TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
PROGRESS REPORT
ON

SELF DRIVING CAR: MAPPING, PATH PLANNING AND OBSTACLE AVOIDANCE

SUBMITTED BY:
AARJAN BUDATHOKI (PUL077BEI004)
ABHIGYAN BHUSAL (PUL077BEI007)
MANISH GURUWACHARYA (PUL077BEI021)

SUBMITTED TO:
DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

FEB,2024

Acknowledgement

We would like to express our heartfelt gratitude to all those who have encouraged and inspired us in the work on this project.

We would like to thank the **Department of Electronics and Computer Engineering** of IOE, Pulchowk Campus for providing us with an opportunity to work on this project. Similarly, we would like to express our sincere gratitude to the **Robotics Club** for their valuable support. We are especially indebted to our junior Devish Phuyal for his help with fixing the drive-train of our test vehicle and getting our vehicle up and running.

We are thankful to our seniors for their guidance and technical support. Their feedback in the planning phases has helped us gain clarity in what we are trying to do. The guidance they gave us was instrumental in shaping the scope and design of our project.

Finally, we are thankful to our family and friends for their moral support and love.

Abstract

This project focuses on the development of a Self-Driving Car(SDC), emphasizing Mapping, Path Planning, Obstacle Detection and Avoidance. The initial phase involves the precise tracking of the vehicle using sensor fusion algorithms and advanced Artificial Intelligence(AI) techniques. Additionally, a high precision mapping system is created using physical systems such as Lidar and Camera enabling accurate environmental perception.

For the optimal desired motion of the vehicle, the system is implemented with Proportional-Integral-Derivative (PID) controller as well as Model Predictive Control (MPC). Ensuring the security and safety of the vehicle, the project utilizes the Robot Operating System (ROS) framework, along with Real Time Operating System (RTOS) implementation for enhanced system speed.

Afterward, advanced algorithms process real time data for path planning, calculating optimal routes while considering pedestrians on the road. The system autonomously navigates utilizing an Obstacle Detection algorithm to avoid objects in its path. To further enhance safety, the vehicle incorporates emergency override electronic system other fail-safe devices in case of potential system malfunctions.

Keywords: Self-Driving Car(SDC), Artificial Intelligence(AI), Proportional-Integral-Derivative (PID), Model Predictive Control (MPC), Robot Operating System(ROS)

Contents

List of Figures

Abbreviations

SDC	Self Driving Car
IMU	Inertial Measurement Unit
PID	Proportional, Integral and Derivative
MPC	Model Predictive Control
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
LoRa	Long Range
RTOS	Realtime Operating System
SLAM	Simultaneous Localization and Mapping
ICP	Iterative Closest Points
ROS	Robot Operating System
STM	ST Microelectronics
DARPA	Defence Advanced Research Project Agency
CMU	Carnegie Mellon University
EDA	Electronics Design Automation

1 Introduction

1.1 Background

Over the years, the evolution of transportation has seen a significant increase in road traffic, leading to congestion, accidents, and environmental concerns. The need for a safer, more efficient, and sustainable mode of transportation became evident. The introduction of SDCs emerged as a compelling solution to these challenges, leveraging advancements in robotics and AI. SDCs, equipped with sensors and AI algorithms, have the potential to eliminate human errors, making roads safer. The introduction of self driving cars represents a groundbreaking convergence of robotics and artificial intelligence (AI) technologies, marking a transformative era in transportation.

