive multiple readings for each laser.

4.1.3 - Mechanical Systems Standards

The following are important standards relevant to the design and manufacturing of the mechanical systems of the 1/10th scale vehicles. The listed standards are related to the performance modifications and additions made to the vehicles.

ISO 8855

As stated in the standard scope, this standard defines the principal terms used for road vehicle dynamics. The terms apply to passenger cars, buses, and commercial vehicles with one or more steered axles, and to multi-unit vehicle combinations [53]. Although this standard is not directed to small-scale vehicles, it can be generally applied to all vehicles - including the 1/10th scale or similar vehicles.

ISO 27548

As stated in the standard scope, this standard provides test procedures related to operating conditions for measuring particle and chemical emission rates emitted from desktop 3D-printing machines. This standard informs manufacturers and consumers on testing methods available to determine the toxicity of 3D printers [54]. This standard can guide our selection of and conditions of use of 3D-printers.

ISO/ASTM 52927

As stated in the standard scope, this standard provides an overview of test methods for the characterization of the mechanical properties of metals, ceramics, and polymers. It lists all the applicable standards based on specimens manufactured in a traditional process and gives the complement applicable when these specimens are manufactured by additive manufacturing [55]. In testing the strength properties of our manufactured parts, this standard can be used to guide and verify our processes.

ISO 1660

As stated in the standard scope, this standard gives the rules for geometrical specifications of integral and derived features [56]. In part drawing for out manufactured components, following well known standards such as ISO 1660 ensures that the manufacturing process of our components will be as smooth as possible as they would adhere to rules that have been adopted by manufacturing services and manufacturing technologies.

ASME Y14.5-2018

As stated in the standard description, this standard serves as a guideline for the design language of geometric dimensioning and tolerancing (GD&T.) It establishes symbols, rules, definitions, requirements, defaults, and recommended practices for stating and interpreting GD&T and related requirements for use on engineering drawings, models defined in digital data files, and related documents [57]. Similar to ISO 1660, this standard service to guide dimensioning and tolerancing of our manufactured parts, but is more widely used in the United States.

4.2 - Constraints

Just like any other project, constraints were placed upon the autonomous racing vehicle that impacted the construction and testing of the vehicle. With limited time and money - among many other constraints - our vehicle had to be developed in a specific way to adhere to these constraints.

Time Constraint

The time constraint for our autonomous racing vehicle is largely impacted by the fixed deadline of our live demonstration, set for early April 2025. This affects the pacing and research prioritization of the design and testing for our project. While additional time would allow for more in-depth research and learning, this deadline necessitates careful consideration of each decision's learning curve. In order to provide a working and reliable autonomous vehicle, sacrifices had to be made during the selection of components and functionality.

One instance of such sacrifice that was heavily influenced by this constraint appears in the choice of microcontroller for the power management system. Rather than learning an unfamiliar microcontroller within this timeframe, which risked surpassing our deadline or leaving insufficient time for testing, a microcontroller was selected that already had hardware and software familiarity within our group. Although a more advanced microcontroller could have better performance, it risks the delay of completing goals and objectives by specified time frames. The selection of a microcontroller that gets the job done and is intuitive for team members to use ensures the time limit is met. Other component choices were similarly affected by the time required to understand and integrate, including the selected power monitor and the voltage regulator.

To ensure constant progress and overcome potential setbacks in a timely manner, our team holds weekly meetings to view project status, discuss short-term goals, assess upcoming tasks, and address any potential roadblocks promptly. By supporting our weekly meetings using a project management platform called Jira - which further aids coordination outside of these meetings - we are able to assign tasks, monitor deadlines, define deliverables, and find any stalls in progress so as to address it immediately. With the coordination of the entire group every week, finishing the autonomous racing vehicle project by the deadline is much more manageable.

Other decisions that were impacted by the time constraint include electing to use off-the-

shelf components rather than fabricating completely new components and parallelizing testing procedures with research and development efforts to minimize total time needed. By using pre-made components for certain solutions, we bypass the time needed to construct our own specially-made part. Examples of this seen in our project include the vehicle itself; we are not making the RC car from scratch, but instead using an already existing vehicle to which we will affix our own technology to make it an autonomous racing vehicle, and the onboard computer; there is no need to reinvent the wheel so to speak, we just need to buy an already made computer that can handle the intense AI computations and drive the vehicle by itself. These decisions impact the time required for the project immensely because they are significant components that could even take years to manufacture by ourselves. By performing testing at the same time as research and development, not only do we save time over performing research and development then testing, but it can also help us detect and resolve issues as they arise.

Purchasing important components early into the project is another approach necessitated by the time constraint. Some components may have a large lead time, where waiting on a supplier to ship something to us could take quite a while. To mitigate this, many items that were very apparently necessary to the project were ordered early into the project's inception. The vehicle itself is another great example of this, as it was known from a very early point in the project that the specific RC truck that was chosen is the one that will be used. Another example of this is the laser RangeFinder used in the project to get information about the surrounding environment. By ordering these parts early, we ensure their timely arrival and provide opportunity for group members to begin project construction as soon as possible.

Economic Constraint

The economic constraint that comes with working with a limited budget significantly impacts the construction of our autonomous racing vehicle. Because we have a finite amount of resources to utilize in our respective contributions towards the project, cautious planning was performed to decide where to allocate the limited funds provided to us by our sponsors. Another consideration when selecting components besides financial limitation was the approvals needed for funding. For example, purchasing a microcontroller breakout board required multiple sponsor approvals, leading us to use personal funds to expedite access to testing equipment. While this approach allowed for faster prototyping, it meant opting for less expensive microcontrollers to stay within our own budget limits.

Most of the financial limitations placed on many components in the autonomous racing vehicle stemmed from the allocation of funds to top-priority, essential items like the Jetson Xavier and the vehicle itself, which includes the electric motor and the vehicle chassis. These components proved to take up a large portion of the budget, meaning limited funds were available to be spent on other areas of the vehicle. To alleviate the burden of financial limitations in these areas, decisions were made in the selection of components that opted for less expensive, but still reliable, solutions. One such example of these choices lies in the selection for the power monitor. We elected to use a power monitor that meets the requirements set forth by ourselves, but is by no means industrial grade or "top of the line." By finding this more affordable alternative, we are able to save costs to be used in other

areas.

Prototyping is another area where economic constraint has a major impact. Instead of outsourcing to manufacturing companies that produce high quality - but expensive - prototypes, we elected to use 3D printing because it is a much less expensive option. We were also able to use repurposed items from previous projects to save costs on buying new components or even fabricating our own. This not only helped the financial savings effort, but also aided in expediting the development process. Two such examples of this include the use of already-owned ESP32 microcontrollers and the use of an old power distribution system. A team member already owned two ESP32 microcontrollers from a previous class, so these were used for prototyping and program development rather than buying two brand new ESP32s. For the other case, a power distribution system that was used in an old F1Tenth vehicle was utilized to test the software functionality on the autonomous racing vehicle before the power management system was finished being constructed. This made testing much less expensive as a new one was not necessary to purchase, and the testing was able to be performed much faster as the wait time for the power management system construction was able to be bypassed.

The decision to purchase components based on versatility and scalability was another financially-driven choice. In the power management subsystem, voltage regulators that had a wide range for the voltage input and output were selected because they can be reused for a variety of voltage levels. This ensures that they will be compatible with any potential changes in the future. A great example of how this could happen is the onboard computer, the Jetson Xavier NX. The original component that we looked at using required an input voltage of 19V. Upon purchase and usage, we realized that it actually required an input voltage of 12V. Because of the versatility of the selected voltage regulator, this becomes an easy fix, and prevents the need to purchase a brand new component that will have a different output voltage. Instead, we can change some values of the peripheral components attached to the voltage regulator to adjust the exact voltage that it’s outputting.

Safety Constraint

Safety is paramount in any electrical project, especially given the high currents in our power management system. With high power flowing through the circuit, every component was selected with safety in mind to prevent overheating, combustion, or potential explosions. Ensuring that each part could withstand the system's power demands was crucial to protect both team members and others near the vehicle. Additionally, to further mitigate risks of electric shock and short circuits, we implemented conformal coating across the board. This coating enhances protection, ensuring greater safety under high-current conditions, while supporting durability in a potentially harsh operating environment.

Environmental Constraint

Operating conditions for the autonomous vehicle impose an environmental constraint on the power management system, as it must function reliably in a range of weather and road conditions. The power management components must tolerate exposure to moisture, dust,

and potential temperature fluctuations, which is critical for outdoor testing and racing scenarios.

To address this, all components are selected based on their rated environmental resistance, with additional protective enclosures considered for delicate parts. Furthermore, using sealed connectors and water-resistant housings reduces the likelihood of environmental interference, ensuring stable performance regardless of external conditions.

Weight Constraint

Given the high-speed racing requirements of the vehicle, weight is a key constraint. Every component within the power management system must be lightweight to avoid adding unnecessary mass, which could compromise speed and maneuverability. This constraint influenced our choice of materials and component sizes, as larger or heavier options were deemed unsuitable despite potential performance advantages. Where possible, compact and efficient components are prioritized, balancing functionality with a minimal footprint. This focus on weight optimization ensures the vehicle maintains peak agility and responsiveness, critical for competitive performance.

Power Efficiency Constraint

With finite battery capacity, the power management system must be highly efficient, imposing a power efficiency constraint. Each component in the system was chosen for its low power consumption and minimal thermal output to extend battery life, especially crucial for a racing vehicle where power demand is consistently high. This constraint guided decisions on voltage regulators, capacitors, and other components, aiming to minimize energy losses during power conversion. Efficiency is further enhanced through intelligent power routing to avoid wasting energy on inactive or low-priority systems, maximizing run time and ensuring power is directed where it’s most needed.

Interference Constraint

The presence of high-speed digital systems and motors in close proximity introduces an interference constraint, as electromagnetic noise can affect the power management system’s performance. This was particularly important when selecting components for the microcontroller and communication modules, where sensitivity to interference could disrupt data transmission and monitoring accuracy. Shielded cables, grounded enclosures, and filtering capacitors are implemented to mitigate electromagnetic interference (EMI) and ensure reliable power monitoring and communication. Adhering to these EMI mitigation techniques allows stable operation even in the electrically noisy environment of a racing vehicle.

Maintenance and Accessibility Constraint

With the need for quick diagnostics and potential field repairs, maintenance and accessibility represent an important constraint. The layout of the power management system was designed to ensure that frequently accessed components, like connectors, are easily reachable. This design approach allows for efficient troubleshooting, with modular

components that can be replaced quickly if needed. Additionally, diagnostic LEDs and test points are included where feasible to simplify maintenance, supporting faster response times in the event of an issue. This emphasis on accessibility ensures that any necessary adjustments or repairs can be made swiftly, minimizing downtime.

Integration and Compatibility Constraint

As the power management system must interface with various subsystems, an integration and compatibility constraint is imposed. Each component needs to communicate effectively with the Jetson Xavier, LIDAR, electric motor, and sensors, requiring careful selection of protocols and voltages to ensure compatibility. This constraint impacted decisions on power levels and data communication standards, as seamless integration is critical for coordinated functionality. To support this, standard communication protocols like I2C and Wi-Fi are implemented to enable straightforward interfacing, ensuring all subsystems work cohesively within the power management framework.

5 - Comparison of ChatGPT with Similar Platforms

In the rapidly evolving field of large language models (LLMs), multiple platforms offer powerful tools for generating human-like text, assisting with tasks, and enabling intelligent automation. ChatGPT, developed by OpenAI, stands out as one of the leading platforms due to its advanced capabilities, ease of integration, and broad application scope. However, to fully understand its strengths and limitations, it is essential to compare it with similar platforms offered by other providers.

5.1 - Comparison of platforms

We explored a detailed comparison of ChatGPT with other LLM providers, focusing on features, performance, usability, and unique differentiators. Table 5.1.1 presents a side-by-side evaluation of these platforms to highlight how they align with various use cases and project requirements. This analysis will provide valuable insights into the capabilities of ChatGPT in relation to its competitors.

Table 5.1.1: Comparison of LLM providers

	Can read documents	Can understand images	Can generate images	Can search the internet	Integrated multi-step reasoning
Grok	X	X	√	√	X
Claude	√	√	X	X	X
Perplexity	√	X	X	√√	√

ChatGPT	√	√	√	√	√
Gemini	√	√	√	√	~

Though OpenAI's ChatGPT and Anthropic's Claude often take center stage as the main LLM providers, Perplexity, Grok, and Gemini also exist as well-known and viable alternatives. However, ChatGPT remains the provider with far and away the greatest multimodal capabilities, able to read text from images just as well as PDFs and quickly generate images with text that closely align with a user's prompt.

Grok, the Twitter AI, remains the only major LLM provider that does not provide support for reading text from supporting documents. However, not all providers support document uploads for all users. Gemini only allows users on a Gemini advanced plan to upload non-image documents [117]; however, image uploads are available to all users. By contrast, Claude, ChatGPT, and Perplexity all support document uploads for free users. Each platform also has its own restrictions for how large uploaded files can be, and how this is measured varies depending on how files are categorized by the service.

OpenAI's ChatGPT divides non-image documents into text documents and spreadsheets, and sets a 50 megabyte limit for spreadsheets and a 2 million token limit for text documents [118]. This is because spreadsheets often lean towards being composed of numerical data that a token limit doesn't make sense for, while for text documents the number of tokens is more directly related to the size of the document. By contrast Gemini sets a universal limit of 100MB per file regardless of type, and has stricter rules about what file extensions are allowed [119]. Perplexity follows this mold, albeit with a much lower limit at only 25MB per uploaded file.

Additionally, Perplexity sets a hard limit on the number of documents which can be uploaded at 4 documents per query, and only supports uploading text based files. Anthropic has yet another way of measuring its file uploads, setting a dual limit of 32MB and 100 pages [120]. Uniquely, Anthropic's pdf support is vision based, allowing Claude to read scanned pdfs in addition to those which natively contain text. While this does impose additional limitations as the files must fall within the regular limits on Claude's vision system, this greatly expanded pdf processing capability gives Claude a unique edge in the document processing department.

While some state of the art LLM models are capable of understanding images through the use of an encoder which processes images into the same latent embedding space as the language it is otherwise operating over, no commercial model is capable of jointly generating images and text. This is because of the sharp difference between the autoregressive token decoding that powers text generation and the iterative diffusion steps that go into generating coherent images; the former is inherently linear while the latter is inherently an "all at once" process.

To overcome this shortcoming, many providers have created separate image generation models and made them available for the language models to use as tools. Instead of generating an image directly, the model just indicates that it wants an image generated, and

the image is then generated separately and included.

The providers that support this functionality are Grok, ChatGPT, and Gemini. While Google and OpenAI both trained their own image generation models, X's Grok uses the Flux image generator from Black Forest Labs [121]. This state of the art model substantially outperforms Dall-E and Gemini, which both have a glossy "ai-generated" look that frequently puts their generations in the uncanny valley. This is evident from the examples given in Appendix E. The woman in the first image generated by Grok has a much more human looking smile than the woman in ChatGPT's generation, and the depth in the background looks more realistic than the very 2-D background in ChatGPT's generation. In the second generation (involving a castle), it is worth noting the difference in the "default art style" of the models on the same prompt.

Gemini, likely trained with street view data, decides to make a photorealistic castle, while both other models err on the side of an illustration. Again, Grok's illustration is superior to OpenAI's, as OpenAI's generation is plagued by wavy lines and poorly defined borders between sections of the building. Due to the high costs associated with generating images as opposed to text, all three platforms place restrictions on the availability of image generation. Grok only supports image generation for its X.com premium users, ChatGPT only supports two image generations per day, and Gemini limits image generations by image content (e.g., generating images of people is only possible with an Advanced plan).

While Grok does have the best image generator, it unfortunately is not able to use it in chats which already include images; as such it cannot be used to refine an existing image. Of the two who can, Dall-E provides a better result; given a picture of a team member's living room, it maintains the spatial arrangement of items in the room while preserving to some degree the image on the flag in addition to the text. Gemini, however, did not, instead leaving out the TV entirely and producing barely recognizable flags. Given that both systems work by having the text model generate a prompt which is then fed to the image generation model, it seems that ChatGPT is more capable of generating a detailed prompt and DALL-E is more capable of following through with it.

LLMs such as those in the table above are increasingly used by consumers as a personalized alternative to traditional search engines. However, given that the responses generated by the models are based solely on text included in the prompt and training set, in order for up-to-date information to be included in the response it must be provided to the models as context. As such, a number of LLM providers have built functionality that either enables their models to search the internet using an API or retrieving text from articles matching the prompt ahead of time and including it immediately as context. Perplexity was the first to offer this kind of service, and remains a top performer.

When asked to retrieve recent research papers about a topic, it is able to do so, retrieving 3 relevant papers. The other providers with search functionality built in are ChatGPT, Gemini, and Grok. ChatGPT, when given the same prompt, is able to produce 4, and both systems provide correct links to each entry. While Gemini's response includes those same papers, the links provided to some of them are matched incorrectly; this is because Gemini uses Google search to verify the accuracy of its response after it is already generated rather than pulling in web-based context ahead of time. Finally, Grok retrieved 5 papers, and

while they are real it does not format its response to include the link (it is instead provided in a separate context, which is somewhat suboptimal from a UX perspective).

Though ChatGPT has largely caught up in terms of performance with its SearchGPT feature, Perplexity retains a unique edge in its ability to search for images and videos directly from the chat. Outside of chat-based web searching, Google has integrated Gemini into Google search directly, using it to create a summary of results almost instantly when a search is made. Unlike using Google search inside of the Gemini chat to check if an answer lines up with the sources available on Google, this feature does read the context of the web pages before writing the response, and thus it is generally more accurate (though less able to be directly asked complex questions).

One key difference between the way that LLMs "think" and how humans think is that humans are capable of ruminating on more difficult problems for longer in order to think through the steps required for a correct answer, while LLMs autoregressively decode their output immediately given the prompt. As such, without training to the contrary, LLMs will always parrot an answer that simply sounds correct, and will usually be wrong for problems that require complex thinking. To get around this, multiple calls can be chained together to simulate chain of thought reasoning by having the model have a conversation with itself.

However, this is a hacky solution to the wider problem, as the models themselves still cannot "think" in the ways users often expect it to be able to. ChatGPT and Perplexity have both made efforts to correct this. Perplexity does so by obscuring the repeated calls and prompt engineering within their own backend, following a set of steps after the initial request to create a valid answer. OpenAI took a more direct approach to the problem, creating a new model specifically trained to generate chain of thought reasoning by default, and to thus decode for longer before returning its final answer [122].

While this "bare metal" approach produces highly visible results, it is necessarily more expensive and time consuming to run as the bound on how much computation the model does per request is substantially higher. The additional computation time also means slower responses, and at time of writing paying OpenAI users are still limited to 50 requests with the new model per week, while free members are not granted access at all.

Though less advertised, Google rolled out a similar feature to OpenAI's o1 model some time before that model was released. Though it isn't a brand new model on its face, Google Gemini's multi-step reasoning is an update that causes the model to automatically produce multiple steps of reasoning when the prompt is deemed complex enough. However, it is not triggered unless deemed necessary for the purpose of limiting latency.

While LLMs are typically used as information generation/modification tools presently, as they become increasingly precise with their text generation they become increasingly capable of taking actions for users by making API calls. LLM-based systems capable of taking actions are typically referred to as agents, and a number of LLM providers are beginning to explore commercializing such features to gain a competitive edge.

ChatGPT currently supports a feature called GPTs [123], which are customized chatbots that can store pre-set additional context, prompts, and API schemas that can be used by the

agent to make API calls. Outside of that web-based feature, their API specifically supports "json mode", where the response to a prompt is forced to conform to JSON formatting. This means that the model, with that mode enabled, can be used to create the body of an API request (though there is still no guarantee that the request will work, or work as the user intended it to). Outside of calling APIs, this can also be very useful when attempting to include AI features in a software application.

OpenAI also released a feature known as a canvas, which allows ChatGPT to collaboratively edit documents alongside the user [124]. While this deviates from the traditional agentic archetype of a model which chats with the user while making API requests, it still can be considered an agent since the actions it takes are simply with the canvas instead of with respect to external resources. While Claude can't natively search the web, it also has access to a tool use mode that forces it to return its outputs in a provided format. In this vein, Claude has the unique ability to use tools that can control apps on a computer desktop, allowing it to behave much like a personal assistant with tasks that simply can't be done in a browser [68].

For our project, however, the additional features (save for web searching, which can be extremely helpful with finding sources for writing papers) provided by each of these LLM providers are not as important as the LLM's ability to reason effectively and be productive in assisting with reviewing code. We include an example of the difference in coding ability between the models in Appendix E, and from asking the models to create a pacman-style game playable in a web browser Claude and GPT-4o stand out as winners. GPT-4o was the only one of the models tried which created a map with multiple walls, similar to the original pacman game, and a ghost that could cause a game over.

However, the pac man created was unable to eat any of the white dots, and the score was not kept. Claude easily got the closest, creating a game with a blue border, multiple ghosts with eyes, a pacman with a moving mouth that could eat dots, a score counter, and a game over screen. However, it did not add any walls, leaving only an open grid of dots and no restrictions on movement within it. Additionally, the game speed is extremely high for the first couple of seconds before returning to normal. GPT o1 similarly created a grid of dots, but with no ghosts, no score, and a pacman represented as a single yellow dot.

Only Grok performed worse, creating an empty white rectangle with a featureless yellow dot that could move around. Out of all the models, the performance of Gemini was by far the worst, as even with explicit instructions to generate fully functional and complete code it generated a file full of comments indicating that code should be added there later. As a result, its solution did not run at all, and consisted of simply a yellow dot on a white canvas. Claude showed itself to be amenable to corrections, creating a very convincing result after being told to add walls and slow down the game speed.

Gpt-4o managed to correct its initial failure to make the dots collectable after being told to do so, but was unable to replicate the moving mouth of pacman despite being told to make the dots more stylized. These results were somewhat surprising since, while we used the OpenAI models only available to premium users, the free Claude model still greatly outperformed the others. We had expected the free models to perform similarly; the Gemini free model was clearly not up to the task asked of it by comparison.

5.2 - Learning Outcomes

The existence of LLM provider platforms like ChatGPT, Claude, Grok, and Perplexity is undoubtedly changing the way people get their education in a permanent way. It is clear that it is possible for students to abuse these platforms to their own detriment, but it is clear that they are equally capable of helping students learn and achieve more when used wisely. Below are examples of areas where LLMs have an impact on learning outcomes along with supporting case studies.

- **Debugging:** The existence of large language models trained on, in addition to the rest of the internet, nearly all of the stackoverflow questions and GitHub issues in the world makes for a powerful ally while debugging code. While setting up our F1Tenth Gazebo simulator, our main GPU-enabled workstation became unable to boot due to an incorrectly installed Nvidia driver.