In the context of Nepal, the vehicle industry hasn't taken off, there has not been any significant efforts in the industry to make it happen either. The implementation of self-driving cars (SDCs) faces challenges in Nepal due to its diverse and unpredictable environment. The country's rugged terrain, lack of standardized road infrastructure, and dynamic conditions make it difficult for SDCs, which rely on precise mapping and AI adaptability. Overcoming these hurdles requires a specialized approach to ensure the effectiveness and safety of autonomous vehicles in Nepal's unique driving landscape. The integration of robotics and AI in SDCs represents a strategic evolution aimed at creating a more efficient, safer, and sustainable future for transportation. The synergy with AI is indispensable, as it empowers these self-driving cars to perceive and interpret their surroundings, make real-time decisions, and adapt to dynamic driving conditions. Machine learning algorithms enable these vehicles to continuously improve their performance by learning from vast amounts of data generated during various driving scenari

1.2 Problem Statements

Following are the question that drove our interest in the project. Our works will be based on the questions below:

- How does an actual vehicle system work?
- How can we control the basic locomotion of a car?
- How does the system perform using feedback mechanism?
- How can we make a system reactive to its environment?

- Can the system be reliable enough to handle its errors on it own?

1.3 Objectives

- To build an autonomous vehicle capable of moving from one location to another location, stay on route and make decisions about changing routes that cause least damage to itself.
- To build the autonomous vehicle capable of obstacle detection and obstacle tracking.
- To design a robust electronic system.
- To show an appropriate project management scheme covering all the important aspects of project management and work record keeping.

1.4 Scopes

For making the project well defined and achievable, we define the coverage and limit sets for our goals which are listed below:

- Working environment for the system will have well knowledge about track/path.
- Resetting the PID controller's integral term to zero in the presence of high output values to enhance stability.
- Include a dedicated emergency stop switch as a critical safety feature in the event of a vehicle malfunction.

2 Literature Review

2.1 A Brief History of Autonomous Vehicles

The first traces of modern autonomous navigation technology can be found in the rovers, space probes and missile systems starting from the late 1960s. The technologies developed in this era, no doubt accelerated by the Cold War and space race, permeated slowly to cars as well. In the 1980s, Ernst Dickmanns, a German pioneer, and his team from the University of Bundeswehr retro-fitted a Mercedes Benz van into VaMoRs, an autonomous vehicle that used computer vision to analyze road-facing camera images [20]. Dickmanns' success led to the large PROMETHEUS project which led to a decade of steady improvements, culminating in the VITA vehicle, which set the then record for autonomy by completing an almost 1000-mile trip at 95% autonomy. Across the pond, the NavLab 5 vehicle from Carnegie Mellon University also demonstrated autonomous driving, setting a 98.2% autonomy record [CMU].

A distinctive inflection point in the history of self-driving cars is the DARPA challenge [DAR] held in 2003, 2004, and 2007. The 2003 edition concluded with none of the participants able to complete the challenge. 2004 saw Stanford's Stanley win, led by Sebastian Thrun. CMU won the 2007 challenge.

A more complete survey of recent methods can be found in [Bad+19].

SAE International, formerly the Society of Automotive Engineers, defines six levels of driving automation, ranging from no driving automation (level 0) to full driving automation (level 5) [Com21]. This taxonomy is widely used throughout the industry to classify current methods and compare them against one another.

2.2 State Estimation

An important part of every self-driving system is the fusion of data from multiple sources to estimate the state of the vehicle. Sensor fusion is usually done using filters.

2.2.1 Kalman Filter

The Kalman Filter [Kal60] is a Bayes optimal filter for the estimation of random variates that follow the normal distribution, and whose process, control, and sensor models are linear.

The Kalman Filter assumes a Markov model for the evolution of the state, such

that the state of the robot, x_k is dependent on the last state $\hat{x}_{k-1}$ and the control inputs u_k. The process model relates $\hat{x}_{k-1}$ with x_k, and the control-input model B_k relates u_k to a corresponding change in the state brought about by the input. Thus, the state evolution model is as follows, including a process noise w_k to model process uncertainty.

$$x_k = F_k\hat{x}_{k-1} + B_k u_k + w_k$$

Readings from the sensors are integrated into the estimate using the observation model corresponding to the sensor. An important detail to note is that this integration is done not in the state space of the system we are tracking, but in the space of sensor readings. A Kalman Gain is computed by comparing the sensor reading to a predicted sensor reading, which is then used to produce a new estimate.

$$\hat{x}_k = (I - K_k H_k)x_k + K_k z_k$$

Two assumptions lie at the heart of the Kalman Filter:

- The state variables and sensor readings are normally distributed, with some known mean and covariance.
- The process model, sensor-input model, and the control-input model are all linear transformations.