  The BIOS was locked behind a long lost password and unable to be wiped due to the TPM being enabled, and with fast boot on it was impossible to get to the GRUB menu. In the end, o1 found a way to open a shell after startup failed because of the driver, and guided us to force the grub menu to appear. This enabled us to add the nomodeset parameter to the boot options, circumventing the graphics card during startup and allowing us to remove the corrupt driver.

  Without o1, it is likely that the computer would have had to be wiped. However, there are limitations to its ability to help with debugging. Namely, LLMs are excellent at helping with issues, even niche issues, in well-known environments like Ubuntu. Issues, even common ones, in niche hardware or software are generally outside the scope of its capabilities. In my work with PixHawk-based autopilots in pre-senior design projects, I experienced many issues with calibrating electronic speed controllers to understand the PWM signals coming from the autopilot. In debugging this issue ChatGPT provided no help, and in the end I resolved the issue through several days of trial and error.

  Thus, while there are limitations, the ability to have an LLM as an assistant in debugging code/computer issues is a major asset to our group in senior design, helping us create a better final product. In doing so, it also benefits our learning; the assistance it provides allows us to work through issues faster, gaining experience from more issues resolved in the same amount of time.

- **Rapid Prototyping:** Tools like GitHub Copilot that make use of GPT-type models are immensely helpful for rapid prototyping in that they have a shockingly good understanding of a user's intentions with respect to the code they are about to write. While this does not enable such tools to solve design or engineering questions where the user themselves doesn't know what the code they want to write should look like, it does enable users to get the code they are already envisioning written extremely quickly.

  In most cases, this takes the form of autocompleting function calls or repetitive blocks of code that involve slight differences. Still, this can and often does result in

significant time savings. This has a result much akin to the ability of large language models to assist with debugging. By increasing the speed of the development process, it lets us create more iterations in the same span of time, leading to a more finished and polished final product for our stakeholders.

At the same time, working through more iterations is inherently correlated with gaining more experience in solving bugs and refining our approaches to create the best result possible. Each second not spent typing code one has already worked through in their head is a second that can instead be spent resolving more pressing code issues and designing optimizations in our approach.

Most recently, using GitHub copilot assisted us in rapidly converting the ROS 1 simulator we were previously using into a dockerized ROS 2 simulator using the latest version of Gazebo. That process represented a substantial learning curve as much of ROS functionality is different between ROS 1 and ROS 2, and the new version of Gazebo had also been entirely restructured, but due to the amount of ROS 2 and Gazebo code present in the training sets behind the models that power GitHub Copilot, it was able to make helpful suggestions that helped us get the hang of how to effectively use the newer versions.

In particular, it helped with creating the new ROS 2 launch file using Python (we were forced to use Python for the launch file format since some processing had to happen during launch that wasn't possible when using the XML or YAML equivalents); some parts of the launch file are fairly repetitive, and using GitHub copilot to assist with writing them helped get the launch file written quickly while making correct educated guesses as to how parts of the formatting we didn't have prior experience with were supposed to look.

- **Knowledge Gathering:** A more traditional use for large language models, we have found them, particularly web search capable models like Perplexity and ChatGPT, to be great starting points for learning more about a particular topic. For example, one can ask a model to gather the basic information about a topic like SLAM, and the model can return a couple top sources explaining how that family of algorithms works and how we can implement it.

  From there, one can read those sources and use the links on those pages to find additional information. They are also surprisingly adept at distinguishing academic from non-academic sources, making them great tools for finding relevant related works. For example, if one is looking for all of the recent works related to open set multimodal 3D semantic mapping, a tool like Perplexity would be able to retrieve the top 3 related works published recently (response given in Appendix E). This ease of information retrieval cuts out time spent searching for information, allowing us to spend more time digesting it.

  In writing this paper, a good deal of time was saved using this approach. In particular, if there is something we know from prior experience to be correct that we still need a citation for, LLMs have been very helpful in not wasting time finding those sources.

6 - Hardware Design

The hardware design of our autonomous vehicle is a cornerstone of its performance, durability, and ability to adapt to diverse and challenging environments. This phase of development centers on crafting a robust, efficient, and scalable mechanical system that not only meets the vehicle's operational demands but also integrates seamlessly with its software and electronic subsystems. By combining innovative engineering principles with cutting-edge technology, the hardware design ensures that the vehicle operates at peak performance under varying conditions.

The design process is informed by extensive research and analysis of key performance factors such as structural integrity, energy efficiency, and environmental adaptability. This approach enables us to optimize the vehicle for both functionality and long-term reliability, ensuring it can handle the rigors of autonomous operation while maintaining high standards of safety and efficiency. Ultimately, the hardware design serves as the foundation upon which the vehicle's advanced capabilities are built.

6.1 - Vehicle Mechanical Systems

The vehicles will feature 3 main performance modifications/ additions related to the following:

1. Drivetrain
2. Weight distribution and management
3. Suspension geometry modifications and center of gravity placement

The following design concepts and plan have been informed by the research conducted on each of the 3 aspects above.

6.1.1 - Drivetrain

The drivetrain modifications will involve the replacement of the Velineon 3500Kv brushless motor and its replacement with the KingVal 4300Kv brushless motor. The Kingval motor will be paired with a belt-drive mechanism that will transfer the motor’s power to the vehicle’s transmission. A mount that spans the width of the vehicle will be designed and manufactured using PLA. This mount will be secured onto mounting points on the vehicle using screws, and the motor will then mount onto a surface on the mount. In the original vehicle, the Velineon motor interfaced with a 13-tooth, 32-inch pitch pinion gear with a 3mm bore diameter.

The goal of the belt-drive mechanism is to allow the new motor to drive the gear that the stock pinion gear drove (the driven gear) without increasing or decreasing the gear ratio between the motor and the driven gear. For this reason, the pulleys/ gears that the belt-drive mechanism is composed of will be identical. Finally, the original pinion gear of the vehicle will be attached to the belt-drive mechanism and will drive the driven gear. This concept is illustrated in the top view, left side view, and 3D concept model - Figures 6.1.1.1, 6.1.1.2, and 6.1.1.3, respectively.

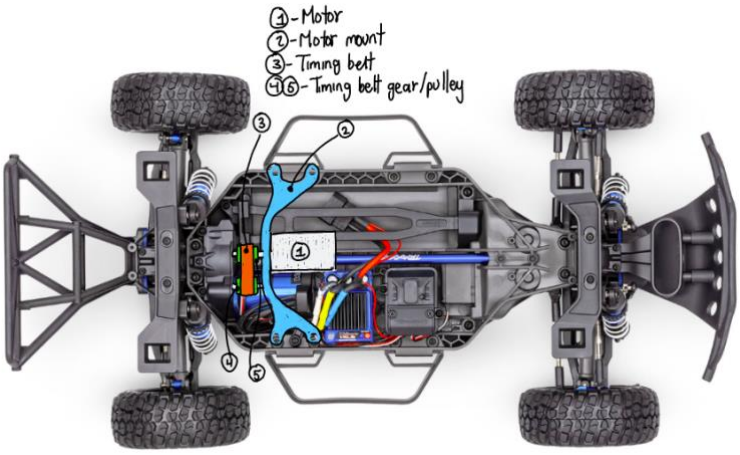


Figure 6.1.1.1: Drivetrain modifications concept. Top view.

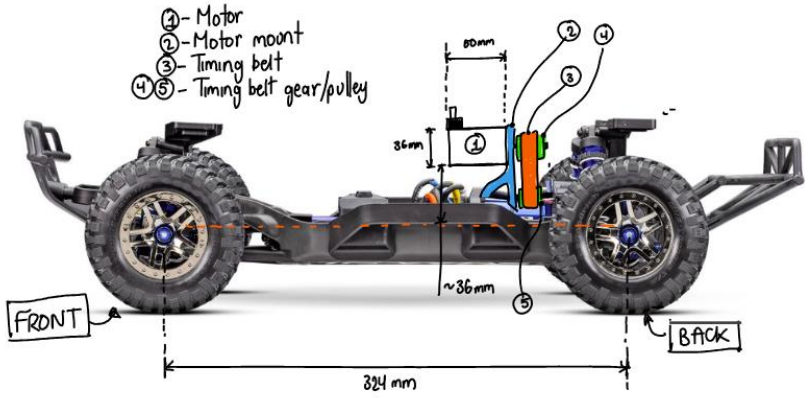


Figure 6.1.1.2: Drivetrain modifications concept. Left side view

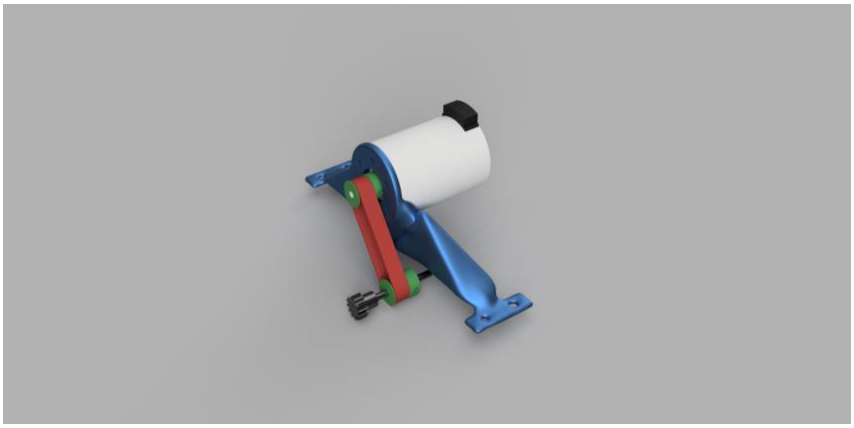


Figure 6.1.1.3: Drivetrain modifications 3D concept model

As mentioned previously, the mount used to secure the motor onto the vehicle chassis will be 3D-printed using PLA filament/ material. Research into material properties showed that

ABS and PLA would be good candidates for use in the 3D printing of mounting solutions for vehicle components. PLA will be used over ABS because PLA has a lower coefficient of thermal expansion and a higher ultimate tensile strength compared to ABS. The mount will be exposed to a motor with a high heat output; therefore, it is important that the mount is made out of a material that will have less deformation per unit temperature increase. Additionally, the higher ultimate tensile strength of PLA will allow the mount to withstand higher stresses imparted onto it through vibration or impact. The shaft of the KingVal 4300Kv motor is 3.175 millimeters in diameter and needs to be coupled with a belt-drive mechanism pulley with the same bore diameter. Common and readily available timing belt pulleys are the GT2 Aluminum pulleys. The GT2 pulley that best fits the KingVal 4300Kv motor is a 20-tooth, 3-millimeter bore pulley. In order to get the motor shaft fitted into the pulley and secured by the pulley's set screws, the pulley borehole would require enlarging through drilling or milling using a 9/16" bit. This would increase the bore radius and allow the motor shaft to interface with the pulley. As in the 3D concept model, a second GT2 pulley would be on the same 3-millimeter shaft as the 12-tooth pinion gear and would be driven by the timing belt. Although not illustrated, additional bearings/ supports will be used to ensure smooth movement of the shaft while effectively managing loads from the timing belt.

6.1.2 - Weight Distribution and Management

Balancing the weight of the vehicle between the front and rear of the vehicle will be a challenging task, but will be made manageable using a variety of filaments and through strategic component placement. Course adjustments to the center of mass of the vehicle can and will be made by moving large components such as the vehicle's battery, sensors, computers, and electronics boards/ controllers by significant amounts without negatively impacting electronic hardware or autonomous software system functionality or performance. Fine adjustments to the center of mass of the vehicle can and will be made by moving components in small amounts until the desired balance is reached. In the event that certain components cannot be moved due to space limitations or evidence of negative consequences of their movement, the material used to manufacture the component mount can be changed between PLA or ABS. The difference in density between the two materials would enable fine adjustments of vehicle mass balance.

Figure 6.1.1.4 shows the planned component organization on the vehicle. The major components are illustrated: The KingVal 4300Kv motor, the Traxxass 5000mAh battery, the Hokuyo UST-10LX LiDAR, the Trampa VESC, the Jetson Xavier NX computer, the power distribution boards and other lower level PCBs, and the vehicle center of mass. As illustrated, the components are in the rough locations they will be when the vehicle is fully assembled. The lidar, nvidia computer, and motor controller will be on the front half of the vehicle (right/ positive side of the vehicle's length centerline) whereas the PCBs, motor, and battery will be on the back half of the vehicle (left/ negative side of the vehicle's length centerline). Additionally, an assumption that the vehicle's mass center – without all the aforementioned components – will be on the back half of the vehicle is made.

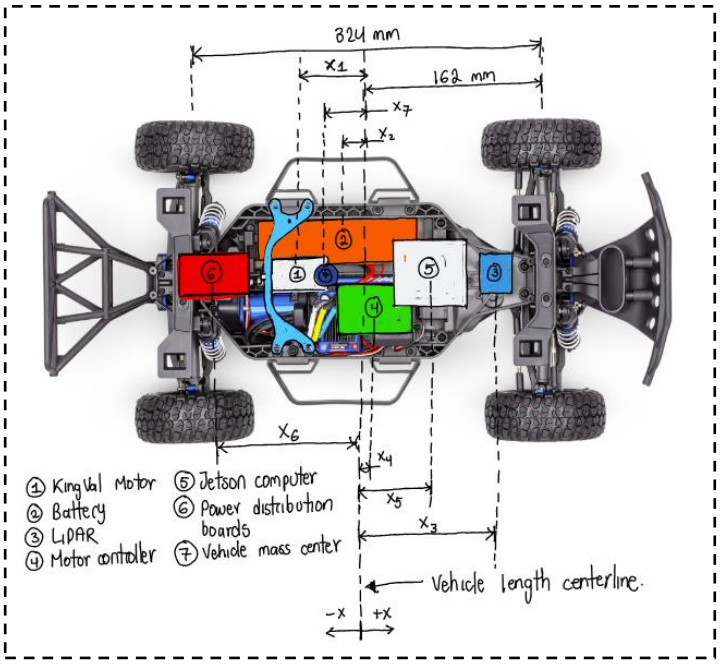


Figure 6.1.1.4: Major component organization on the vehicle

Each component center of mass is assumed to be at the geometric center of the component. This assumption simplifies the design with negligible, but non-zero, affects on the final calculations made using the center mass position. Each center mass position with respect to the vehicle's length center line is given as (x_i). The design challenge of achieving a 50/50 weight distribution between the front and rear axles of the vehicle involves selecting the center mass positions with respect to the vehicle's center line so they could result in a net zero moment with respect to the centerline. Equation 5 defines this relation. The values for each mass (m_i) is given in table 6.1.2.1.

$$m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + m_5x_5 + m_6x_6 + m_7x_7 = 0 \quad \textit{Equation 5: Center of Mass}$$

Table 6.1.2.1: Major component masses

Component	Mass (grams)
Motor (m_1)	74
Battery (m_2)	28.349
LiDAR (m_3)	130
Motor Controller (m_4)	232
Computer (m_5)	630
PCBs (m_6)	250 (Estimated)

Vehicle center mass (m_7)	2280

By fixing the positions of components such as the motor, battery (which had its own compartment and will be fixed regardless), LiDAR, motor controller, computer, and the vehicle center of mass, the final variable can be solved for. Theoretically, the resulting/ final component center of mass positions should result in a 50/50 weight distribution between the front and rear axles of the vehicle. In the event that the distribution is not near the desired 50/50 distribution, the material with which the component mounts are made out of (PLA or ABS) can be varied to bias the weight distribution towards the targeted distribution.

6.1.3 - Suspension Geometry Modifications and Center of Gravity Placement

The method selected to reduce the roll angle of the vehicle was the modification of the vehicle's roll center and center of gravity position. As discussed in the earlier research, the center of gravity of the vehicle can be adjusted continuously, whereas the roll center of the vehicle can only change discreetly. The center of mass of the vehicle can be positioned using ballast weight and further tuned using 3D-printed mechanisms that gradually lower the ballast weight to a desired position. Roll center can be modified on the Traxxas Slash 1/10th vehicles through the movement of each axel's suspension arms to set locations.

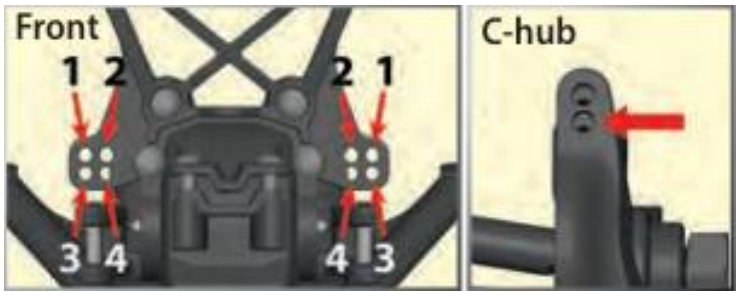


Figure 6.1.3.1: Front suspension, roll center adjustment points. Source: [125]

Figure 6.1.3.2: Rear suspension, roll center adjustment point. Source: [125]

The Traxxas slash 4x4 vehicles have attachment holes - as pictured above - for the front and rear camber links that enable the lowering or raising of the roll center of the vehicle. In our vehicle's case, there are a few methods that could be used to positively affect/ reduce the roll of the vehicle during corning. Each of these methods aims to reduce the distance

between the roll center and the center of gravity or mass and, typically, the center of gravity is above the roll center of the vehicle.

The first method would involve raising the roll center of the vehicle while maintaining the position of the center of gravity. This method would decrease the distance between the roll center and the center of gravity, which in turn would decrease the roll angle of the vehicle.

The second method would involve lowering the center of gravity of the vehicle while maintaining the position of the roll center of the vehicle. This method would also decrease the distance between the roll center and center of gravity and the roll angle. The third and last method would involve raising the roll center while also lowering the center of gravity. Like the first two methods, this would decrease the distance between the center of gravity and the roll center, thereby reducing the roll angle.

On the front suspension, position 4 is the default suspension setting. This setting is good for general use of the vehicle but can be changed by moving the suspension's camber links to position 1 or 2 - doing so would lower the roll center of the vehicle. Additional lowering of the front suspension's roll center can be achieved by moving the camber links to the "C-hub" of the vehicle.

For the rear suspension, the roll center can be lowered by moving the camber links to positions 4 or 5 of the real attachment holes. This process of lowering the roll center can be done blindly without concern for the exact position of the roll center, but for more accurate and predictable results, we aim to fully diagram the suspension setup of the vehicle after adjustments in order to calculate the exact position of the roll center and thereafter use analytical equations to predict the vehicle's roll angle through corners.

Plans to tune our vehicle's center of gravity involve the purchase of ballast weight such as the adhesive wheel balancing weights that can be sourced from Amazon. Such weights provide fairly granular adjustments to the extra weight added to the vehicle. Additionally, using a 3D-printed mechanism or structure to more precisely place the weights would improve precision and provide a continuous set of possible positions for the center of gravity. Such control of the center of gravity's location would be beneficial in testing vehicle behaviors and performance over a continuous set of possible roll angles. The resulting data can then inform our team of the most optimal center of gravity height for a desired roll angle target.

In order to meet the roll-angle reduction mechanism effectiveness of > 1.25, the distance between the roll center and vehicle center of gravity needs to be reduced. Equation 6 defines how effectiveness for the roll-angle reduction mechanism will be calculated. The equation is the inverse of what is typically the effectiveness of a system (The effectiveness is usually defined as the quotient of the measurable quantity with the system and the measurable quantity without the system). This inversion is due to the fact that the roll-angle reduction mechanism reduces the roll angle instead of increasing it. The typical effectiveness equation's numerator is usually the value that increases with the application/use of a system.

Equation 6:

$$\text{Mechanism Effectiveness } (\epsilon) =$$

$$\frac{Roll\ angle\ wihout\ modifications\ (\varphi_{without})}{Roll\ angle\ with\ modifications\ (\varphi_{with})}$$

Mechanism Effectivemeness

Additionally, the equation for roll-angle is given by (7). Where the distance between the roll center and the center of gravity (Δh). The relation between the roll angle and the distance between the roll center and center of gravity is directly proportional: As this distance decreases so does the roll angle (φ) of the vehicle - given all else is constant. The mechanism's effectiveness and roll angle equations can be simplified to produce (8), an inequality, given the vehicle's mass (m), the lateral acceleration of the body (g), and the front and rear roll stiffness of the front and rear axles ($c_{Ro,f} + c_{Ro,r}$) are constant.

$$\varphi = \frac{m \cdot g \cdot \Delta h}{c_{Ro,f} + c_{Ro,r}}$$

Equation 7: Roll-Angle

$$\Delta h_{with} < \frac{\Delta h_{without}}{1.25}$$

Equation 8: Center of Mass

Equation 8 stipulates that the distance between the roll center and the center of gravity () - or lever arm of the body around the roll axis - with adjustments made (Δh_{with}) to the position of the roll center or the center of gravity, must be lower than the quotient of the lever arm of the body around the roll axis without any adjustments ($\Delta h_{without}$) and 1.25/ the base effectiveness parameter. For example, if the lever arm of the body around the roll axis without any adjustments of the stock vehicle/ unmodified vehicle is 10 millimeters, the length of the lever arm must be less than 8 millimeters in order to have an mechanism effectiveness of greater than 1.25. This inequality will be used in the placement of the roll center and center of gravity in order to ensure that the vehicle suspension parameter is met.

6.2 - Vehicle Mechanisms System Failure Modes and Effects Analysis

It's important to recognize potential points and modes of failure of a system. The table below identifies a few possible modes of failure of the modifications made to the vehicle, details the impact the failure could have on the system or surroundings, rates its severity (low, medium or high), details how the failure could be detected, rates how detectable the failure is/ would be (low, medium or high), and suggests possible actions that could be taken in order to mitigate or avoid the failure.

Table 6.2.1: Potential mechanical system failure modes and effects analysis

Component	Failure Type	Potential Impact	Severity	Detection Mode	Detectable	Recommended Actions
Motor mount	Deformation due to heating from the	Power transmission increased inefficien	Med	Belt tension checks before and after the	Med	Insulation material to remove thermal contact between the motor and the

	motor	cy or failure		operation		mount and/or heat sink for the motor
GT2 pulley	Loose on the motor drive shaft	Power transmission inefficiency or failure	Low	Free movement of the pulley on the motor shaft	High	Use of thread-locking chemical solutions on the pulley set screw
Mounting screws	Loosening due to vibration	Component detachment from vehicle	Med	Loose screws	Med	Use of thread-locking chemical solutions on all important screws
Motor	Overheating	Loss of power after damage	Med	Temperature gauge for the motor	Low	Motor heatsink or power throttling

6.3 - System Status Indicator

Designing hardware for a system status indicator involves creating a robust, reliable, and user-friendly interface that communicates the operational state of a system. We explored the essential components and considerations for developing such an indicator, from choosing appropriate sensors and signal processors to selecting display technologies and ensuring seamless integration with the overall system architecture. By focusing on factors like power efficiency, scalability, and durability, we will outline best practices and practical strategies to build a hardware solution that is both functional and aesthetically aligned with user expectations. Below is a schematic design of the Printed Circuit Board (PCB) of that module.

Designing hardware for a system status indicator involves creating a robust, reliable, and user-friendly interface that effectively communicates the operational state of a system to its users. The development of such an indicator requires careful consideration of various components and design principles to ensure functionality, durability, and seamless integration into the larger system architecture.

We delved into the essential elements of hardware design for a system status indicator. We explored the process of selecting appropriate sensors and signal processors to detect and interpret the system's real-time state accurately. We examined the various display technologies available and evaluated their suitability based on our requirements. Some key design considerations are power efficiency to minimize energy usage, scalability for future enhancements, and durability to ensure long-term performance in various operational environments.

Below is the start of a detailed schematic of the module's Printed Circuit Board (PCB). It is currently not finished. This design will serve as a blueprint, showcasing the integration

of components and the layout considerations that contribute to the efficiency and effectiveness of the final product.

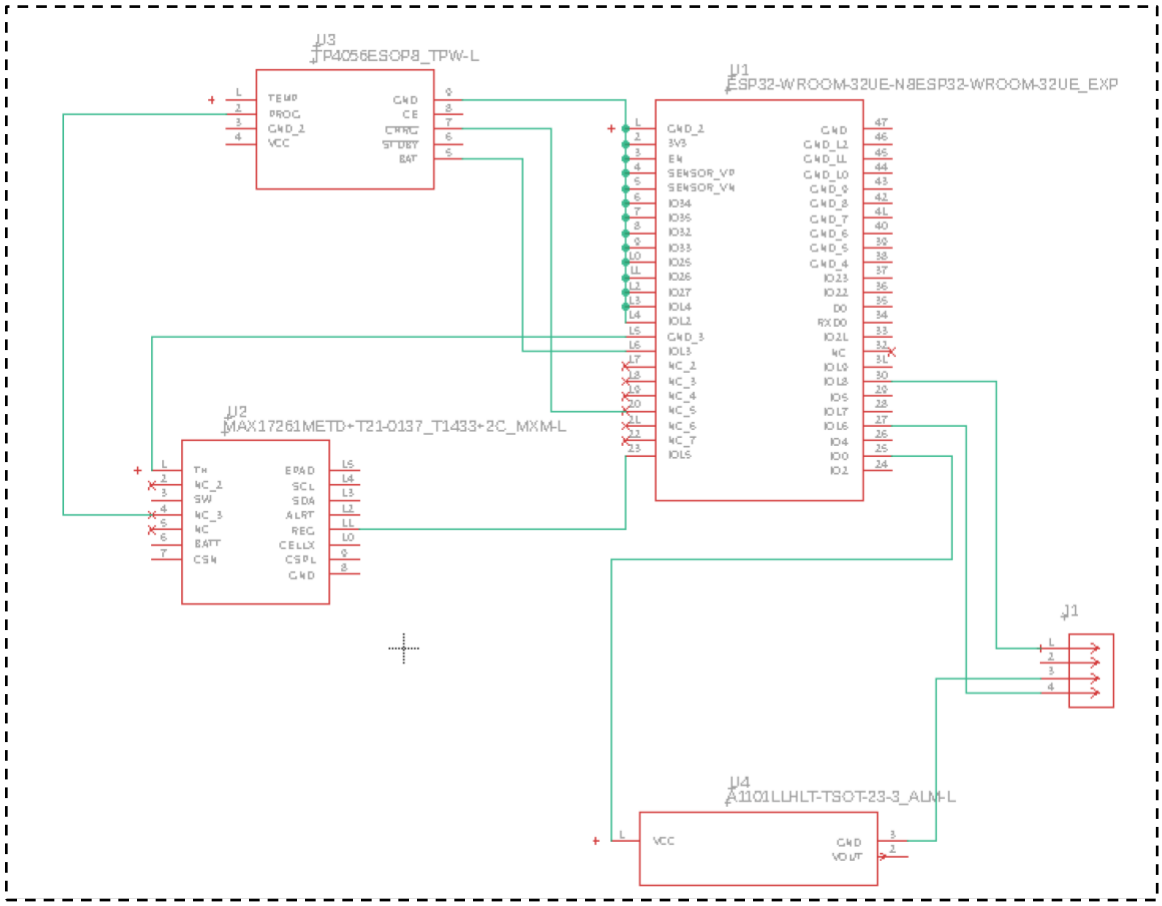


Figure 6.3.1: System Status Indicator Schematic

This is the schematic of the system status indicator subsystem pcb. It houses all the necessary on board peripherals for the subsystem and boasts the required connections in order for the subsystem to operate accordingly. It will also be connected to the system status indicator secondary pcb located on the vehicle chassis, allowing the subsystem to acquire data accordingly.

6.4 - Power Management System

The Power Management System (PMS) serves as the backbone of any advanced vehicle, ensuring not only the seamless distribution of power to various subsystems but also the continuous monitoring and analysis of power usage. This system plays a dual role: efficiently supplying energy to meet the diverse demands of the vehicle while simultaneously acting as a comprehensive diagnostic tool by tracking real-time power statistics. These statistics are critical for maintaining operational efficiency and troubleshooting potential issues, as they provide valuable insights into the health and performance of the power network.

One of the key features of the PMS is its ability to communicate effectively with the remote indicator subsystem. By transmitting power data—such as voltage levels, current consumption, and fault detections—the system enables centralized monitoring and control. This functionality ensures that users or automated systems can make informed decisions to optimize energy use, prevent overloads, and enhance overall system reliability.

We explored the foundational aspects of the PMS, focusing on its core components, input management strategies, and the integration of sensors and controllers that facilitate precise power monitoring. Emphasis will be placed on the design considerations that ensure scalability, robustness, and energy efficiency while addressing challenges like thermal management and fault tolerance. Below is the basic schematic functionality for the Power Management System inputs.

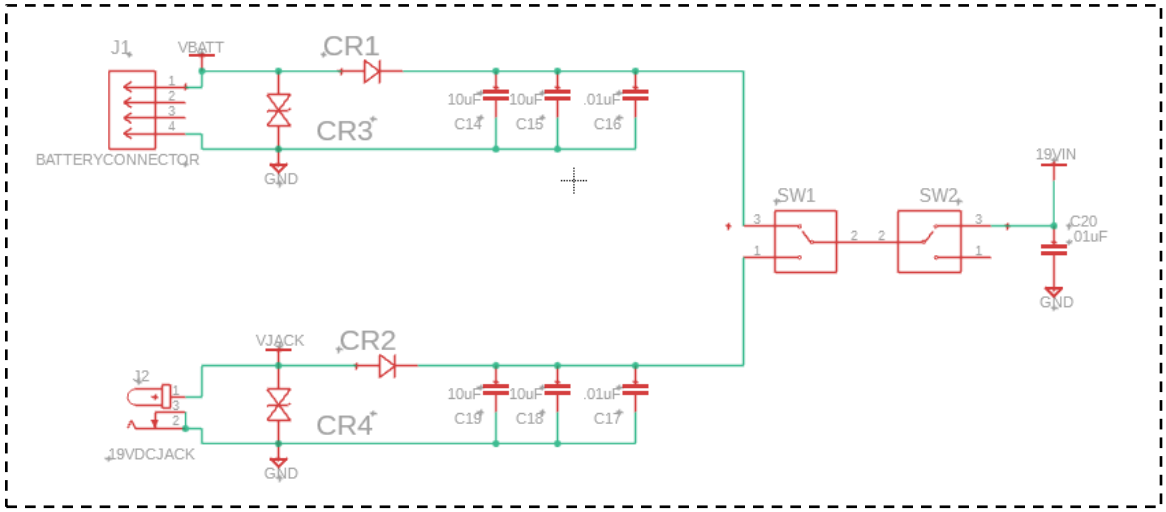


Figure 6.4.1: Power Management System Input

As illustrated in Figure 6.4.1, the system is designed to accommodate two primary power input options: a Lithium Polymer (LiPo) battery pack and a 19V DC jack. These two inputs offer flexibility in powering the system, ensuring compatibility with various operational scenarios and user preferences. The selection between these power sources is managed via switch SW1, which acts as a manual input selector. This allows users to seamlessly transition between the LiPo battery and the DC jack depending on availability or specific use-case requirements, such as mobile versus stationary operation.

Additionally, switch SW2 serves a crucial role as the system's master power control. By toggling this switch, users can either enable or disable power delivery to the entire system, effectively acting as a global on/off mechanism. This ensures a straightforward and reliable way to manage the system's operational state, enhancing both user convenience and power management.

Once the power source is selected and SW2 is engaged, the 19V input is distributed to all onboard power regulators. These regulators are critical components, as they ensure that the incoming voltage is stepped down or adjusted to meet the specific requirements of various subsystems. The regulation process guarantees stable and efficient power delivery,

protecting sensitive components from potential damage due to voltage fluctuations or overcurrent conditions.

To provide a clearer understanding, a detailed schematic is presented below, illustrating how the 19V input is integrated into the system's power regulation architecture. This schematic highlights the routing of power through multiple regulation stages, emphasizing the modularity and scalability of the design. By following this approach, the system ensures both flexibility in input management and reliability in power distribution, forming the foundation for a robust and efficient power management solution.

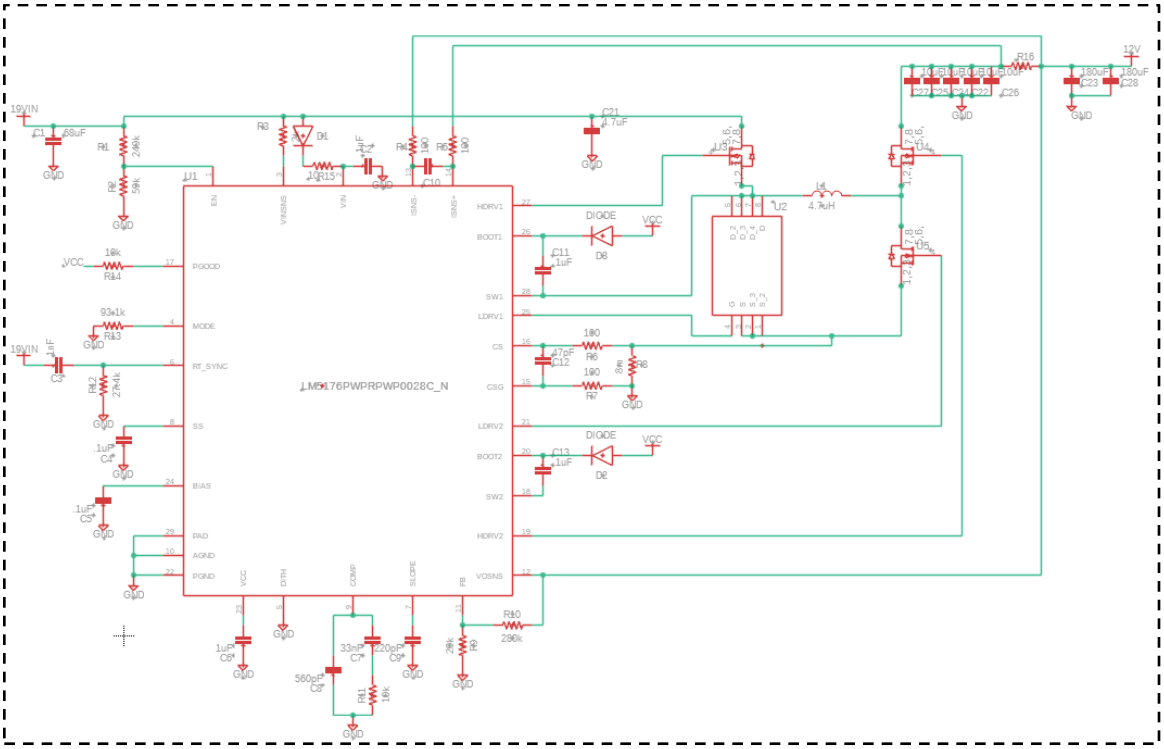


Figure 6.4.2: 12V Buck-Boost Power Regulator

This schematic is adapted from the reference design provided in the LM5176 datasheet, configured to produce a stable 12V regulated output. Under standard circumstances, this 12V output would be directly supplied to various peripherals requiring a 12V power source. However, in the context of our Power Management System, the design includes an additional layer of functionality to enhance monitoring and control.

Before reaching the peripherals, the 12V output is routed through a dedicated power monitor. This power monitor serves a critical role, continuously tracking key metrics such as voltage levels, current consumption, and potential faults. This data is essential for ensuring that the power delivered to the peripherals is both stable and within safe operating limits.

Moreover, this monitoring strategy is not limited to the 12V regulator alone. Each power regulator in the system is equipped with its own dedicated power monitor, as illustrated in the schematic below. This architecture allows for granular tracking of power statistics

across all regulated outputs, providing comprehensive oversight of the system's power distribution network. By incorporating these power monitors, the Power Management System gains the ability to detect anomalies, optimize energy use, and enhance the overall reliability of the system.

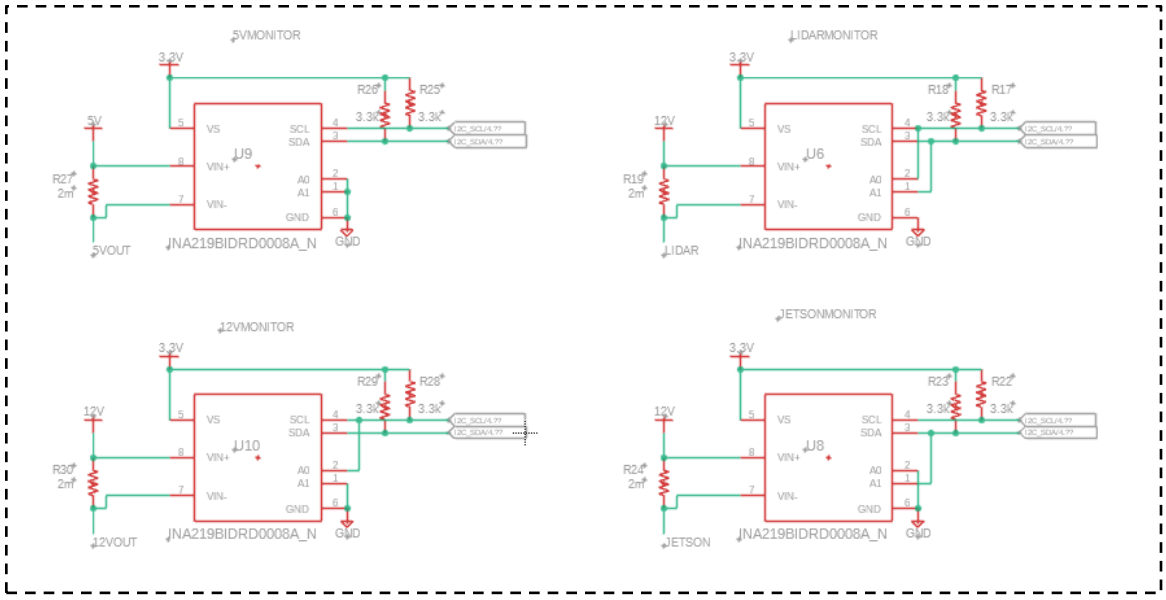


Figure 6.4.3: Power Monitors for Each Output

The Power Management System requires three distinct voltage regulator outputs: 12V, 5V, and 3.3V. Among these, only the 12V and 5V outputs are actively monitored, as the 3.3V output is directly utilized to power the monitoring framework itself. This framework includes four power monitors, implemented using the INA219 current sensors, and the central microcontroller, the ESP32.

The INA219 sensors play a crucial role in tracking the current and voltage of the 12V and 5V outputs, providing real-time feedback to ensure the system's stability and performance. These sensors communicate with the ESP32 microcontroller using the I2C protocol, a highly efficient two-wire communication standard. By utilizing only two lines—SCL (clock) and SDA (data)—the I2C setup simplifies connectivity and minimizes the number of GPIO pins required.

In this design, the SCL and SDA lines are routed to the ESP32's GPIO pins IO23 and IO22, respectively. This configuration ensures seamless data transmission between the power monitors and the microcontroller, allowing for precise measurement and monitoring of the power being delivered to the system's subsystems. The schematic representation of this setup is shown below, illustrating how the components are interconnected to form a robust and efficient power management network.

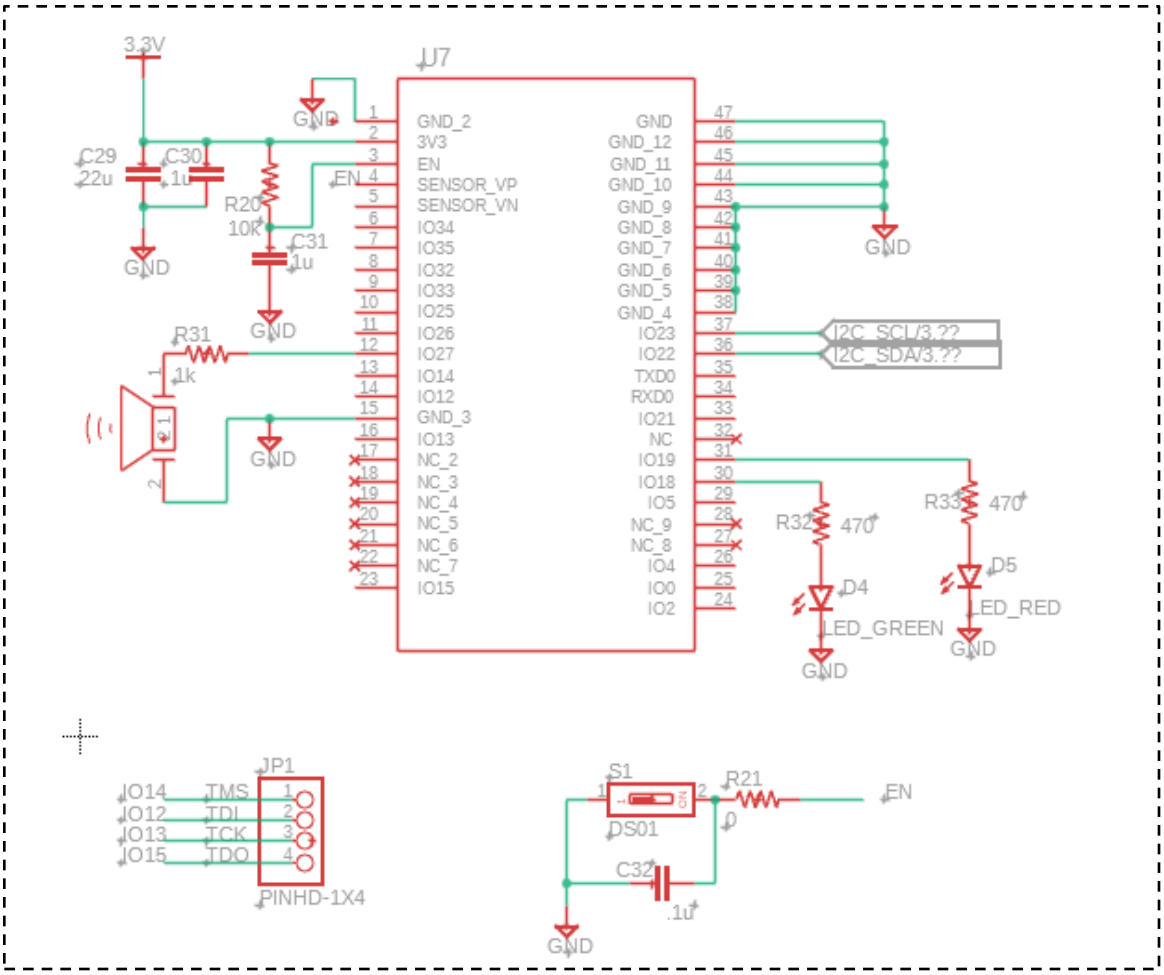


Figure 6.4.4: ESP32 Microcontroller

The necessary connections to the ESP32 microcontroller, as depicted in Figure 6.2.4, are designed to enable efficient programming, communication, and control within the Power Management System. Key connections include:

- JTAG Connector: This interface is utilized for programming and debugging the microcontroller, allowing developers to upload firmware and troubleshoot the system effectively.
- I2C Lines: The I2C lines facilitate communication with the power monitors and provide scalability for additional peripherals that may require communication in the future. These lines are essential for the microcontroller to gather real-time power data and maintain system oversight.
- On/Off Switch: Connected to the EN (enable) pin, this switch allows the microcontroller to be powered on or off as needed, ensuring energy efficiency and controlled operation.
- 3.3V Power Supply: This connection provides the necessary power to the microcontroller, ensuring its operation and enabling it to act as the central controller for the Power Management System.

Once the power monitoring process is complete and the data is read and processed by the ESP32 microcontroller, the final stage of the Power Management System involves distributing power to the necessary peripherals. The schematic below illustrates all the connectors dedicated to supplying power to various components of the vehicle. These include:

- LiDAR Sensor: Two connectors are provided to accommodate potential changes in sensor selection, offering flexibility in sensor integration.
- Jetson Xavier: This high-performance module requires dedicated power for advanced computation and processing tasks.
- Vehicle Power Outputs: Dedicated 5V and 12V outputs ensure that the vehicle's power requirements are met, supporting subsystems and ensuring reliable operation.

This design ensures a streamlined, efficient power distribution network while maintaining the ability to monitor and manage power usage effectively across all connected devices.