These assumptions make the update and estimation steps tractable since the normal distribution has several properties that favor its use. Chiefly, normal variates are closed under addition, multiplication, and affine transforms, and the conjugate prior of a normal distribution's mean is another normal distribution itself. Hence, all models used in Kalman Filters are linear.

2.2.2 Non-Linear State Estimation

Nonlinear systems pose a problem because a normal variate transformed through a nonlinear function does not result in a normal variate. For nonlinear systems, two main extensions to KF are popular: the Extended KF [23] which uses locally linearized versions of all non-linear models using Taylor Series extension upto the first-order terms, and the Unscented KF (UKF) [WV00] which transforms a small set of so-called sigma points through the non-linear models, and then fits a normal distribution on the resulting transformed points.

2.3 Perception

Localization, mapping, and perception have been very active areas of research in the last few decades. For localization using a 2D lidar, a representative method is outlined in [Hes+16]. Visual localization, and the more general problem of Simultaneous Localization and Mapping, have been approached through multiple ways, but a popular and robust method based on matching features in images that have some desirable invariant properties, is ORB-SLAM [MMT15]. Perception involves the detection of obstacles, classification of drivable areas in images, and scene understanding in general. We refer the reader to [RF18] [RFB15] [LLL22] for a few representative methods.

2.4 Control

Good control of actuators like steering and torque on the wheels is crucial to the technology. Two popular approaches are the Proportional-Integral-Derivative (PID) controller [Min22] and Model Predictive Control [GPM89].

2.4.1 PID Control

PID is a simple and perhaps the most widely used controller for linear systems. It has three terms, corresponding to the error, its integration over time, and its time derivative. The weights for each of these terms are used to control their impacts on the overall control output. The overall control output function of the PID controller in continuous form is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau \mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

In practice, a discrete formulation is used more often, which approximates the integral using quadrature and the derivative using finite difference. The selection of the three coefficients is done through a process called PID tuning. There is a range of techniques for tuning PID controllers, from manual tuning to simple rules on observed system response, to automatic in-loop algorithms that select the best parameters using genetic optimization or fuzzy search [BSS12].

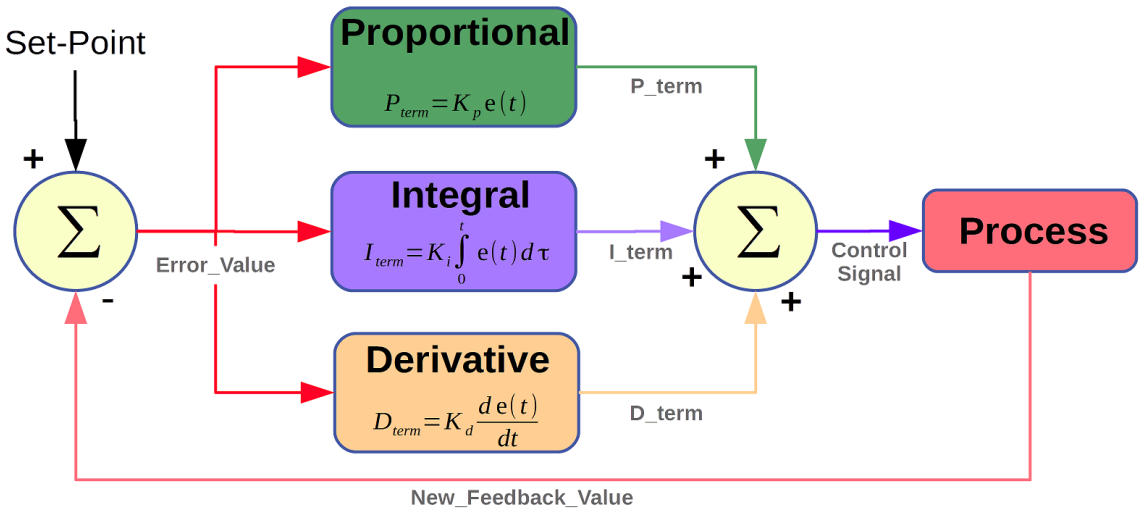


Figure 1: PID Block Diagram

2.4.2 MPC

Model Predictive Control is a controller that uses a model of the system for predicting the system states up to a finite time horizon. It then computes an optimum control output for achieving the defined set point, subject to constraints.