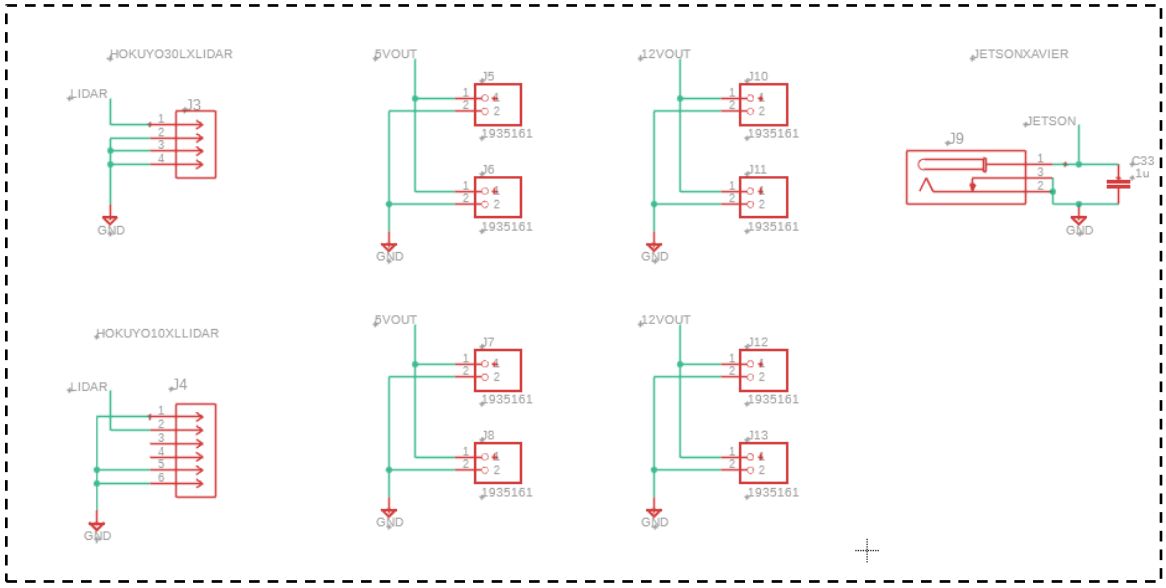


Figure 6.4.5: Peripheral Connectors

The schematics for the Power Management System, as presented, are subject to modification and refinement based on the evolving requirements of our group. Potential adjustments may include the addition of test points and possibly more connectors to accommodate any new peripherals or components that require power from the system.

Adding test points will be particularly critical for the testing, debugging, and verification phases. These test points will enable direct measurement of voltage at each power regulator, allowing the team to confirm that the regulators are outputting the correct voltages as intended. This feature will enhance our ability to troubleshoot issues, validate system performance, and ensure compliance with design specifications.

Furthermore, the flexibility to include additional connectors ensures that the Power Management System remains adaptable to future needs. This could involve supporting

additional sensors, modules, or other power-dependent subsystems that may be integrated into the vehicle over time. By proactively planning for scalability and diagnostics, the system is designed to support both current requirements and potential future expansions efficiently.

7 - Software Design

The software design for our autonomous vehicle serves as the backbone of its functionality, enabling seamless coordination between perception, planning, and control systems. This design encompasses a structured approach to managing the vehicle's complex behaviors and ensuring responsiveness to its environment. Using ROS2 as our foundational framework, the software integrates modular components to achieve robust and scalable performance.

At its core, the control software is divided into five distinct components: the Initialization Controller, Behavior Tree, Exploratory Planner, State Lattice Planner, and Speed/Heading Controller. Each component plays a critical role in managing specific aspects of the vehicle's operation, from starting up systems to generating and executing motion plans. By leveraging ROS2's capabilities, including its support for real-time data processing and multi-node communication, the software is designed to facilitate reliable and adaptive control of the car, whether navigating structured environments or exploring unknown terrains.

7.1 - Car control

As mentioned before, the design of the software controlling the car can be broken down into 5 distinct components:

1. Initialization controller
2. Behavior tree
3. Exploratory Planner
4. State Lattice Planner
5. Speed/Heading Controller

Further, we are using the ROS2/Nav2 stack for our vehicle’s navigation. The Nav2 framework is conceptually broken down into a set of ROS 2 action servers, with each action server having access to a number of registered plugins which action requests can be routed to.

For example, if the behavior tree or Nav2 parameters determine that the robot should navigate to a waypoint using MPC, when the controller server receives a new route it will pass it along to the MPC plugin which will run the appropriate computations. This modularity is crucial for our project as it enables us to use the robust and well-tested skeleton of the Nav2 package as an organizational tool of the components of our software while maintaining the flexibility to develop the implementations ourselves.

The three key basic action servers which we will use are the behavior tree navigation server, the planner server, and the controller server. The behavior tree navigation server hosts the

top level behavior tree, which can be configured in the Nav2 settings file. This is the server that will host our system's behavior tree, and in turn determines whether the system should be exploring or using the state lattice planner.

The planner server hosts global planner plugins whose job it is, in most cases, to turn the desired goal pose into a path that the robot can follow. In the case of our exploratory planner and state lattice planners, we will be including additional code that has them constantly re-selecting the target pose as the robot continues to move down the track to ensure it doesn't enter into a move-stop-move-stop pattern. The controller server hosts the plugins that convert from the path returned by the planner server to the velocity command to send to the vehicle.

For most projects using the ROS2/Nav2 stack, this is simply a pre-implemented trajectory controller, but since we are operating our PixHawk in offboard control mode we will need to implement low-level controls on the speed and heading as well to ensure there are no discontinuities that would cause erratic steering. For this reason, we are also creating our own custom plugin for the controller server.

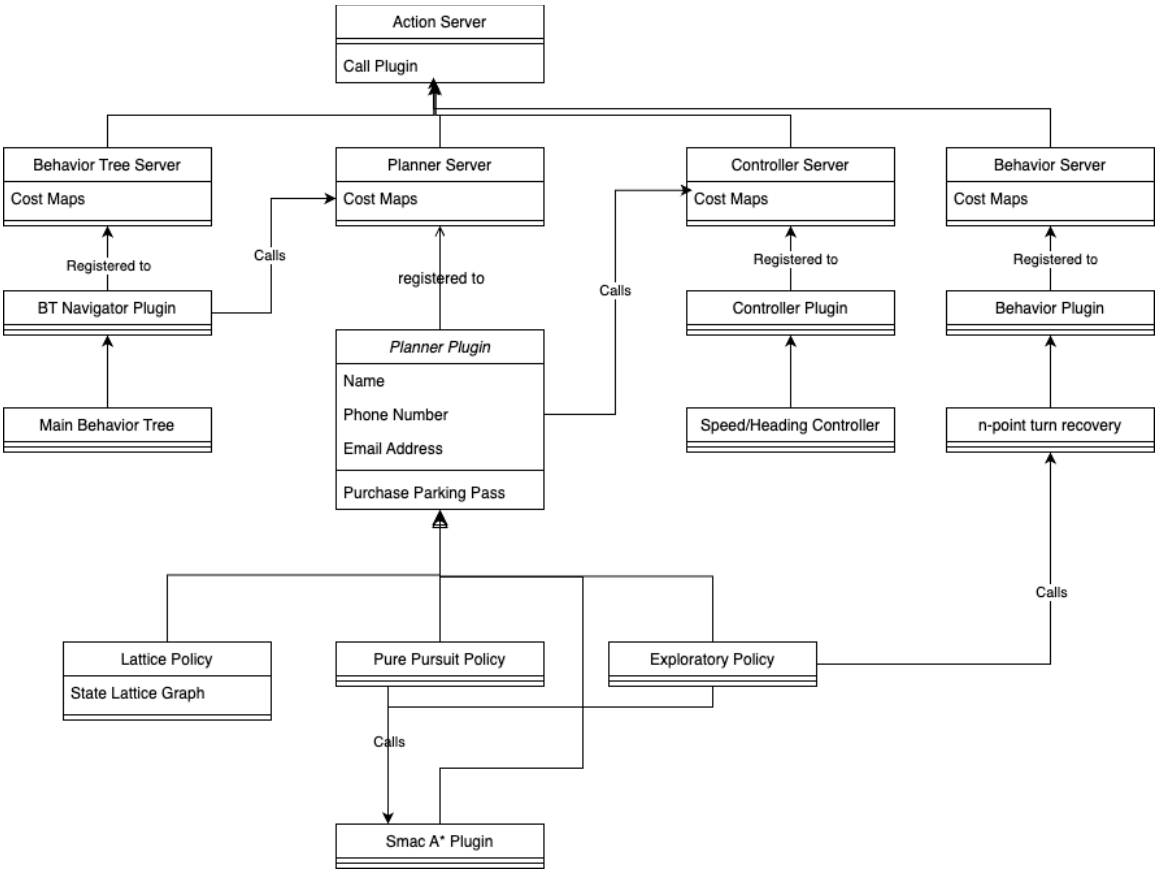


Figure 7.1.1: Vehicle Control Diagram

7.1.1 - Initialization controller

The initialization controller is the top level of the racing control program, and is above the only part of the system that doesn't directly fit into a category defined by Nav2. This is

because it has to handle hardware initialization steps that are inherently sequential and much easier to implement using Python than c++; one of those steps is also initializing Nav2 once all of the required topics are being published, requiring it to exist outside of that ecosystem.

As an independent ROS node, it automatically runs through an internal checklist of sensors and communication channels, and then loads the required packages (Slam Toolbox and Nav2) At this point, it displays a successful setup message on the indicator board and waits for user input on the SSH terminal from which the node was launched to continue with the race. At the start of the race, the user takes the appropriate action, and the initialization controller publishes the message indicating that the race has started.

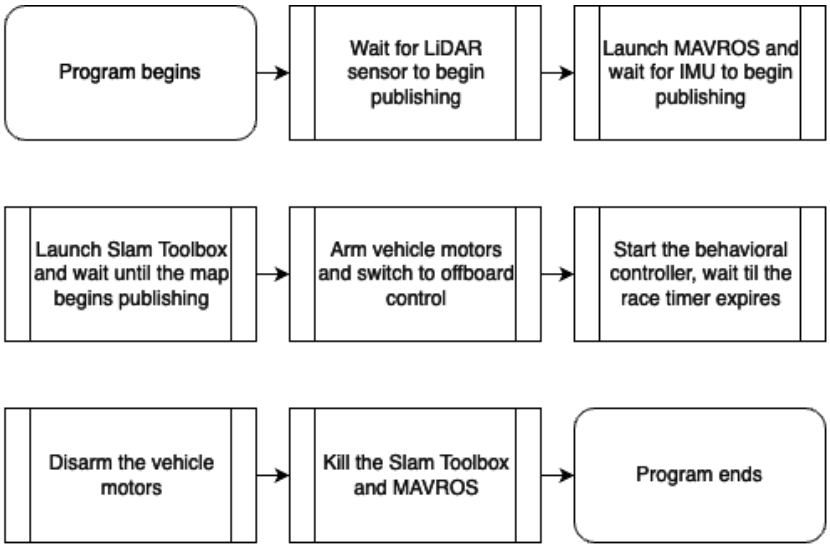


Figure 7.1.1.1: Initialization Controller Diagram

7.1.2 - Behavior Tree

Following Nav2 convention, the portion of the car control software that does the majority of the heavy lifting in terms of determining what the car is supposed to be doing at any point in time is a behavior tree. In Nav2, behavior trees can be specified just like behavior trees anywhere else, but are able to directly hook into the Nav2 action servers to start long-running tasks. We leverage that heavily, and cast the two major phases of the race within our architecture as separate global planner plugins. The behavior tree then determines when each one is used. This division into two sections allows us to achieve the best possible speed while limiting penalty points for small brushes with the walls or other cars. The first of these two divisions is the exploratory planner, which is used before the map is done being explored to safely drive the car around in an unexplored environment. The second of these is a state lattice-based planner, which takes in a graph of states and finds the most efficient route from the car's current position in the map to some goal position according to the weights assigned to each of the edges.

The behavior tree will be implemented as an XML file and loaded into the Nav2 Behavior Tree Navigator server using the Nav2 configuration file. When the initialization controller

transmits the message to begin navigating around the race track, the ticks will first flow from the initialization down to the compute state lattice action. Nav2's behavior trees are built on top of BehaviorTree CPP V4, which supports asynchronous actions [67]. We use one of these such actions to start a script which runs in the background of the program and begins creating the state lattice. The state lattice is a graph composed of sequential rows of nodes perpendicular to the direction of the track placed at equal intervals, with splines from each node along one row to each node along the next row in the direction of the race representing kinematically feasible paths for the car. Once that node is live, the ticks flow first to the check if the map is complete or not and then to the exploratory policy, which will continue to run uninterrupted until the map complete check becomes true, at which point the ticks will skip that branch and flow to the compute race line action. Because we can't afford to have the car stop and wait for the race line to be computed we use another asynchronous action to call a script which begins computing the race line in the background. In the meantime, the car switches smoothly over to the lattice policy, which it uses for the duration of the race. The races we will be competing in are always time based, with scores determined by a multiplier of the number of laps completed in the allotted time less any penalties for "touching" (when the car makes contact with an obstacle or track wall but not strongly enough to move it or stop itself; a glancing blow) or crashing (when the object is noticeably moved or the car becomes unable to move without corrective action). As such, for the remainder of the program the "is race over" check is monitored until it becomes true, at which point the navigation activity stops. The initialization controller will receive notice that the action it requested has completed, and automatically runs the shutdown routine.

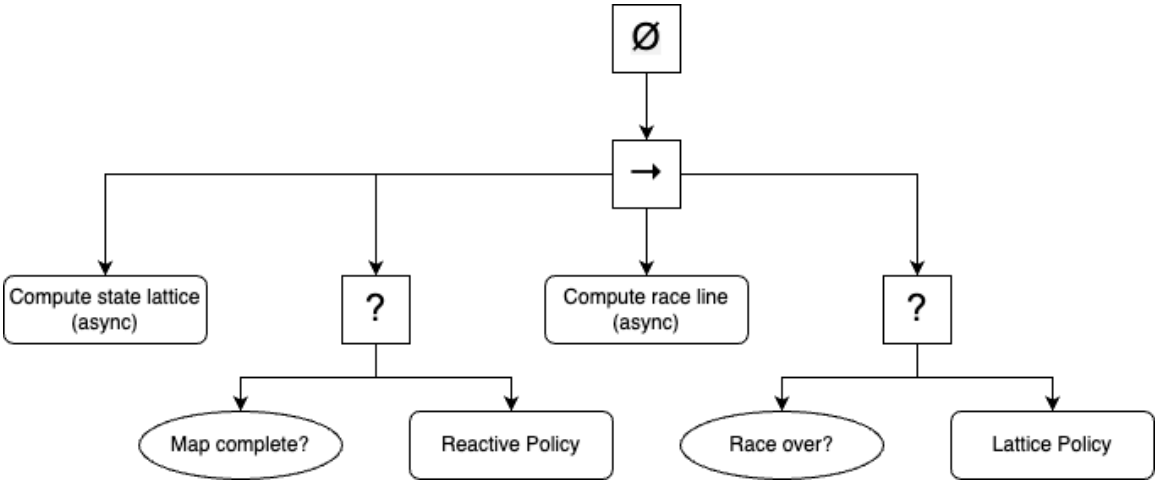


Figure 7.1.2.1: Behavior Tree

7.1.3 Exploratory Policy

The exploratory policy is used to direct the car around the unexplored track until the code corresponding to the "map complete" check in the behavior tree reports that loop closure has occurred in the slam toolbox pose graph, indicating that the car has completed a lap in the track. Before that happens, the map will effectively resemble a tube with walls and both ends open corresponding to the limits of what has been mapped. For the purposes of the exploratory policy, we call the set of grid spaces that have both mapped and unmapped

neighbors boundary spaces, and divide them into two sets: the forward boundary set and the rear boundary set. The rear boundary set refers to the set of boundary points that were behind the car when it was in its initial position.

The exploratory policy must be capable of planning paths that always direct the car in the direction of the flow of the race while effectively avoiding obstacles and being able to recover from dead ends. In order to meet the first criteria, as soon as the exploratory policy action is activated and the forward and rear boundary sets are calculated, every point in the rear boundary set is marked as ineligible to become a goal position. This means that the first point selected is guaranteed to be from the forward boundary set, moving the forward boundary further in the right direction and setting things in motion in the correct direction. To further improve safety, to choose our goal point out of the set of allowed boundary points, we pick the one furthest from any obstacle at a given point in time, guaranteeing a path that, in the absence of other vehicles, will stay in the center of the track. If there is a tie, the path goes to whichever one is closest to the vehicle at that time. If the car does end up going down the wrong path of a fork in the track and into a dead end, it will progress until it has fully explored the dead end and realizes there are no more boundary points there. At that point, the system will pick a path on the right side of the fork and attempt to route there. One of two cases can occur.

1. The dead end is wide enough for the car to simply make a u-turn and drive out of it.

2. The dead end is not wide enough for the car to make a u-turn.

If planning fails to generate a path which escapes the dead end using a u-turn, we call the n-point turn behavior plugin we design for this situation. In addition to the "core" action servers provided by Nav2, they provide a behavior server where plugins that are designed to direct a vehicle out of a failure state can be placed. Once the plugin is called, it determines the radius of the circle with its center on the car's base_link frame such that the circle touches only the wall point closest to the car. From there, it makes the maximum possible left turn until it reaches the edge of the circle (minus some safety margin), then reverses with the wheels to the right until it is again at the edge (minus some safety margin), and continues this process until the reported IMU heading is within a certain threshold of the direction of the planned path. This potentially many point turn ensures that the car will almost always be able to rotate itself out of a tight dead end and return to the race without making contact with the track walls or obstacles.

The one remaining failure case for this design for the exploratory policy, then, is the situation where loop closure is not achieved but there are no more boundary points anywhere in the map. For this to happen, the map is either not a loop, which we don't have to worry about since we can guarantee that the racetrack is a loop, or SLAM failed to correctly create the map. In this case, we display a warning in all red on the indicator board declaring that mapping has failed. During testing, this will be crucial in alerting the user to the source of the program so they know to open Rviz and see what the car is seeing. During the actual race, it will still be a nice feature to know why the car is behaving the way it is. Once the warning is displayed, the SLAM node is reset and the car begins planning again from the first laser scan it received. This is unfortunately necessary because, while the

SLAM node stands a decent chance of eventually correcting itself as more data is collected and the car is driven around, in order to drive around without the lattice complete the car needs to have the boundary set available to determine the correct direction of movement.

The primary differences between this approach and the default for planning in Nav2 is that instead of supplying a single goal point, which the planner then replans continuously for until it's reached, this planner selects the goal points itself and will run forever until it is either stopped externally or there are no more boundary points. However, to actually create the path once the goal point has been selected we do fall back to the thoroughly tested Nav2 standard. Specifically, we use the Smac Hybrid-A* planner since it supports ackermann steering which this car uses [65]. In order to use it, we include the corresponding plugin in the planner server as well, and simply call it as needed from our own custom plugin. Once it returns the plan, we pass that along to the controller server which processes it into movement commands and conveys it to the motors.

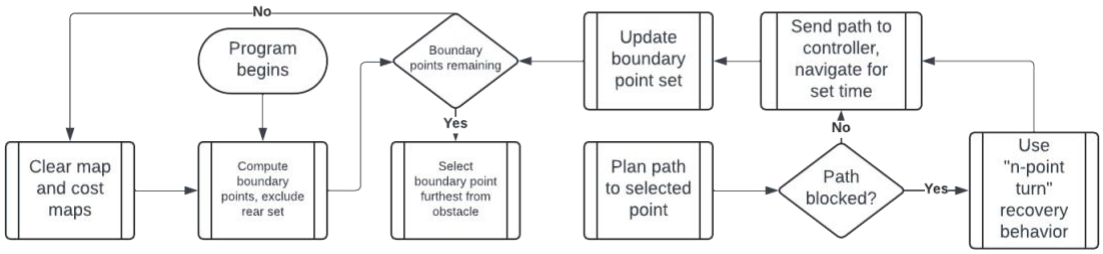


Figure 7.1.3: Exploratory Diagram

7.1.4 Lattice Planner

The state lattice planner will control the motion of the car for the vast majority of the race, and is absolutely essential for enabling the car to move at competitive speeds. The core concept of the planner relies on the idea that for a given race course there is a single ideal raceline representing the trajectory around the track that allows the racing vehicle to have the best lap time. This is tempered by the fact that this is only true in the absence of other vehicles and obstacles, which won't be true in our F1Tenth race. Thus, there must be "second best" alternatives available to choose from. Since we can't know ahead of time where and when an obstacle is going to appear, any planner that takes these concepts into account must therefore be able to have a second best option ready anywhere and any time during the race. This is achieved using the state lattice planner described earlier, which uses a conformal state lattice of possible trajectories given the car's kinematics to select a path which adheres as best as possible to that race line. When an obstacle is detected, the set of state lattice graph segments overlapping the footprint of the obstacle (or within some pre-set safety distance of it) are made unavailable to the planner, preventing it from choosing a path that results in a collision. Then, n frontiers ahead of the car a virtual node is created which is connected to all the nodes along that frontier with a cost of 0. At that point, a shortest path algorithm will run, finding the optimal path from the car's current position to the virtual node. However, despite representing the same thing physically, the path through the graph is very different mathematically than the splines which are generally used as the representation of a path. Furthermore, depending on slight variations in the

vehicle's instantaneous heading at that time compared to the heading of the selected route, adjustments must be made to ensure the resulting spline is continuous both with itself and the real world pose of the car. In order for the trajectory controller to use the path to compute the velocity profile, it must be converted to a continuous spline. This is done by adjusting the boundary conditions to "stitch" the selected splines together, and then using a smoothing function such as a weighted average to ensure the transitions have no unexpected jerks. This finished path geometry is then sent to the controller server to be executed on the car, where it will be first transformed into an optimal velocity profile and then executed by a low level planner that computes the actual steering to be executed on the hardware [64].

Calculating the race line involves two steps, calculating the geometry of the path and calculating a corresponding velocity profile for it. Together, these form a trajectory where each point along it has a vector value. The math involved in doing this well is very complicated, and as a result we are using a pre-built implementation to compute this which takes into account factors such as friction coefficients for the tracks and wheels and the torque of the vehicle's power train [66]. This level of detail means that our ideal race line will be as optimal as possible, enabling the vehicle to move aggressively while limiting the risk of a collision that would be present if we used a solver that ignored some of those variables. On a mathematical level, what the solver is doing is iteratively solving an optimal control problem to minimize the time it takes for the vehicle to drive around the track while using different levels of acceleration at different points.

In order for the state lattice planner to function each of the state lattice graph segments must have a cost assigned to it. Without that, the shortest path algorithm would have no way of knowing how much shorter one path is versus another, and would not be able to produce a plan that guarantees any sort of optimality with respect to travel time. Ideally, that cost will be determined by how far the state lattice segment is from the corresponding segment of the race line optimized with the vehicle's kinematics in mind. However, after loop closure is achieved and the system moves into the stage of the race which uses the lattice planner, it will have just started the calculation of the race line. Even though the algorithm is well optimized, this complicated process cannot happen instantly, so when the state lattice planner activates the state lattice must have costs assigned to its segments that are derived from something else. To account for this, as the state lattice is being created during the exploratory control phase of the race, the costs are calculated as distance from the centerline of the race track. This is not an optimal objective, but it is suboptimal in terms of speed instead of safety which is better for us than the other possibility. Since the shortest path algorithm simply runs with high frequency and takes the most recent version of the graph as input, we simply wait until the race line is optimized, calculate a new set of costs using the race line as the ideal trajectory, and replace the version the shortest path algorithm references with the updated one. When it runs again, it will pull that version in, and the car will immediately begin speeding up along the new ideal path.

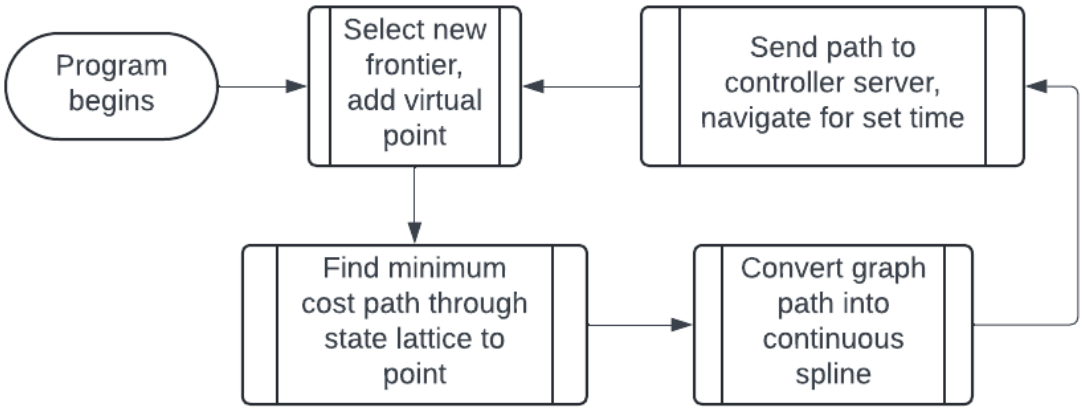


Figure 7.1.4.1: Planner Diagram

7.1.5 - Pure Pursuit Planner

While conformal state lattice planners are the state of the art in terms of small-scale autonomous vehicle racing, the large number of complicated optimization steps that must be undertaken to get good performance will make it very time consuming to implement. As such we have made it a stretch goal to ensure the success of the project even if unexpected road blocks prevent this technique from working as well as it is intended to. Even without implementing the complex logic to compute and utilize the conformal state lattice graph, we will have a complete map and an optimized race line by this point in the race, which is more than sufficient for planning. As such, our initial implementation will be a pure pursuit planner. When this planner is activated, it will start off by pursuing points along the centerline of the track, and after the race line is computed it will switch to following points along the race line. Much like the state lattice planner iteratively selects new frontiers a certain distance in front of the car to plot routes to, the state lattice planner iteratively picks points along whatever line it is following for the same purpose. Similar to the exploratory policy, we will use the Smac Hybrid-A* planner to create the path geometry, with the rest of the plugin acting as a way to monitor its progress and update the set point as is appropriate [65].

7.1.6 - Controller servers

Once the planner computes a path geometry, the vehicle needs to know how fast to travel on it to ensure it can move as fast as possible without risking drifting into a wall or other obstacle. We use a forward backward solver that, given a path, computes an optimal velocity profile given the car's physical information (e.g. weight and wheel friction coefficients) that runs along that path while minimizing the time it takes to reach the end. This works by first making a pass in the direction of the race computing acceleration limits at each point, then backwards computing deceleration limits. Combining these limits, the maximum speed at each point can be found respecting the limits of the powertrain and the friction of the car to prevent unwanted drifting. Unfortunately, we are not able to calculate these values ahead of time like we can the geometric portion of the graph because since this profile is only relevant to the section of the track the car is currently on the profile must

blend smoothly with the car's current motion, which necessitates including the vehicles current speed. Furthermore, we are not able to use the velocity profile created by the race line optimization (even though it does create a velocity profile) because it only applies to the car along that specific line, and attempting to extrapolate that to the many different race lines embedded in the state lattice graph would not be possible. That said, outside of necessity computing the velocity profile online has some major benefits. Primarily, this is that the car can dynamically take advantage of overtake opportunities without having to program a dedicated subroutine to do so. Additionally, it can slow down well in advance if an obstacle happens to block its path. With this velocity profile complete, it can be combined with the path geometry to form our final trajectory for this point of the race [64].

Even though we now have a trajectory which includes the vehicle's speed, the realities of the real world mean that the car is guaranteed to deviate from the prescribed route and velocity profile by at least some amount. If this is the case and we ignored it, instead issuing the velocity command corresponding to where the car is closest to on the current trajectory, the car would be almost guaranteed to crash. For instance, if the current planning policy was to issue an update to the trajectory that accounts for an obstacle being detected but gets the current pose of the car wrong due to errors in the odometry, the car could be expected to start at a different steering angle than it actually is. Without another means of control, the car would steer wildly to match where it is told it should be, and could potentially spin out or collide with an obstacle. Such errors could also occur if the car slightly deviates from the path while driving along it before a new plan is issued simply because positioning is never perfect, especially during a high speed race. The solution is a local controller such as PID, MPC, or Stanley. This kind of controller simply attempts to keep the state of some system near a goal point without oscillation and with as rapid of updates as possible. In our case, this means keeping the values of steering and velocity as close to the velocity profile as possible while preventing any sudden jumps.

In some cases, flight controllers like the PixHawk that run firmware like ArduPilot or PX4 include such controllers internally which handle this behind the scenes. However, in order to give the controller commands using an external computer instead of a pre-set routine programmed through a ground control station, we need to set the PixHawk to offboard control mode. This takes the movement commands directly over the telemetry port and sends them to the ESC unmodified, meaning the car is more than susceptible to the kind of steering failure described earlier.

As with the exploratory and lattice policies, our implementation will come in the form of a Nav2 plugin. However, this plugin will be registered to the control server instead, where it can be called by either of our policies to ensure the car moves as we intend it to. Both the forward backwards optimizer and the local controller will be implemented here. Our implementation will come in two parts, one of which will be the core implementation and one of which will be a stretch goal. For the first implementation we're going to use a PID controller for managing the velocity of the car and a Stanley controller to manage the steering angle. This is a system that some of our team already has experience with implementing in other projects that is known to work well and be easy to debug, so it guarantees that this part of the project is not uncharted territory. However, because we are using two separate controllers to control the two core values of the movement command,

there is the possibility that the two values selected by the controller are not optimal together given the car's kinematics. Furthermore, PID controllers are intended for linear systems. This is because the proportional, integral, and derivative coefficients are constants that are optimized offline, and cannot change if the dynamics of the system changes. This works well for simple systems like controlling the temperature of a boiler, where the dynamics of restricting or allowing more airflow always have roughly the same effect. However, as the car navigates around tight corners versus on straightaways, the acceleration needs of the car are very different. A PID controller can be tuned to work very well on straightaways but may struggle as the car faces a hairpin turn, or vice versa. Thus, one of our stretch goals is to implement a model predictive control controller. This kind of controller is designed with the physical characteristics of the vehicle in mind, and uses that information to make predictions about the car's motion given different commands for steering angle and speed jointly. This way, it is able to find the value of these that minimize its cost function and produce the movement closest to what is desired. MPC controllers are very different from vehicle to vehicle, and as such we will have to do most of the mathematical heavy lifting for implementing this controller ourselves. Thus, we made it a stretch goal; after all, perfectly executing a poorly computed velocity profile is pointless when the velocity profile itself could be improved and executed adequately by a simpler controller.

8 - System Fabrication/ Prototype Construction

At this stage of our project, we are transitioning from the research and planning phase to bringing our autonomous vehicle design to life. Most of the parts have already arrived, and we've completed the groundwork to ensure that our prototype construction is as smooth as possible. Our goal now is to assemble, integrate, and validate the various components we've chosen, moving closer to a functional system.

This phase is crucial because it allows us to see how the theoretical aspects of our design translate into a physical working prototype. Along the way, we'll be identifying and resolving challenges, refining the integration of hardware and software, and preparing for rigorous testing. By the end of this phase, we aim to have a fully operational prototype that will serve as the foundation for further development and optimization.

The fabrication and construction phase marks an exciting turning point in our project, as we begin transforming our designs and plans into a tangible, working prototype. This phase is where all the research, component selection, and planning come together, giving us the first opportunity to see our autonomous vehicle in action.

This stage is critical for transitioning from theory to reality. While design and simulation have provided us with a solid foundation, it is only through building and testing a physical prototype that we can truly validate our concepts. During this phase, we'll identify and address any gaps between our theoretical model and practical implementation, ensuring that all components work seamlessly together.

Our primary goals for prototype construction are operational testing and system refinement. By assembling and integrating the various subsystems, we'll be able to test key functionalities, such as motor control, sensor performance, and software responsiveness,

in a real-world setting. Additionally, this phase will help us refine the overall system, making it more reliable and ready for further development and rigorous testing. It’s an essential step toward bringing our vision of an autonomous vehicle closer to reality.

8.1 - Vehicle Mechanical Systems

Assembling the mechanical system of the vehicle is a critical step in bringing our autonomous vehicle prototype to life. This process involves integrating the chassis, drive system, and mounting structures to create a stable and functional platform for the electronic and software systems.

We will begin by assembling the chassis, which serves as the foundation for the entire vehicle. The chassis is designed to be lightweight yet durable, capable of supporting the weight of all components while maintaining structural integrity during operation. Mounting brackets and fixtures will be attached to ensure secure placement of sensors, controllers, and other hardware. We will also ensure proper alignment and balance of the chassis to optimize stability and performance.

The drive system, consisting of motors, wheels, and drivetrain components, will be installed next. Motors will be mounted securely to the designated motor housings on the chassis.

For our suspension systems, the suspension will be adjusted to provide adequate ground clearance and shock absorption, while the steering system will be tested for smooth and precise motion.

We will also include provisions for mounting sensors and electronic components. Custom brackets or 3D-printed mounts will be used to secure cameras, LiDAR, or other sensors to the vehicle. These mounts will be positioned to maximize the sensors' field of view and functionality while protecting them from damage during operation.

Once the mechanical components are assembled, we will perform a series of checks to ensure everything is properly aligned and secured. This includes verifying the tightness of fasteners, checking for any mechanical interference between moving parts, and ensuring the wheels rotate freely without obstruction. Adjustments will be made as needed to prepare the vehicle for the integration of its electrical and software systems.

This step sets the stage for integrating the vehicle's electronics, sensors, and control systems, ensuring the mechanical system provides a reliable and efficient foundation for further development.

8.1.1 - Drivetrain

The major steps to creating the prototype of the new drivetrain would be to create CAD models of all the components to be used in the prototype, create an assembly with those components, verify that each component would fit well together and not interfere with other vehicle components, start part manufacturing if needed, and assemble the prototype system once all the parts are available.

Visualizing a design in CAD is a useful tool as it allows a designer to ensure that their design would would work spatially or aesthetically before the construction of a physical prototype. For CAD design, assembly and visualization, we will be using Fusion 360. The CAD process for the drivetrain will begin by creating or obtaining models of purchased or already existing hardware/components. This library of parts will be used later to assemble the vehicle in CAD. For the drivetrain, one of the major components is the KingVal 4300Kv brushless motor. Although CAD models for this motor are not readily available online, drawings of the motor are provided by their sellers. Our team has already used the KingVal's drawings to create a 3D model of the motor for future use. Additionally, we have sourced open-source models of the GT2 pulleys we plan to use for the timing belt, belt drive mechanism. The pinion gear that received power from the belt drive mechanism will also be modelled through measurements and analysis of the actual pinion gear. This pinion gear is Traxxas's stock 13 tooth motor pinion gear. Addition components for the new drivetrain such as the 3millimeter pinion gear shaft, bearings, and screws can be easily modelled or sourced online through sites such as GrabCad or McMasterCarr. The mount is also another major component of the drivetrain. Our plan is to design the mount to be secured to the chassis of the vehicle and have the motor secured to the mount using M3 screws on the mount's motor interface.

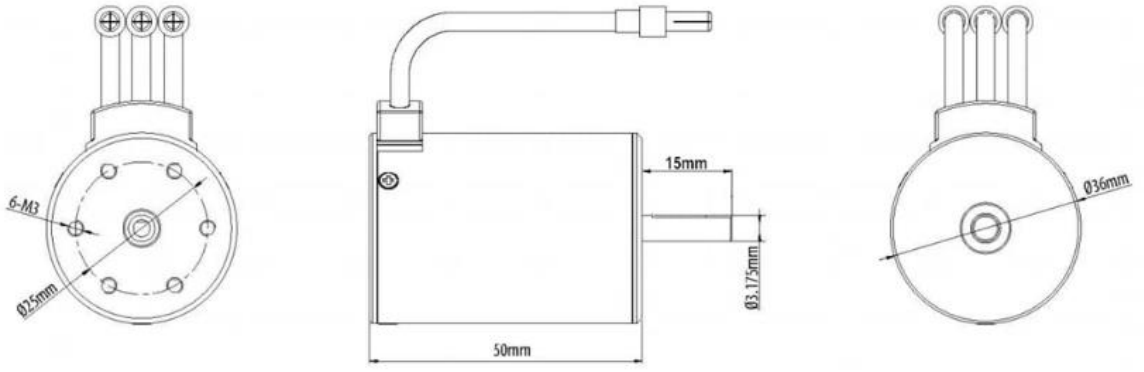


Figure 8.1.1.1. KingVal 3650 4300Kv motor drawing. Source: [62]

Once all the components have been sourced and added to the part library, an assembly of the vehicle can be created to assess fit and spatial harmony. This process will conclude once all the vehicle's power management components, power supply, autonomy sensors, computer, motor control boards, drive motor with its accompanying components, all mounts, and attachment solutions/screws and washers are represented on the CAD model. This assembly will then be used to ensure that each component fits well in the assembly and has plenty of space to operate optimally or as needed. The fully assembled model of the vehicle will also be useful in the event components need to be repositioned to meet mechanical, electrical, or software needs.

Part manufacturing can begin once a satisfactory vehicle assembly has been made. 3D-printing will be our primary prototype manufacturing method. For the drivetrain, the main part that will be 3D-printed will be the motor mount. The mount will be printed using PLA filament using printers available at the University of Central Florida Texas Instrument lab. Print settings such as part orientation and part infill will be determined in order to optimize

the part’s strength and weight. This process for all the parts that require 3D-printing is expected to take at least a couple of days. Once all the parts have been manufactured or purchased, the prototype vehicle can be assembled.

8.2 - Power Management System

The power management system is a critical subsystem that ensures the safe, efficient, and reliable distribution of power to all components of the vehicle. Proper assembly and integration of this system are essential for maintaining consistent performance and preventing power-related issues during operation.

We will begin by selecting and securely mounting the primary power source, which may include batteries or another energy storage device. For our project, we have chosen batteries that meet the voltage and capacity requirements of the vehicle's motors. Battery mounts or enclosures will be used to protect the batteries from physical damage and environmental factors such as dust or moisture.

The next step involves assembling the power distribution system, which includes:

- Voltage Regulators: These will ensure that each component receives the correct voltage, protecting sensitive electronics from overvoltage or undervoltage conditions.
- Fuses and Circuit Breakers: Safety features like fuses or circuit breakers will be incorporated to prevent damage in the event of a power surge or short circuit.
- Power Distribution Board (PDB): The PDB will centralize power distribution to various subsystems, such as motors, sensors, and processors. This ensures a clean and organized setup, reducing wiring complexity.
- Our custom power PCB will be installed during this phase. The PCB will handle tasks such as voltage regulation, current sensing, and power switching, ensuring optimal power delivery to all components. Connections to the PCB will be made using robust connectors to maintain reliability.

Proper wiring is essential for the power management system. During this step, we will:

- Use color-coded wires and heat-shrink tubing to organize and protect electrical connections.
- Route wiring to minimize interference with other components and prevent physical strain on connections.
- Verify secure and stable connections using locking connectors or soldered joints where necessary.

Once the power management system is assembled, we will conduct thorough testing to ensure all components function as expected. This includes:

- Load Testing: Verifying that the power source and distribution system can handle the maximum expected current draw without overheating or voltage drops.
- Voltage Verification: Using a multimeter to confirm that each subsystem receives the appropriate voltage levels.
- Safety Checks: Ensuring that fuses and circuit breakers activate as designed under fault conditions.
- For the power monitoring features we will integrate and test these as well. This will allow us to monitor power consumption in real time, aiding in diagnostics and performance optimization.

By completing these steps, we will have a robust and efficient power management system that supports the reliable operation of the autonomous vehicle, laying the groundwork for the integration of the control and sensing systems.

8.3 - Programming Main Computing Unit

The high-level software programming on the NVIDIA Jetson Xavier NX is a pivotal step in integrating the vehicle's perception, planning, and control systems. Using the Robot Operating System (ROS), we will create modular and scalable software that enables the autonomous functionality of the vehicle. Here's what we will do:

Set Up the ROS Environment

- Install ROS2: Install ROS2 Foxy optimized for the Xavier NX.
- System Configuration: Configure the Xavier NX to work with ROS2, including setting up environment variables and dependencies.
- Package Management: Create a structured workspace with separate packages for perception, planning, control, and other subsystems.

Perception System Development

- Sensor Integration: Use ROS drivers to interface with sensors such as cameras, LiDAR, and IMUs.
- Verify real-time data streaming through ROS topics.
- Sensor Data Processing: Use libraries like OpenCV and PCL for preprocessing sensor data (e.g., filtering, edge detection).
- Implement sensor fusion techniques to combine data from multiple sensors for more accurate environmental perception.
- Object Detection and Tracking: Integrate AI models (e.g., YOLO, TensorFlow) accelerated by the Xavier NX's GPU for detecting obstacles or lane markings.
- Publish processed data as ROS topics for use in planning and control.

Localization and Mapping

- SLAM Integration: Use existing ROS2 SLAM packages to generate real-time maps of the environment.
- Publish map and localization data for path planning.
- Positioning: Fuse IMU data using ROS2 packages like robot_localization for precise localization.

Path Planning and Obstacle Avoidance

- Path Planning Algorithms: Implement or adapt planning algorithms for generating feasible paths.
- Use ROS2 Navigation (Nav2) for high-level path planning.
- Obstacle Avoidance: Integrate real-time obstacle detection data into the planning process to dynamically adjust paths.
- Test reactive local planners (e.g., Dynamic Window Approach, TEB) to avoid collisions.

Control System Implementation

- Motor and Actuator Control: Write ROS nodes for controlling the motor and steering systems using commands from the high-level planners.
- Ensure compatibility with the autopilot using ROS2 message formats.
- Feedback Loops: Implement feedback control loops (e.g., PID controllers) to achieve stable and accurate motion based on sensor input and desired trajectories.

Communication and System Integration

- Inter-Node Communication: Define and standardize ROS message formats for efficient communication between perception, planning, and control nodes.
- Use ROS2 DDS (Data Distribution Service) for low-latency, real-time communication.
- Middleware Customization: Configure the ROS2 middleware for optimal performance on the Xavier NX, taking advantage of its GPU and CPU capabilities.

High-Level Software Testing

- Simulation Testing: Test high-level software in our simulation environment before deploying it on the vehicle.
- Real-World Validation: Perform field tests to validate the software’s performance under real-world conditions.

- Fine-tune parameters such as sensor calibration, planner configurations, and controller gains.

Performance Optimization

- GPU Acceleration: Leverage the Xavier NX's CUDA cores for computationally intensive tasks like AI inference or SLAM processing.
- Multi-Threading and Real-Time Performance: Use ROS2's real-time capabilities and multi-threading support to ensure the system runs efficiently.

Documentation and Maintenance

- Code Documentation: Maintain clear and organized documentation for all ROS packages, nodes, and parameters.
- Version Control: Use Git or similar tools to manage software changes and facilitate collaboration.

By following this structured approach, we will create a robust and scalable high-level software framework on the Xavier NX using ROS2, enabling our autonomous vehicle to perform effectively in both simulation and real-world environments.

8.4 - Autopilot Configuration

Configuring the autopilot system is a critical step in ensuring precise control and navigation for the autonomous vehicle. The autopilot serves as the bridge between high-level software commands and the vehicle's hardware, managing tasks such as motor control, sensor fusion, and communication. To begin, we will select and install MAVLink-compatible firmware PX4. PX4 offers a lightweight and configurable platform suitable for high-speed applications. The firmware will be installed and configured using tools like QGroundControl or Mission Planner.

Once the firmware is installed, we will calibrate the onboard sensors to ensure accurate data collection and system performance. This includes calibrating the Inertial Measurement Unit (IMU) for motion tracking, and the compass for accurate heading. External sensors, such as LiDAR or cameras, will also be integrated and configured for proper communication with the autopilot. MAVLink will serve as the communication protocol between the autopilot and the high-level software on the Xavier NX, enabling real-time telemetry for data exchange like position, and error messages. Additionally, we will configure remote control (RC) inputs for manual override and define failsafe behaviors to handle critical scenarios like signal loss or low battery.

To enable autonomous operations, we will configure various flight modes, such as Position Hold or Mission Mode, and fine-tune the autopilot’s PID controllers for smooth motor control. The output channels for motors will be mapped and verified to ensure correct direction and speed control, and additional actuators like servos for steering will be calibrated. Sensor fusion algorithms within the firmware will combine data from the IMU, GPS, and external localization sources like SLAM to enhance navigation accuracy. For

autonomous navigation, we will program waypoint commands and test their precision in simulation and real-world conditions.

Following successful simulation tests from previous steps, hardware validation will ensure the autopilot can control motors, sensors, and navigation systems effectively. Data logging will be enabled to collect telemetry for further performance analysis and optimization. Throughout the process, we will maintain detailed documentation of the autopilot configuration, including wiring diagrams, parameter settings, and calibration steps, to support future troubleshooting and refinement. This comprehensive configuration process will ensure seamless integration of the autopilot with the vehicle's hardware and high-level software, providing robust and reliable control and navigation capabilities.

8.5 - System Status Indicator Board

The system status indicator board is an essential component for monitoring and communicating the operational state of the autonomous vehicle. It provides visual feedback to help the team quickly assess the system's status during development, testing, and operation. Here’s what we will do to design, assemble, and integrate this board.

We will start by defining the key system states that the board will display, such as:

- Power Status: Indicators for whether the system is powered on and if specific subsystems (e.g., motors, sensors) are active.
- Communication Status: Lights or signals to indicate successful communication between the control systems and subsystems.
- Error States: Visual or audible alerts for system faults, such as low battery, sensor malfunctions, or motor errors.
- Operational States: Indicators for modes like idle, active, or emergency stop.

After the custom PCB for the indicator board is designed, we will proceed with its assembly. As we are doing so, we will keep in mind:

- Component Placement: Position LEDs and displays on the PCB in an organized manner that allows easy identification of each status.
- Circuit Design: Ensure that each indicator is connected to the appropriate control signal, using resistors or transistors as needed to protect components.
- Power Supply: Integrate a dedicated power supply line or connect to the main power system with appropriate voltage regulation.

For wiring and integration, we will:

- Wire the board to the microcontroller or autopilot system that will provide the status signals.

- For more complex indicators like displays, establish data communication protocols between the microcontroller and the indicator board.
- Securely attach the status indicator board to the vehicle chassis in a visible and accessible location.

We will also write the software needed to drive the system status indicators. We will program different patterns or colors for LEDs to represent various states (e.g., blinking for warnings, solid for active). We will test the communication between the main system and the indicator board to ensure responsiveness and accuracy. Once assembled and programmed, we will test the board under various conditions to verify that each indicator activates correctly based on the system's state.

Finally, we will document the functionality and wiring of the system status indicator board for future reference. Any observed issues during testing will be resolved, and the board will be fine-tuned to ensure it provides clear and reliable feedback during the vehicle's operation.

By completing these steps, the system status indicator board will become a valuable tool for debugging and monitoring, enhancing the safety and efficiency of the development and testing processes.

9 - System Testing and Evaluation

After completing the initial research and acquiring the major components for our autonomous vehicle project, our next step was to thoroughly test and evaluate each item to ensure they met our performance expectations. This phase was critical to verify the functionality and reliability of the materials before integrating them into the system. Our goal was to identify and address any potential issues early, ensuring a smooth transition into the development and integration phases.

Following the individual testing of the components, we will need to do some integration testing along with the software set developed for the vehicles.

9.1 - Component Testing

Below are the testing procedures of some of the major hardware components

Jetson Xavier NX

The Jetson Xavier NX is a critical component of our autonomous vehicle project, serving as the primary computing unit for processing sensor data, running AI models, and controlling the vehicle. To ensure its functionality, we conducted a series of tests upon receiving it.

- Initial Setup and Boot Testing:
    - Connected the Jetson Xavier NX to a compatible power supply.

    - Attached the necessary peripherals, including a monitor, keyboard, and mouse, to monitor its output.
    - Booted the device to confirm that it powered up correctly and displayed the operating system interface.
    - Checked for errors during the boot process and ensured the system reached the desktop environment.
- Connectivity Testing
    - Wi-Fi and Ethernet: Tested network connectivity by connecting to a wired Ethernet connection. We found that the unit we purchased does not come with a Wi-Fi module and that we would need to buy the Wi-Fi module separately for the Jetson Xavier NX.
    - Verified internet access via Ethernet connection and did an internet speed test to confirm stable data transmission.
- USB and GPIO Ports:
    - Connected external devices, such as a USB webcam to test the functionality of USB ports and GPIO pins.
    - Used a multimeter to check the GPIO pins' voltage output and responsiveness.
- Thermal Management
    - Monitored the temperature of the Jetson Xavier NX under varying workloads using onboard diagnostics tools.
    - Tested the fan and heat sink performance by running computationally intensive tasks and ensuring the system maintained safe operating temperatures.

Through these tests, we confirmed that the Jetson Xavier NX is fully operational and capable of meeting the computational demands of our autonomous vehicle project. It is now ready for integration once the vehicle's platform is complete.

Storage Device for Jetson Xavier NX

To ensure data storage components work reliably and meet the demands of our autonomous vehicle project, we conducted thorough tests on both the 32 GB Micro SD card and the 250 GB NVMe SSD card.

- Micro SD Card (32 GB): The Micro SD card is crucial for loading the operating system on devices like the Jetson Xavier NX and storing essential files during development for easy transfer.
    - Physical Inspection: Checked the Micro SD card for any physical damage

or defects, ensuring connectors were clean and intact.

    - Formatting and Compatibility: Inserted the Micro SD card into a card reader and connected it to a PC.
    - Reformatted the card to the appropriate file system (ext4 for Linux compatibility) using a disk management tool.
    - Ensured the formatting process completed without errors.
    - Write and Read Speed Tests: Ran benchmarking tools using AJA System Light software to measure the card's read and write speeds, ensuring it met the advertised performance metrics.

- NVMe SSD Card (250 GB): The NVMe SSD provides high-speed, large-capacity storage for data-intensive tasks such as recording sensor data, running machine learning models, and storing logs.
    - Physical Inspection: Examined the SSD for visible defects or damage to the connectors.
    - Installed the NVMe SSD into the Jetson Xavier NX's M.2 slot.
    - Powered on the Jetson and verified that the SSD was recognized in the system's storage devices list.
    - Formatting: Initialized and formatted the SSD with the ext4 file system for compatibility with the Jetson Xavier NX.

Traxxas Slash 4X4 "Ultimate" RTR 4WD Short Course Truck

This short-course truck will serve as the base platform for the autonomous vehicle. Testing will be conducted to ensure its mechanical and electrical components are fully functional.

- Mechanical Testing:
    - Suspension setup testing will be conducted to ensure that the center of gravity and roll center are positioned at their specified positions, and that the vehicles' roll-reduction mechanism has achieved an effectiveness of at least 1.25. Center of gravity testing will involve suspending the vehicle from wires where the front of the vehicle faces upwards. The steady state angular rotation of the vehicle will be observed and used to determine where the string should be attached in order to have no angular rotation of the vehicle's body; This location of attachment will indicate the location of the center of gravity of the vehicle. Inertial Measurement Units can be attached to the vehicle during cornering to measure the lateral acceleration of the vehicle during a corner. Using this data and vehicle specifications as in (7), the vehicle's roll angle can be determined. Comparing the roll angle after the mechanism has been implemented and before it was implemented will help verify effectiveness.

- Drivetrain testing will consist of timed runs of the vehicle with an upgraded motor. The runs will measure the vehicle's time to 14.5 mph over numerous trials in order to obtain an average time that the vehicle takes to go from rest to 14.5 mph. Tests should verify the that the upgraded motor have in-fact improved the vehicles 0 to 14.5 mph time by at least 5 percent.

- In order to verify that the vehicle has the desired 50/50 weight distribution, a test similar to the suspension center of gravity test will be conducted. This test will involve suspensing the vehicle on wires where the top of the vehicle faces upwards relative to the ground. By varying the location of the wire's attachment point, the vehicle should exhibit varying levels of angular rotation. If the vehicle exhibits no rotation when the wires are attached at the axis that lies between the front and rear axle axes, then the vehicle has the desired 50/50 weight distribution between the front and rear axles.

- The ideal power to weight ratio of the vehicle can be tested by measuring the weight of the vehicles on a scale and using the theoretical motor power rating. The vehicles' power to weight ratio - by definition - will be the ratio/quotient of the vehicles' theoretical power rating and the measured weight of the vehicle.

- Electrical Testing:

  - Tested the Electronic Speed Controller (ESC) and motor combination by gradually ramping up throttle via the remote control.

  - Assessed the steering servo's responsiveness and range of motion.

  - Verified the binding of the radio transmitter and receiver to ensure control signals were properly received.

- Performance Testing:

  - Conducted on-road and off-road trials to evaluate the truck's handling, traction, and speed capabilities.

  - Simulated payload conditions to confirm the truck could handle added weight without adverse effects on performance.

Lipo Batteries and Charger Combo

The power source is critical for the vehicle's operation, especially considering the high current demands of the Traxxas Slash and additional electronics we will add.

- Lipo Batteries Testing:

  - Measured the voltage of in the batteries with a multimeter to ensure they were not completely deplated before the first charge.

  - Performed an initial charge using the included charger to verify the battery

reaches an around 11.1V.

    - Conducted a discharge test by running the Traxxas Slash under various loads to evaluate the battery's capacity, output consistency, and thermal performance.
    - Monitored for overheating, swelling, or voltage drops, which would indicate potential defects.

- Charger Testing:
    - Verified the charger settings and operation by setting it to the correct battery type (Lipo) and desired charge current.
    - Charge the battery to the operational voltage (7.4 to 11.1) for the vehicles and make sure that the vehicles work properly [26].
    - Tested the safety feature automatic cutoff at full charge. Made sure that the charger does not try to charge it over 12.6V [63].

TRX to XT90 Adapter

The TRX to XT90 adapter allows us to connect TRX connectors from the battery to XT90 connectors commonly used with other components.

Testing Procedure:

- Inspected the adapter for physical integrity, ensuring proper soldering and no visible damage.
- Connected a TRX-equipped battery to a multimeter via the XT90 adapter and monitored for a secure connection without voltage drop or overheating.

TRX ID Connector Converter

TRX ID Connector Converter facilitates compatibility between TRX ID batteries and standard TRX setups.

Testing Procedure:

- Verified the connector conversion worked by linking a TRX ID battery to a TRX input port.
- Measured voltage with a multimeter before and after the conversion to confirm accurate transmission.
- Ensured the converter securely latched and withstood sudden movement during operation.

Barrel Jack to Pigtail

The Barrel Jack to Pigtail provides a flexible power connection for devices that require

barrel jack input like the Jetson Xavier NX.

Testing Procedure:

- Confirmed polarity of the pigtail wires using a multimeter to ensure they matched the barrel jack's specifications.
- Connected the pigtail to a power source and measured the voltage with a multimeter to ensure that the cable works fine.
- Assessed stability under operation and ensured no loose connections.

Bullet Adapter 4mm Male to 3.5mm Female

The Bullet Adapter 4mm Male to 3.5mm Female enables compatibility between bullet connectors of different sizes especially the one from the VESC ESC to the motor.

Testing Procedure:

- Physically inspected the adapters for quality and ensured a snug fit with 4mm male and 3.5mm female bullet connectors.
- Conducted a continuity test using a multimeter to confirm electrical connectivity.

VESC 3S PPM Cable

The VESC 3S PPM Cable transmits PPM signals for controlling the VESC.

Testing Procedure:

- Inspected the cable for manufacturing defects, ensuring the connectors were firmly attached and free of damage.
- Connected the PPM cable between the receiver and the VESC, confirming a secure fit at both ends.
- Conducted a continuity test using a multimeter to confirm electrical connectivity.

These testing procedures ensured that all components were in working order and ready for integration into the autonomous vehicle system. This foundational verification is critical as we move into the development and integration phase of the project.

9.2 - Overall Integration

With all the major components for our autonomous vehicle project now in hand, we've started planning the integration process to bring the system together. However, since we are still in the early stages of assembling the vehicle, several critical components remain under development. The mounting platform, which will house and organize the electronic and mechanical systems, has not yet been fabricated. Similarly, the powerboard, responsible for managing the vehicle's power distribution, is still in the production phase. The indicator board, which will provide feedback and status updates for the vehicle, also

remains unfinished.

As a result, we have not yet been able to test the vehicle with all the components assembled and fully operational. Instead, our focus has been on ensuring individual components function properly and gathering design insights for efficient integration. Once these essential pieces are completed and installed, we'll proceed with comprehensive testing of the fully assembled system to ensure seamless operation and identify any necessary adjustments. This staged approach allows us to address challenges incrementally while maintaining steady progress toward our final goal.

10 - Administration

Effective project administration is key to the successful completion of any complex undertaking. It involves the coordination of resources, time, and efforts to ensure that objectives are achieved within scope, budget, and schedule. This requires structured planning, clear communication, and diligent monitoring of progress. In this section, the administrative strategies for the project are outlined, including milestone tracking, budget management, and team collaboration to ensure a streamlined and efficient development process.

10.1 - Project Milestones

To ensure that project milestones are achieved on schedule, it is essential to maintain consistent and effective communication among all team members. Weekly meetings are necessary to assess progress, address potential roadblocks, and verify that our objectives are being met. Additionally, open and ongoing communication throughout the project timeline will be crucial for adapting to any unforeseen challenges and ensuring that tasks are completed efficiently.

The milestones outlined below are tentative and will be revised as needed to reflect the evolving needs and priorities of the project. Flexibility will be maintained to accommodate changes while staying focused on achieving the overall goals.

10.1.1 - Senior Design 1

The table below highlights the primary tasks and accomplishments completed during Senior Design 1. These tasks laid the groundwork for the project's development, ensuring a strong foundation for subsequent phases.

Table 10.1.1: Senior Design 1 Milestones

Start Date	End Date	Task	Description
8/19/24	8/22/24	Form group	Form the team and get acquainted with group members
8/19/24	9/19/24	Research/ familiarization	Familiarize with software being used and study autonomous vehicle techniques

8/22/24	9/5/24	Requirement analysis	Define system requirements (hardware and software)
9/9/24	9/26/24	D&C revision	Prepare to turn in divide and conquer document revision (due 9/27 @ noon)
10/25/24	10/25/24	60-page draft report	Submit 60-page draft report (due 10/25 @ noon)
9/12/24	10/31/24	Hardware setup	Obtain hardware and begin testing components; begin building vehicle
9/12/24	10/31/24	Algorithm development	Setup simulation environment and begin implementing algorithms in said simulation; optimize algorithms
11/8/24	11/8/24	60-page report revision	Submit 60-page report revision (due 11/8 @ noon)
11/7/24	11/26/24	Rapping up research phase.	Refine and complete the necessary research in order to move on to next steps.
11/26/24	11/26/24	Final report/ mini demo video	Submit final report and mini demo video (due 11/26 @ noon)

10.1.2 - Senior Design 2

The tasks and objectives outlined for Senior Design 2 build on the progress made in the initial phase. During this phase, we will focus on completing key deliverables, refining designs, finalizing prototypes, and preparing for presentation. The detailed breakdown of tasks for Senior Design 2 is provided below, with each step aligned to achieve the project's overarching goals.

Table 10.1.2: Senior Design 2 Milestones

Start Date	**End Date**	**Task**	**Description**
1/6/25	2/27/25	Real-world testing (continued)	Continuation of real-world testing that began the previous semester (as needed)
2/20/25	4/3/25	Documentation/ presentation	Compile detailed documentation of system design, algorithms, and performance specs; create a final report and prepare a live demonstration (if possible)
4/1/25	4/3/25	Live presentation	Present and demonstrate all findings/deliverables

By maintaining a structured yet flexible approach, our team will work collaboratively to address challenges and ensure the successful completion of the project within Senior Design 1 & 2.

10.2 - Budget and Financing

Our project is integrally connected to the Connected and Autonomous Vehicle Research Lab (CAVREL) at the University of Central Florida. This affiliation significantly influences our budgeting and financing strategies. Since the hardware and components developed for this project will serve as foundational assets for future research endeavors within the lab, we prioritize durability and high performance in our component selection. This ensures that our investments are not only beneficial for the current project but also provide long-term value to ongoing and future projects at CAVREL.

10.2.1 - Importance of Durability and Performance

- Longevity for Future Projects: By investing in durable components, we reduce the need for frequent replacements, thereby saving costs over time and ensuring that future projects can leverage these components without additional expenditure.
- Enhanced Research Capabilities: High-performance parts enable us to push the boundaries of our research, allowing for more advanced testing and development in connected and autonomous vehicle technologies.
- Cost-Effectiveness: Although high-quality components may have a higher upfront cost, they offer better value over their lifespan due to reduced maintenance and replacement needs.
- Reliability and Safety: Durable and high-performing components contribute to the overall reliability and safety of the vehicle, which is paramount in research settings where consistent results are crucial.
- As a multidisciplinary project, the costs span several categories, including hardware, software, testing, and logistics. Below is a breakdown of expected expenses.

10.2.2 - Estimated Costs of the Project

Below is a detailed breakdown of the estimated costs associated with the project, emphasizing components that offer durability and high performance.

Table 10.2.1: Initial Project Estimate

Component	**Unit Cost**	**Quantity**	**Total Cost**
Mechanical Components			
Chassis and Frame	$500	2	$1,000

Improved Motor	$100	1	$100
Additional Mechanical Improvement Components	~ *Varies*	~ *Varies*	$250
Miscellaneous Hardware (Fasteners, brackets, etc)	~ *Varies*	~ *Varies*	$100
Electrical Components			
Batteries	$100	4	$400
Battery Charger	$50	2	$100
Electronic Speed Controller	$100	2	$200
Display Board	$50	1	$50
Electrical Wiring and Connectors	~ *Varies*	~ *Varies*	$50
Computing Unit, Sensors, and Electronics			
Computing Unit	$1000	2	$2,000
Data Storage Device	$50	2	$100
High-Speed LiDAR	$2000	2	$4,000
Depth Camera	$300	2	$600
9 Axis IMU	$150	2	$300
Microcontrollers	~ *Varies*	~ *Varies*	$50
Communication Modules	~ *Varies*	~ *Varies*	$50
Integrated Circuits Components	~ *Varies*	~ *Varies*	$50
Manufacturing and fabrication			
Peripherals Mounting Platform	~ *Varies*	~ *Varies*	$50
Printed Circuit Board (PCB)	~ *Varies*	~ *Varies*	$50
Uncategorized Miscellaneous			
Racetrack Wall Material	~ *Varies*	~ *Varies*	$200

Software Licensing	~ *Varies*	~ *Varies*	$100
Unexpected Expenses	~ *Varies*	~ *Varies*	$200
Total			$10,000

10.2.3 - Financing

The financing for this project is fully provided by the Connected and Autonomous Vehicle Research Lab (CAVREL) and Dr. Yaser Fallah, our project sponsor. The direct sponsorship by CAVREL and Dr. Fallah's commitment to funding the project provides a stable financial foundation, allowing us to prioritize quality and durability in our component selection. This financial support not only enables the successful execution of the current project but also ensures that the developed systems and hardware will serve as valuable assets for future research within the lab. By focusing on long-term value and integrating our work closely with CAVREL's objectives, we contribute to the ongoing advancement of connected and autonomous vehicle technologies at the University of Central Florida

10.2.4 - Bill of Material for Known Expenses

As we progress toward the completion of our Small Scale Autonomous Vehicle project, we have acquired the majority of the necessary hardware components. Below is a detailed bill of materials (BOM) that reflects both the research conducted during the planning phase and the actual costs of components that have been purchased. This BOM serves as a comprehensive record of the materials used, aiding in budget tracking and future reference for similar projects.

Table 10.2.2: Bill of Material

Component	**Quantity**	**Unit Cost**	**Total Cost**
Chasis and Power Related			
Traxxas Slash 4X4 "Ultimate" RTR 4WD Short Course Truck	2	$500	$1,000
2 Lipos and Charger Combo	2	$240	$480
TRX to XT90 Adapter Pack	2	$10	$20
TRX ID Connector Converter Pack	2	$6	$ 12
LiPo safety bag like the Aketek Silver Large Size Lipo Battery Guard Sleeve/Bag for Charge & Storage.	2	$10	$20

Barrel Jack to Pigtail Pack	2	$5	$10
Bullet Adapter 4mm Male 3.5mm Female Pack	2	$9	$18
VESC 6 MkV	2	$258	$516
VESC 3S ppm cable	2	$5	$10
Sensing and Computing Related			
Jetson Xavier NX	2	$659	$1,318
Hokuyo UST-10LX Scanning Laser Rangefinder	2	$1,670	$3,340
Intel RealSense D345i	2	$234	$468
Short (~1 ft) A USB-to-micro USB cable - Pack	2	$8	$16
Sony DUALSHOCK 4 Wireless Controller for PlayStation 4	2	$58	$116
Wifi Antenna	2	$9	$18
Micro SD Card 32 GB	2	$14	$28
NVMe SSD Card 250 Gb	2	$46	$92
Miscellaneous			
M2 - M5 Socket Head Assortment Kit	1	$25	$25
M2 - M4 Standoff Kit	1	$23	$23
HDMI emulator	2	$14	$28
Header Pins Pack	1	$9	$9
Air Duct Material (for creating a race Track)	8	$32	$256
Total			**$7,805**

This current bill of materials reflects our commitment to acquiring high-quality components that ensure both the success of the current project and their utility for future research within CAVREL. By carefully selecting suppliers, leveraging discounts, and utilizing university resources, we have managed to optimize our expenditures while maintaining the integrity and performance standards required for advanced autonomous vehicle development.

As we move forward, we will continue to monitor our spending closely, updating the BOM as additional components are purchased or if any adjustments are necessary due to design changes or testing outcomes.

10.3 Division of Project Responsibilities

To successfully develop our two small-scale autonomous racing vehicles, we need to design, program, and integrate various subsystems to ensure the vehicles are fully functional. To optimize efficiency and capitalize on each team member's strengths, we will assign tasks based on individual expertise while maintaining close collaboration throughout the project. Below is an outline of the primary responsibilities assigned to each team member.

Tevin - Mechanical Engineering

- Vehicle Mechanics and Dynamics
    - Design, manufacture, and integrate mounting solutions for sensors and electronics.
    - Design and assemble the chassis and vehicle frame using appropriate manufacturing techniques.
    - Ensure through design that the drivetrain and suspension can handle accelerations and velocities commanded by the autonomy software.
    - Optimize the vehicle's weight and weight distribution for good handling across varying track conditions.
- Overall Physical System Testing
    - Assist in physical testing and troubleshooting of the assembled vehicles, focusing on mechanical performance (suspension response, vehicle speed, vehicle handling, etc.).

Casey - Electrical Engineering

- Power Distribution System
    - Design and implement the power management and distribution system, ensuring proper power delivery to all components.
    - Take charge of overall electrical architecture.
    - Calculate and budget the total power requirements for the vehicle (motors, sensors, compute units, etc.).
- Electrical Safety Monitoring
    - Design and program a monitoring module in the power management and distribution system

- Design a battery management system (BMS) that monitors and protects the batteries from overcharging/discharging.
- Ensure safe and efficient power delivery, including voltage regulation and protection circuits.

Asa - Computer Engineering

- System Status Indicator Module: Hardware
  - Design and fabricate a printed circuit board (PCB) that interfaces with the vehicle's sensors and subsystems, routing data to the status module
  - Interface components such as microcontrollers, sensors, and communication modules, to enable real-time data collection and transmission.
  - Ensure that the hardware is power-efficient by selecting low-power components and designing circuits that minimize energy consumption
- System Status Indicator Module: Software
  - Filter and process the incoming data to highlight critical metrics, such as power usage/issues and any system anomalies.
  - Develop a user interface for visualizing status information
  - Program the system to trigger alerts or warnings when critical thresholds are reached

Owen - Computer Science

- Overall High-Level Vehicle Programmation
  - Develop a script that initializes the subsystems required for the autonomous section of the vehicle.
- Race Line Computing
  - Develop algorithms that compute the optimal race line based on track data and vehicle dynamics.
  - Factor in, vehicle capabilities (e.g., maximum speed, steering angle) and track conditions (such as sharp turns, and straightaways) when computing the race line.
  - Continuously update the race line in real-time as the vehicle navigates the track, ensuring responsiveness to dynamic obstacles or changes in track conditions.
- SLAM (Simultaneous Localization and Mapping)

    - Implement SLAM algorithms to enable the vehicle to build a map of its surroundings and localize itself within that map.
    - Fuse sensor data (such as LiDAR, camera, IMU) to continuously update the vehicle's position and orientation in real-time.
    - Ensure that the SLAM system can function in dynamic environments, where obstacles and features may change over time.
    - Optimize SLAM for real-time performance, ensuring low latency and high accuracy.

Israel - Computer Engineering

- Vehicle Steering Control System
    - Develop a precise steering control algorithm to ensure the vehicle can perform accurate and smooth maneuvers during high-speed racing.
    - Utilize control theory and techniques such as Proportional-Integral-Derivative (PID) controllers to enhance stability and response.
    - Simulate the steering behavior in a software environment before integrating it with the physical vehicle to identify and address potential issues.
    - Integrate feedback mechanisms using data from sensors like gyroscopes, accelerometers, and encoders to adjust the steering in real time.
    - Implement predictive modeling to anticipate changes in terrain or obstacles for preemptive steering adjustments.
- Follow the Gap Algorithm
    - Research and adapt the "Follow the Gap" algorithm to optimize for speed, safety, and efficiency in dynamic racing environments.
    - Integrate LiDAR data processing with the algorithm to identify the largest open gap in the surroundings while continuously updating path options.
    - Process sensor data such as LiDAR point clouds and camera feeds to map obstacles and open gaps in real-time.
    - Develop a path-planning module to calculate the safest and most efficient trajectory through the identified gap.
    - Combine the algorithm with additional safety layers to ensure the vehicle avoids close proximity collisions and maintains stability.
- Vehicle Subsystems Assembly and Integration
    - Calibrate the physical steering mechanism to match the control software

parameters, ensuring precise alignment and responsiveness.

    - Mount sensors such as LiDAR, cameras, and inertial measurement units (IMUs) on the vehicle chassis.
    - Adjust and align the sensors to maximize field-of-view and accuracy, conducting preliminary tests to verify proper functioning.
    - Connect and secure all subsystems to ensure smooth communication between components and to avoid damage during high-speed operation.

- Project Website Development
    - Design and program a user-friendly website for our project to introduce the project and present the description and goals for the project.
    - Publish detailed instructions, software code, and downloadable files for anyone interested in the project

- Project Management and Coordination
    - Coordinate tasks across team members, ensuring that deadlines are met and dependencies between tasks are managed.
    - Maintain clear communication between team members to ensure all components are integrated smoothly.
    - Track progress on the design, development, and testing of each vehicle subsystem.

10.3.1 - Division of Tasks Summary

Below is a table that summarizes the high-level tasks mentioned above along with the primary and secondary person responsible for those tasks.

Table 10.3.1: Division of Tasks Summary

Task	Description	Primary	Secondary
Vehicle Mechanical System Design and Assembly	Design and assemble the vehicle's mechanical systems—including chassis, drivetrain, and component mounts.	Tevin	Israel
Overall Physical System Testing	Conduct testing of mechanical systems and analyze results with the intention of improving the vehicle's mechanical performance.	Tevin	Israel
Power Distribution	Design and manage the vehicle's	Casey	Asa

System	power system and electrical architecture to ensure proper power delivery to all components.		
Electrical Safety Monitoring	Develop safety features like battery management and monitoring modules to ensure safe and efficient power delivery.	Casey	Asa
System Status Indicator Module: Hardware	Design and build hardware for real-time system status monitoring, including PCBs and interfaces for data collection.	Asa	Casey
System Status Indicator Module: Software	Develop software to process and visualize critical system metrics and trigger alerts for anomalies.	Asa	Casey
Overall High-Level Vehicle Programmation	Create initialization scripts for the autonomous vehicle subsystems.	Owen	Israel
Race Line Computing	Develop real-time algorithms to compute and update the optimal race line based on vehicle dynamics and track conditions.	Owen	Israel
SLAM (Simultaneous Localization and Mapping)	Implement real-time SLAM algorithms for mapping and localization using sensor data in dynamic environments.	Owen	Israel
Vehicle Steering Control System	Develop and test precise steering control algorithms using control theory and sensor feedback for high-speed maneuvering.	Israel	Owen
Follow the Gap Algorithm	Adapt and implement the "Follow the Gap" algorithm for real-time path planning, using sensor data to navigate safely and efficiently.	Israel	Owen
Vehicle Subsystems Assembly and Integration	Assemble and calibrate vehicle subsystems, including sensors and steering mechanisms, ensuring proper integration and functionality.	Israel	Tevin

Project Website Development	Develop a user-friendly website to introduce the project and share resources with the public.	Israel	Owen
Project Management and Coordination	Manage team coordination, task tracking, and communication to ensure timely integration of all project components.	Israel	Owen

10.4 - LLM Declaration

We hereby declare that, to the best of our knowledge and belief, we have not directly copied or extracted more than seven pages of content from any Large Language Model (LLM) during the course of our work. Our engagement with the LLM has been limited to permissible and responsible uses, including but not limited to drafting initial ideas, creating structured outlines, conducting comparative analyses, summarizing complex topics, and refining text through proofreading. These activities have been carried out with the intent to support our own original work, ensuring adherence to ethical practices and intellectual integrity. Furthermore, we confirm that all outputs generated by the LLM have been critically reviewed, edited, and customized to align with the unique objectives and standards of this document.

11 - Conclusion

As we approach the midpoint of our autonomous racing car project, we reflect on the progress we've made and the challenges that lie ahead. This journey has already been immensely rewarding, allowing us to combine our technical skills with our shared passion for robotics and automation. By completing the research phase and acquiring much of the required hardware, we've established a strong foundation for the subsequent phases. As a multidisciplinary team, we came together with diverse expertise and a shared passion for robotics and intelligent systems to work on this project. We believe that this project is an incredible fusion of innovation, learning, and teamwork. Through the challenges we faced, the milestones we achieved, and the challenges and milestones that lie ahead, we have grown and will continue to grow as engineers, collaborators, and problem-solvers.

From the outset, we set out to tackle a cutting-edge challenge—autonomous racing. This ambition pushed us to learn complex technologies, from computer vision to control systems, and to innovate constantly in a fast-paced and highly technical environment. Developing two autonomous vehicles capable of navigating unknown racetracks is no small feat, but the rigor and excitement of this endeavor made every step worthwhile.

11.1 - Reflection on Progress

Reaching this point has been a significant milestone and a testament to the effort, collaboration, and determination of our team. The research phase was incredibly insightful. Diving deep into literature, comparing technologies, and evaluating different algorithms pushed us to think critically about our design choices. We spent countless hours weighing

the pros and cons of various approaches to computer vision, path planning, and control systems. We methodically tackled this phase, ensuring that the foundation we're building on is solid and well-informed. Securing the major components of our system was another important step forward. From the NVIDIA Jetson Xavier NX to the Hokuyo LiDAR and Intel RealSense camera, we've brought together state-of-the-art technologies that we're confident will bring our vision to life.

One of the standout achievements so far has been the mechanical research done on the second car. Recognizing the need for greater mechanical performance to surpass the base performance of the vehicle, we aimed to improve key components and attributes of the vehicle. Weight reduction opportunities and improve weight distribution strategies have been identified as important aspects who's improvement would increase the vehicle's performance without compromising its durability. Modifications to the suspension system and drivetrain aim to optimize stability and handling in corners, and acceleration out of corners. These refinements ensure that the second car is mechanically superior to the first and base car, which will allow better algorithm testing and development.

Another critical milestone has been the research for the design and development of the power distribution board. To efficiently supply power to all components from a single battery source, we are custom-designing a board tailored to our system's unique requirements. The board will provide reliable, regulated power to high-performance components like the NVIDIA Jetson Xavier NX, Hokuyo LiDAR, and various other components, ensuring stable operation even under the demanding conditions of real-time autonomous racing. This work will allow us to maintain a compact and efficient energy system while avoiding the pitfalls of power inconsistencies.

Additionally, the progress for the creation of the system status indicator board represents another important step in enhancing the reliability and usability of our project. Built using an ESP32 microcontroller, this module provides real-time feedback on the car's operational status, such as battery levels, sensor functionality, and computing health. This not only aids in debugging but also ensures that the vehicles can operate with minimal interruptions during real-world testing. The status indicator system is a crucial addition that bolsters the overall robustness of our autonomous race cars.

Moving forward, the integration of these mechanical and electronic systems with advanced software components will be the focus of our efforts. While challenges remain in optimizing algorithms and validating performance on physical tracks, we are confident that our team's technical expertise and problem-solving skills will guide us to success.

11.2 - What We Have Learned

Reaching this midpoint in our project has been an eye-opening experience, and it's remarkable to see how much we've learned along the way. From the research phase to acquiring critical hardware, every step has been a learning opportunity that has deepened our understanding of autonomous systems and strengthened our skills as future engineers.

One of the most important lessons we've learned is the value of comprehensive research. In the early stages, we immersed ourselves in studying cutting-edge technologies,

analyzing various autonomous algorithms, and evaluating hardware options. This process taught us how to critically assess different approaches and make decisions based on performance trade-offs, compatibility, and feasibility. For example, choosing technologies like SLAM for mapping and NVIDIA Jetson Xavier NX for computing involved detailed comparisons and reinforced the importance of aligning our design choices with our goals.

Through our research, we've delved deep into the cutting-edge technologies that power autonomous systems, gaining a solid understanding of their applications and limitations. The careful selection of key components, such as the NVIDIA Jetson Xavier NX and Hokuyo UST-10LX LiDAR, has been instrumental in ensuring the robustness and performance of our design. These choices were driven by a desire to push the boundaries of what small-scale autonomous vehicles can achieve while maintaining practicality and reliability.

We've also gained a better appreciation of how complex systems come together. Working with diverse hardware components—from the LiDAR and IMU to the custom powerboard—has shown us how crucial it is to consider not just individual performance but how these components integrate into a cohesive system. Ensuring compatibility and planning for real-time communication between sensors, processors, and controllers required us to think holistically, balancing technical requirements across disciplines.

Collaboration has been another cornerstone of our learning experience. Our team comes from different engineering backgrounds, and it's been incredible to see how our combined expertise creates solutions we couldn't have achieved individually. This interdisciplinary teamwork has taught us how to communicate effectively across fields and leverage each other's strengths.

Overall, what we've learned so far has been both technical and personal. We've expanded our engineering skill sets, deepened our understanding of autonomous systems, and grown as a team. These lessons will not only carry us through the remainder of this project but also prepare us for the challenges we'll face in our careers.

11.3 - Future Work

While completing the research phase and acquiring the critical components have been significant achievements, the most thrilling parts of the project are still to come. One of the aspects we are most excited about is diving into algorithm and circuit design development and optimization. This is where all our research and planning will start to take shape in a tangible way. Implementing SLAM for mapping and localization, Rapidly Exploring Random Trees (RRT) for path planning, Model Predictive Control (MPC) for decision-making, the System Status Indicator, and the power management module will be a real test of our skills. We are particularly interested to see how the theoretical models we've explored during the research phase will perform in real-world scenarios, both in simulations and eventually on the physical vehicles.

Looking ahead, we are excited to transition into the development and testing phases, where the project will truly come to life. While we anticipate challenges—ranging from algorithmic optimization to real-world validation—we are confident that the

interdisciplinary skills and collaborative spirit of our team will enable us to overcome them.

We are also looking forward to the hands-on work involved in assembling and testing our systems. From integrating the hardware components to fine-tuning the software for real-time performance, this phase will allow us to engage directly with the vehicles we've envisioned.

Of course, the real-world testing phase is where the true excitement lies. Setting up physical racetracks and watching our vehicles tackle obstacles and make decisions in real-time will be a culmination of all our efforts. It will be fascinating to see how our cars handle unknown environments, and we are eager to analyze their performance to refine and optimize the algorithms. This phase will also challenge us to adapt quickly, addressing any unforeseen issues that arise when moving from simulation to reality.

As we prepare for these next phases, we feel confident in our team's ability to tackle the challenges ahead. We've built a strong foundation, and we believe our collaboration, combined with the passion we share for robotics and intelligent systems, will drive us to deliver a successful project. There's a lot of work to be done, but we are eager to dive in and bring our vision closer to reality.

11.4 - Impact and Legacy

This project is about more than just achieving technical milestones; it's about contributing to the broader community of robotics enthusiasts and engineers. By documenting our process and organizing workshops, we aim to inspire others to explore autonomous systems and advance the field further. We remain committed to leaving a lasting legacy at UCF, empowering future students to build on the foundation we've laid.

Though the final demonstration of our autonomous race cars is still ahead, we're proud of the progress we've made and are excited to continue pushing the limits of innovation and teamwork. This project represents not just a capstone requirement but a significant step toward our professional aspirations and a meaningful contribution to the world of intelligent systems.

Appendices

A - References

[1] *Overview*. (n.d.). Indy Autonomous Challenge. Retrieved September 12, 2024, from https://www.indyautonomouschallenge.com/challenge-about

[2] (n.d.). Gymnasium Documentation. Retrieved September 12, 2024, from https://gymnasium.farama.org/

[3] Salloum, A. (2024, July 1). Student Ambassador Blog — Inspirit AI. Inspirit AI. Retrieved September 12, 2024, from https://www.inspiritai.com/blogs/ai-student-blog/

[4] Audi Challenges Students to Program Tiny Autonomous Cars – News – Car and Driver. (2015, March 11). Car and Driver. Retrieved September 12, 2024, from https://www.caranddriver.com/news/a15356141/audis-driving-cup-asks-students-to-program-super-cute-teensy-audis-to-drive-autonomously/

[5] *JetRacer*. (n.d.). NVIDIA Developer. Retrieved September 12, 2024, from https://developer.nvidia.com/embedded/community/jetson-projects/jetracer

[6] (n.d.). F1Tenth. Retrieved September 12, 2024, from https://f1tenth.org/index.html

[7] *Build a car*. (n.d.). Donkey Car. Retrieved September 12, 2024, from https://docs.donkeycar.com/guide/build_hardware/

[8] Drew, S. (n.d.). Perception, Planning, Control, and Coordination for Autonomous Vehicles. MDPI. Retrieved September 6, 2024, from https://www.mdpi.com/2075-1702/5/1/6

[9] Woodford, C., & Neff, T. (2023, May 18). *How LIDAR works: A simple introduction*. Explain that Stuff. Retrieved September 6, 2024, from https://www.explainthatstuff.com/lidar

[10] ABLIC Inc., "What is a Switching Regulator? – ABLIC Inc.," *ABLIC Inc.* https://www.ablic.com/en/semicon/products/power-management-ic/switching-regulator/intro-2/

[11] M. Harris, "Everything You Need to Know about Conformal Coating," *Altium*, Oct. 10, 2024. https://resources.altium.com/p/everything-you-need-know-about-conformal-coating

[12] NXP Semiconductors, *C-bus specification and user manual*. 2021. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[13] V. Andrushchak and O. Dokanov, "Battery State of Charge explained + SOC algorithm setup example," *Lemberg Solutions*, Sep. 04, 2023. https://lembergsolutions.com/blog/battery-state-charge-explained-soc-algorithm-

setup-example

[14] *LTC2945*, Rev. C. Analog Devices, 2015. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ltc2945.pdf

[15] “RP2040 vs. ESP32: How to Choose the Right Microcontrollers | Xecor.” https://www.xecor.com/blog/rp2040-vs-esp32

[16] “Advantages and Disadvantages of Infrared sensor.” https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Infrared-Sensor.html

[17] M. Harris, “Everything You Need to Know about Conformal Coating,” *Altium*, Oct. 10, 2024. https://resources.altium.com/p/everything-you-need-know-about-conformal-coating

[18] C. Ruth, “The evolution of Wi-Fi technology and standards,” *IEEE Standards Association*, Jun. 28, 2024. https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/

[19] Espressif Systems, “ESP32 datasheet,” report, Oct. 2016. [Online]. Available: https://cdn.sparkfun.com/datasheets/IoT/esp32_datasheet_en.pdf

[20] Arduino, *Arduino® UNO R3 Product Reference Manual*. 2024. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno

[21] “NiMH/LiPO DUAL CHARGER INSTRUCTIONS.”

[22] Texas Instruments Incorporated, *MSP430FR698X(1), MSP430FR598X(1) Mixed-Signal Microcontrollers*. 2018. [Online]. Available: https://www.ti.com

[23] Raspberry Pi Foundation, *Raspberry Pi Zero V1.3*.

[24] University of Pennsylvania, “F1Tenth Power Board V2024.1,” Jun. 2024.

[25] Arduino, “Arduino® Nano RP2040 Connect Product Reference Manual,” Oct. 2024.

[26] E-Maxxdude, “Slash® 4x4 ultimate: 1/10 scale 4WD Brushless Short Course Truck with TQITM Radio System, Traxxas LinkTM wireless module, & Traxxas Stability Managment (TSM)®,” E-Maxxdude, https://traxxas.com/products/models/electric/slash-4x4-ultimate-68277-4?t=specs (accessed Oct. 25, 2024).

[27] What is power to weight ratio and how does it affect vehicle performance? - in the garage with Carparts.com, https://www.carparts.com/blog/what-is-power-to-weight-ratio-and-how-does-it-affect-vehicle-performance/ (accessed Oct. 25, 2024).

[28] “Understanding weight transfer for performance car driving,” Total Car Control, https://www.total-car-control.co.uk/performance-driving/weight-transfer-in-driving (accessed Oct. 25, 2024).

[29]“Properties table,” Simplify3D Software, https://www.simplify3d.com/resources/materials-guide/properties-table/ (accessed Oct. 25, 2024).

[30] M. Trzesniowski, *Suspension System*. Wiesbaden, Wiesbaden: Springer Fachmedien Wiesbaden Springer Vieweg, 2023.

[31]“Definitive guide to suspension tuning,” suspensionspot, https://suspensionspot.com/blogs/news/definitive-guide-to-suspension-tuning (accessed Oct. 25, 2024).

[32]“Engineering,”Vaia,https://www.vaia.com/en-us/explanations/engineering/automotive-engineering/center-of-gravity-influence/ (accessed Oct. 25, 2024).

[33]C. Rosales, “How roll center affects your car’s dynamics,” The Drive, https://www.thedrive.com/guides-and-gear/how-roll-center-affects-your-cars-dynamics (accessed Oct. 25, 2024).

[34] D. Routley, “Spring rates and suspension frequencies - plus frequency calculator!,” DRTuned Racing, https://www.drtuned.com/tech-ramblings/2017/10/2/spring-rates-suspension-frequencies (accessed Oct. 25, 2024).

[35] Suspension Secrets, “Wheel rate and chassis roll stiffness – how to adjust and tune suspension secrets,” Suspension Secrets, https://suspensionsecrets.co.uk/wheel-rate-and-chassis-roll-stiffness/ (accessed Oct. 25, 2024).

[36] *How to implement status LED in a system?* (n.d.). Electrical Engineering Stack Exchange. https://electronics.stackexchange.com/questions/334074/how-to-implement-status-led-in-a-system

[37]*Software Setup — Dingo Tutorials 0.0.5 documentation*. (n.d.). https://www.clearpathrobotics.com/assets/guides/melodic/dingo/software_setup.html

[38]Luo, Y. (2019). Capacitive Touch Design flow for MSP430TM MCUs with CapTIvateTM technology. In *Application Report*. https://www.ti.com/lit/an/slaa842b/slaa842b.pdf?ts=1727128636382

[39]*CCSTUDIO IDE, configuration, compiler or debugger | TI.com*. (n.d.). https://www.ti.com/tool/CCSTUDIO#:~:text=It%20comprises%20a%20suite%20of,expected%20to%20be%20CCS%2012.8.

[40]*Code Composer studio*. (n.d.). Integrated Development Environment Supporting Texas Instruments Microcontrollers and Embedded Processors - Third-Party Products & Services - MATLAB & Simulink. https://www.mathworks.com/products/connections/product_detail/code-composer-studio.html

[41]J, M., & J, M. (2024, June 6). *Why is C The Most Preferred Language for Embedded Systems?* https://www.emertxe.com/blog/embedded-systems-

design/2023/08/24/why-is-c-the-most-preferred-language-for-embedded-systems/#:~:text=C%20provides%20optimized%20machine%20instructions,major%20challenge%20in%20embedded%20systems.

[42]Agarwal, T. (2024, January 18). *ESP32 vs Raspberry Pi : Definition & the Main Differences*. ElProCus - Electronic Projects for Engineering Students. https://www.elprocus.com/difference-between-esp32-vs-raspberry-pi/#:~:text=The%20R

[43]*Figure 5: ESP32 functional block diagram.* (n.d.). ResearchGate. https://www.researchgate.net/figure/ESP32-functional-block-diagram_fig5_341446512

[44]Lvgl. (n.d.). *GitHub - lvgl/lvgl: Embedded graphics library to create beautiful UIs for any MCU, MPU and display type.* GitHub. https://github.com/lvgl/lvgl

[45]Arm, C. (n.d.). *New ARM Cortex STM32 Microcontrollers from ST have More of Everything*. Microcontroller.com. https://microcontroller.com/news/arm_cortex_stm1.asp

[46]Mangharam, R., & Zheng, H. (2024). *Lecture 11 - Local Planning: RRT, Spline Based Planner — F1TENTH - Learn latest documentation*. Readthedocs.io. https://f1tenth-coursekit.readthedocs.io/en/latest/lectures/ModuleD/lecture11.html

[47]Zheng, H. (2022, May 18). *F1TENTH Racecar Simulator*. GitHub. https://github.com/f1tenth/f1tenth_simulator

[48]f1tenth. (2020). *GitHub - f1tenth/f1tenth_gym_ros: Containerized ROS communication bridge for F1TENTH gym environment.* GitHub. https://github.com/f1tenth/f1tenth_gym_ros

[49]Team, C. (n.d.). *CARLA*. CARLA Simulator. https://carla.org/

[50]f1tenth-dev. (2020, April 27). *GitHub - f1tenth-dev/simulator: ROS & Gazebo F1/10 Autonomous Racecar Simulator*. GitHub. https://github.com/f1tenth-dev/simulator

[51]"Speed vs torque," Power Electric, https://www.powerelectric.com/motor-blog/speed-vs-torque (accessed Nov. 22, 2024).

[52]Admin, "What is belt drives: Type advantages and disadvantages," SMLease Design, https://www.smlease.com/entries/mechanism/what-is-belt-drives-type-advantages-and-disadvantages (accessed Nov. 22, 2024).

[53]ISO, https://www.iso.org/obp/ui/en/#iso:std:iso:8855:ed-2:v1:en:term:8.1.18 (accessed Nov. 24, 2024).

[54]ISO, https://www.iso.org/obp/ui/en/#iso:std:iso:27548:ed-1:v1:en (accessed Nov. 24, 2024).

[55]ISO, https://www.iso.org/obp/ui/en/#iso:std:iso-astm:52927:ed-1:v1:en (accessed Nov. 24, 2024).

[56]ISO, https://www.iso.org/obp/ui/en/#iso:std:iso:1660:ed-3:v1:en (accessed Nov. 24, 2024).

[57]“Y14.5 dimensioning and Tolerancing,” ASME, https://www.asme.org/codes-standards/find-codes-standards/y14-5-dimensioning-tolerancing (accessed Nov. 24, 2024).

[58]Traxxas, https://traxxas.com/sites/default/files/68086-4-OM-EN-R06.pdf (accessed Nov. 25, 2024).

[59]A. T. Joy, “V-belt - how it works,” Tameson.com, https://tameson.com/pages/v-belt-overview (accessed Nov. 25, 2024).

[60]Mechanix and S. B, “What is Flat Belt Drive?: Its advantages and disadvantages.,” Mechathon, https://mechathon.com/flat-belt-drive/ (accessed Nov. 25, 2024).

[61]Web design by iNet Media Ltd. Digital marketing experts., “Timing belt pros and cons,” Luff Industries Ltd, https://luffindustries.com/blog/timing-belt-pros-and-cons/ (accessed Nov. 25, 2024).

[62]“Replacement 3650 3100KV sensorless Brushless Motor Namibia: Ubuy,” Ubuy Namibia,https://www.ubuy.co.na/product/BNK7WSVLK-kingval-replacement-3650-3100kv-sensorless-brushless-motor-shaft-3-175mm-with-60a-brushless-esc-compatible-with-1-10-rc-car (accessed Nov. 25, 2024).

[63]Traxxas,https://traxxas.com/sites/default/files/KC2238-R00-LiPo-battery-fold-out-large.pdf (accessed Nov. 26, 2024).

[64]T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, “Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios,” arXiv.org, https://arxiv.org/abs/2005.08664 (accessed Nov. 26, 2024).

[65]“SMAC Hybrid-A* planner,” Smac Hybrid-A* Planner - Nav2 1.0.0 documentation, https://docs.nav2.org/configuration/packages/smac/configuring-smac-hybrid.html (accessed Nov. 26, 2024).

[66]Tumftm, “Global_racetrajectory_optimization/opt_mintime_traj at master · TUMFTM/global_racetrajectory_optimization,” GitHub, https://github.com/TUMFTM/global_racetrajectory_optimization/tree/master/opt_mintime_traj (accessed Nov. 26, 2024).

[67]“Asynchronous actions: BehaviorTree.CPP,” BehaviorTreeCPP RSS, https://www.behaviortree.dev/docs/3.8/tutorial-advanced/asynchronous_nodes/ (accessed Nov. 26, 2024).

[68]“Computer use (beta),” Anthropic, https://docs.anthropic.com/en/docs/build-with-claude/computer-use (accessed Nov. 26, 2024).

[69]“Brushed vs. Brushless DC Motors for Electric Cars: How are They Different - Ennovi,” Ennovi, Aug. 21, 2024. https://ennovi.com/brushed-vs-brushless-dc-motors-for-electric-cars-how-are-they-different/

[70]“Build a car. - Donkey Car,” docs.donkeycar.com. https://docs.donkeycar.com/guide/build_hardware/

[71]“HSP 94996 1:8 High Speed 4WD Brushless Off-Road RC Car Savagery Electric RTR Rc Truck,” bometoys, 2024. https://bometoys.com/products/hsp-94996-1-8-high-speed-4wd-brushless-off-road-rc-car-savagery-electric-rtr-rc-truck (accessed Nov. 26, 2024).

[72]“HSP 94118 PRO 1:10 4WD Electric Brushless High Speed Off-Road Rally Racing 2.4G RC Model Car RTR Version,” bometoys, 2024. https://bometoys.com/products/hsp-94118-pro-1-10-4wd-electric-brushless-high-speed-off-road-rally-racing-2-4g-rc-model-car-rtr-version (accessed Nov. 26, 2024).

[73]“1/8スケール ラジオコントロール ブラシレスパワード 4WD レーシングバギー インファーノ MP9e Evo. V2 34111,” Kyoshoamerica.com, 2017. https://kyoshoamerica.com/34111.html (accessed Nov. 26, 2024).

[74]“1/10 Scale Radio Controlled Electric Powered 4WD FAZER Mk2 FZ02L Series Readyset 1970 Chevy® Chevelle® SSTM 454 LS6 Cortez Silver 34416TC,” Kyoshoamerica.com, 2017. https://kyoshoamerica.com/rccar/eponroad/1-10-scale-radio-controlled-electric-powered-4wd-fazer-mk2-fz02l-series-readyset-1970-chevyr-cheveller-sstm-454-ls6-cortez-silver-34416tc.html (accessed Nov. 26, 2024).

[75]F. Caleffi, Lauren, S. Stamboroski, and Brenda Medeiros Pereira, “Small-scale self-driving cars: A systematic literature review,” Journal of Traffic and Transportation Engineering, Apr. 2024, doi: https://doi.org/10.1016/j.jtte.2023.09.005.

[76]“NVIDIA-AI-IOT/jetracer,” GitHub, Apr. 23, 2021. https://github.com/NVIDIA-AI-IOT/jetracer

[77]“Tamiya NSX 1/10 4WD Electric Touring Car Kit (TT-02),” Hobbytown.com, 2021. https://www.hobbytown.com/tamiya-nsx-1-10-4wd-electric-touring-car-kit-tam58634-60a/p1415731 (accessed Nov. 26, 2024).

[78]“Build,” f1tenth.org. https://f1tenth.org/build.html

[79]“Platform,” Mit.edu, 2024. https://racecar.mit.edu/platform (accessed Nov. 26, 2024).

[80]E-Maxxdude, “Slash 4X4 Ultimate | 4X4 RC Truck | Traxxas,” Traxxas.com, Oct. 03, 2018. https://traxxas.com/products/models/electric/slash-4x4-ultimate?t=details (accessed Nov. 26, 2024).

[81]“LaTrax® Rally: 1/18 Scale 4WD Electric Rally Racer | LaTrax,” Latrax.com, 2024. https://latrax.com/products/rally (accessed Nov. 26, 2024).

[82]E-Maxxdude, “Traxxas X-Maxx | Electric RC Monster Truck | Traxxas,” Traxxas.com, Feb. 29, 2024. https://traxxas.com/products/models/electric/x-maxx?t=support (accessed Nov. 26, 2024).

[83]“The Comparison between Microcontrollers and Single board Computer,” Vemeko.com, 2024. https://www.vemeko.com/blog/the-comparison-between-microcontrollers-and-single-board-computer.html

[84]“Flight Controller (Autopilot) Hardware | PX4 Guide (main),” docs.px4.io. https://docs.px4.io/main/en/flight_controller/

[85]“Holybro Pixhawk 6C | PX4 User Guide (main),” docs.px4.io. https://docs.px4.io/main/en/flight_controller/pixhawk6c.html

[86]“ModalAI Flight Core v1 | PX4 Guide (main),” Docs.px4.io, 2024. https://docs.px4.io/main/en/flight_controller/modalai_fc_v1.html (accessed Nov. 26, 2024).

[87]“Sky-Drones AIRLink | PX4 Guide (main),” Docs.px4.io, 2024. https://docs.px4.io/main/en/flight_controller/airlink.html (accessed Nov. 26, 2024).

[88]“ROS 2,” GitHub. https://github.com/ros2

[89]MAVLink, “Introduction · MAVLink Developer Guide,” Mavlink.io, 2009. https://mavlink.io/en/

[90]“Install the ZED Python API - Stereolabs,” @Stereolabs3D, 2024. https://www.stereolabs.com/docs/app-development/python/install (accessed Nov. 26, 2024).

[91]“Open Source Drone Software. Versatile, Trusted, Open. ArduPilot.,” ardupilot.org. https://ardupilot.org/

[92]“PX4 User Guide,” docs.px4.io. https://docs.px4.io/main/en/

[93]“MAVROS,” GitHub, May 22, 2023. https://github.com/mavlink/mavros/blob/master/mavros/README.md

[94]“melodic - ROS Wiki,” wiki.ros.org. https://wiki.ros.org/melodic

[95]“noetic - ROS Wiki,” wiki.ros.org. https://wiki.ros.org/noetic

[96]“Installation — ROS 2 Documentation: Foxy documentation,” docs.ros.org. https://docs.ros.org/en/foxy/Installation.html

[97]“Jazzy Jalisco (jazzy) — ROS 2 Documentation: Rolling documentation,” Ros.org, 2021. https://docs.ros.org/en/rolling/Releases/Release-Jazzy-Jalisco.html (accessed Nov. 26, 2024).

[98]“Cartographer ROS Integration — Cartographer ROS documentation,” google-cartographer-ros.readthedocs.io. https://google-cartographer-ros.readthedocs.io/en/latest/

[99]S. Macenski, “Introduction,” GitHub, Jul. 09, 2022. https://github.com/SteveMacenski/slam_toolbox

[100]tu-darmstadt-ros-pkg, “tu-darmstadt-ros-pkg/hector_slam,” GitHub, Jun. 08, 2018. https://github.com/tu-darmstadt-ros-pkg/hector_slam

[101]Project-MANAS, “GitHub - Project-MANAS/slam_gmapping: Slam Gmapping for ROS2,” GitHub, 2019. https://github.com/Project-MANAS/slam_gmapping

[102]“9.2: P, I, D, PI, PD, and PID control,” Engineering LibreTexts, May 19, 2020. https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09:_Proportional-Integral-Derivative_(PID)_Control/9.02:_P_I_D_PI_PD_and_PID_control

[103]G. Hoffmann, C. Tomlin, M. Montemerlo, and S. Thrun, “Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing †.” Available: https://ai.stanford.edu/~gabeh/papers/hoffmann_stanley_control07.pdf

[104]Ugo Rosolia, A. Carvalho, and F. Borrelli, “Autonomous racing using learning Model Predictive Control,” Advances in Computing and Communications, May 2017, doi: https://doi.org/10.23919/acc.2017.7963748.

[105]“Python Control Systems Library — Python Control Systems Library 0.10.1 documentation,” Readthedocs.io, 2023. https://python-control.readthedocs.io/en/0.10.1/index.html# (accessed Nov. 26, 2024).

[106]“Welcome to the ros2_control documentation! — ROS2_Control: Rolling Oct 2024 documentation,” Ros.org, 2024. https://control.ros.org/rolling/index.html

[107] [Penn], “[Penn] L12 Sampling-based Motion Planning,” Google Docs, 2019. https://docs.google.com/presentation/d/1pBMGNbNAJH1VNka8KikM2C5qJ_Hc-OxCEaJNGT4tcx0/edit#slide=id.g117c562e9a5_0_688 (accessed Nov. 26, 2024).

[108]“navigation - ROS Wiki,” wiki.ros.org. https://wiki.ros.org/navigation

[109]Nav2.org, 2024. https://docs.nav2.org/index.html

[110]“THIS REPO IS NO LONGER MAINTAINED,” GitHub, May 18, 2022. https://github.com/f1tenth/f1tenth_simulator

[111]f1tenth, “GitHub - f1tenth/f1tenth_gym: This is the repository of the F1TENTH Gym environment.,” GitHub, 2020. https://github.com/f1tenth/f1tenth_gym (accessed Nov. 26, 2024).

[112]C. Team, “CARLA,” CARLA Simulator. https://carla.org/

[113]f1tenth-dev, “GitHub - f1tenth-dev/simulator: ROS & Gazebo F1/10 Autonomous Racecar Simulator,” GitHub, Apr. 27, 2020. https://github.com/f1tenth-dev/simulator

[114]“REP 103 -- Standard Units of Measure and Coordinate Conventions (ROS.org),” Ros.org, 2014. https://www.ros.org/reps/rep-0103.html#coordinate-frame-conventions (accessed Nov. 26, 2024).

[115]“REP 105 -- Coordinate Frames for Mobile Platforms (ROS.org),” Ros.org, 2024. https://ros.org/reps/rep-0105.html (accessed Nov. 26, 2024).

[116]“REP 138 -- LaserScan Common Topics, Parameters, and Diagnostic Keys (ROS.org),” Ros.org, 2024. https://ros.org/reps/rep-0138.html (accessed Nov. 26, 2024).

[117]“Upload & analyze files in the Gemini Apps - Computer - Gemini Apps Help,” Google.com, 2019. https://support.google.com/gemini/answer/14903178?hl=en&co=GENIE.Platform%3DDesktop (accessed Nov. 26, 2024).

[118]“What are the file upload size restrictions? | OpenAI Help Center,” Openai.com, 2024. https://help.openai.com/en/articles/8983719-what-are-the-file-upload-size-restrictions (accessed Nov. 26, 2024).

[119]“Upload & analyze files in the Gemini Apps - Computer - Gemini Apps Help,” Google.com, 2019. https://support.google.com/gemini/answer/14903178?hl=en&co=GENIE.Platform%3DDesktop (accessed Nov. 26, 2024).

[120]“PDF support (beta) - Anthropic,” Anthropic.com, 2024. https://docs.anthropic.com/en/docs/build-with-claude/pdf-support (accessed Nov. 26, 2024).

[121]“Grok Image Generator: Revolutionizing Visual AI with Cutting-Edge Technology | Eric Leads,” Ericleads.com, Sep. 10, 2024. https://ericleads.com/grok-image-generator-ai-cutting-edge-technology/ (accessed Nov. 26, 2024).

[122]“Introducing OpenAI o1,” Openai.com, 2015. https://openai.com/o1/

[123]“Introducing GPTs,” Openai.com, 2023. https://openai.com/index/introducing-gpts/

[124]“Introducing canvas,” Openai.com, 2024. https://openai.com/index/introducing-canvas/

[125]Traxxas, https://traxxas.com/sites/default/files/68077-4-OM-EN-R07.pdf (accessed Nov. 26, 2024).

B - Copyright Information

Email request for the use of figure 8.1.1.1

To: info@ubuy.com

To whom it may concern,

My name is Tevin Mukudi and I am writing to request permission to use some of the images on your website: https://www.ubuy.co.na/product/BNK7WSVLK-kingval-replacement-3650-3100kv-sensorless-brushless-motor-shaft-3-175mm-with-60a-brushless-esc-compatible-with-1-10-rc-car. The photos will be used as reference material on a Senior Design Project paper and will be appropriately cited.

Thanks,

Tevin Mukudi
Senior Mechanical Engineering Student
Department of Mechanical and Aerospace Engineering
University of Central Florida

Email request for the use of figure 3.2.1.3

To: support@tameson.com

To whom it may concern,

My name is Tevin Mukudi and I am writing to request permission to use some of the images on your website: https://tameson.com/pages/v-belt-overview. The photos will be used as reference material on a Senior Design Project paper and will be appropriately cited.

Thanks,

Tevin Mukudi
Senior Mechanical Engineering Student
Department of Mechanical and Aerospace Engineering
University of Central Florida

Email request for the use of figure 3.2.3.2

To: tips@thedrive.com

To whom it may concern,

My name is Tevin Mukudi and I am writing to request permission to use some of the images on your website: How Roll Center Affects Your Car's Dynamics. The photos will be used as reference material on a Senior Design Project paper and will be appropriately cited.

Thanks,

Tevin Mukudi
Senior Mechanical Engineering Student
Department of Mechanical and Aerospace Engineering
University of Central Florida

Email request for the use of figures 6.1.3.1 and 6.1.3.2

To: support@traxxas.com

To whom it may concern,

My name is Tevin Mukudi and I am writing to request permission to use some of the images on your website: Slash 4X4 Ultimate | 4X4 RC Truck | Traxxas. The photos will be used as reference material on a Senior Design Project paper and will be appropriately cited.

Thanks,

Tevin Mukudi
Senior Mechanical Engineering Student
Department of Mechanical and Aerospace Engineering
University of Central Florida

C - Large Language Model Prompts and Outcomes

(Large Language Models prompt and replies comparisons)

Prompt 1

“Turn my photo into pixel art”

—

Reply

Gemini (left) and ChatGPT (right)

Prompt 2

“generate a picture of a woman holding a sign that says "I love halloween"”

—

Reply

Grok (left) and ChatGPT (right)

Prompt 3

“generate an image of a castle in europe”

—

Reply

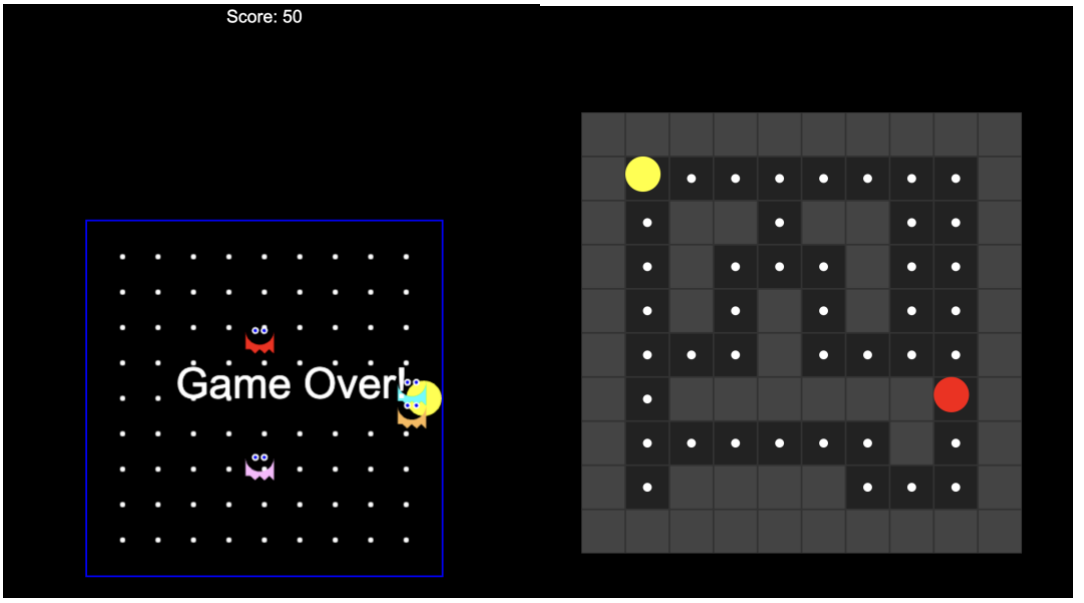
Grok (left), Gemini (center), ChatGPT (right)

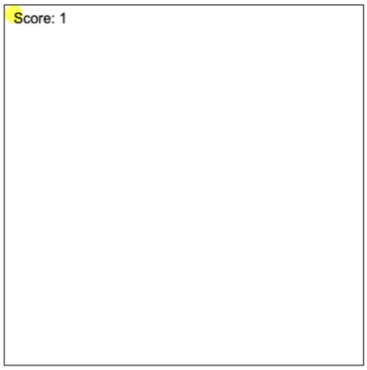
Prompt 4

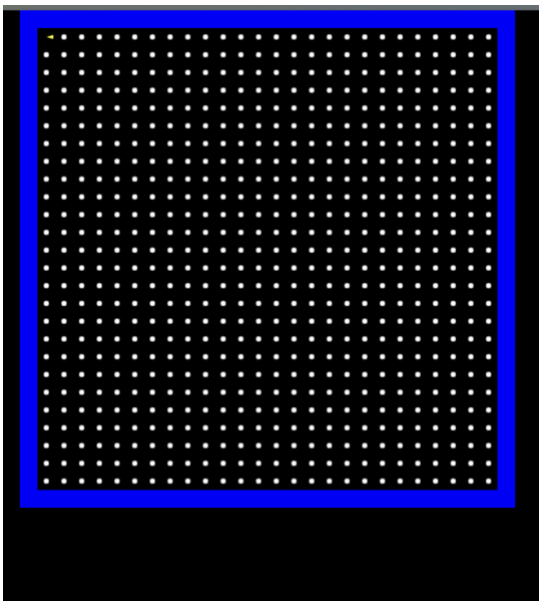
“Write the COMPLETE code for a browser game that plays similar to pac-man and tell me how to run the code you provide in my web browser”

—

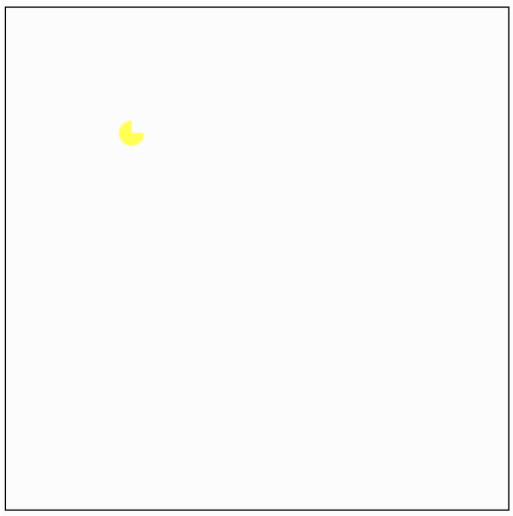
Reply







From top left, clockwise: Claude 3.5 Sonnet, GPT-4o, o1-preview, Grok



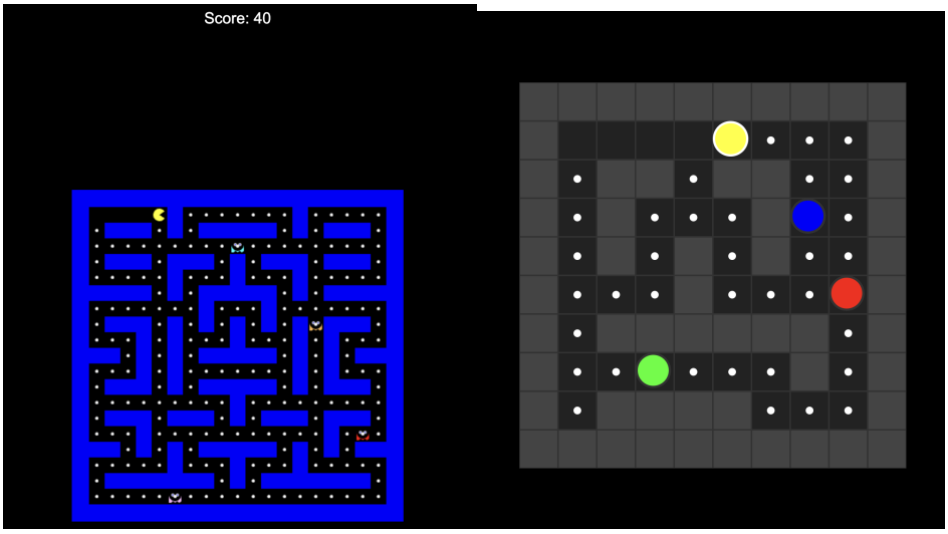
Gemini

Prompt 5

“modify your game to ensure everything moves on a grid, and add walls to make the game more interesting” + “You need to make the game speed slower and ensure that pac man and a ghost don't start on the same space”

—

Reply



Claude 3.5 Sonnet (left) and GPT-4o pacman game after revision

---

Prompt 6

“What are some recent papers related to open set multimodal 3d semantic mapping?”

—

Reply

Sources recovered by Perplexity:

- ConceptFusion: Open-set Multimodal 3D Mapping
- Open-Vocabulary 3D Semantic Segmentation with Foundation Models
- Open-Set 3D Semantic Instance Maps for Vision Language Navigation - O3D-SIM

—

Note: All of these are correct responses, with the latter two published within the last 6 months.