The general MPC framework involves minimization of the following cost function

$$J = \sum_{t=k}^{k+p} W_s(x_t - r_t) + W_c \Delta u_t^2$$

W_s weights the tracking error between reference state r_t and the predicted state x_t, while W_c controls the strength of regularization on the control inputs $u(t)$ by penalizing the derivative. Other terms may be added to this formulation to control required variables, such as jerk and path curvature. p is the time horizon. The states x_t are predicted using the model, which may be linear or non-linear. Depending on the convexity and linearity of the terms, various optimization methods may be used.

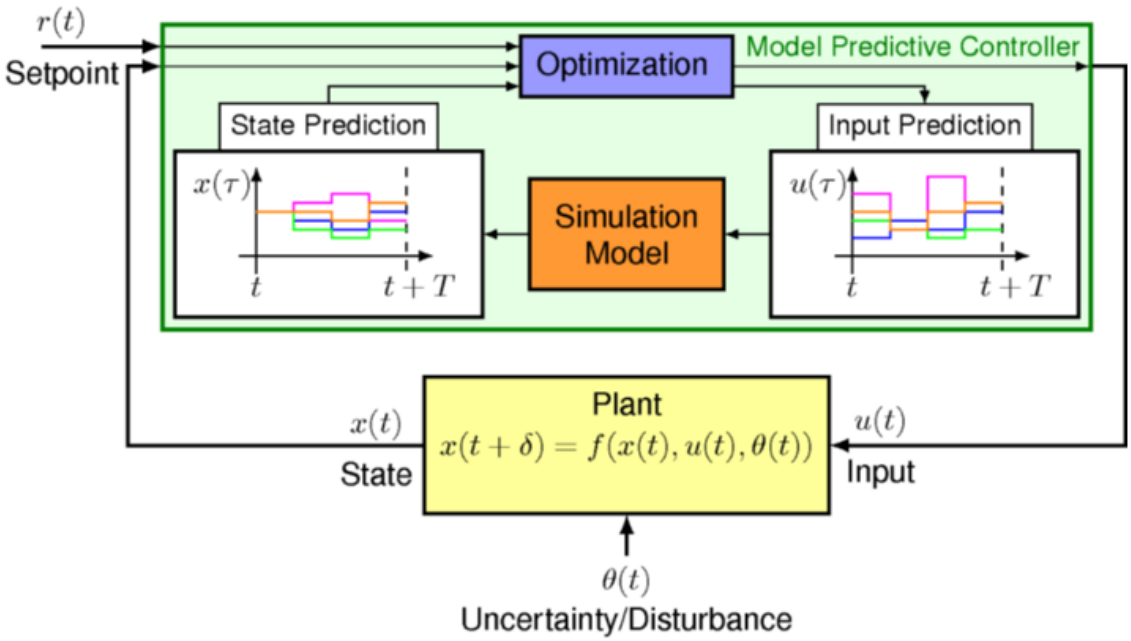


Figure 2: MPC Block Diagram

2.5 Social Impacts and Ethics

As with any new technology, it is important to consider the social impacts and ethics of SDC technology. Keeping in mind the small scope of the project, we defer the discussion of a more general social and regulatory view on SD technology to [McL+22].

3 Robot Operating System(ROS)

3.1 Nodes

Nodes are computational units in ROS Graph. In a working robotic system multiple nodes work together.Each node can send and receive data from other nodes via topics, services, actions, or parameters.

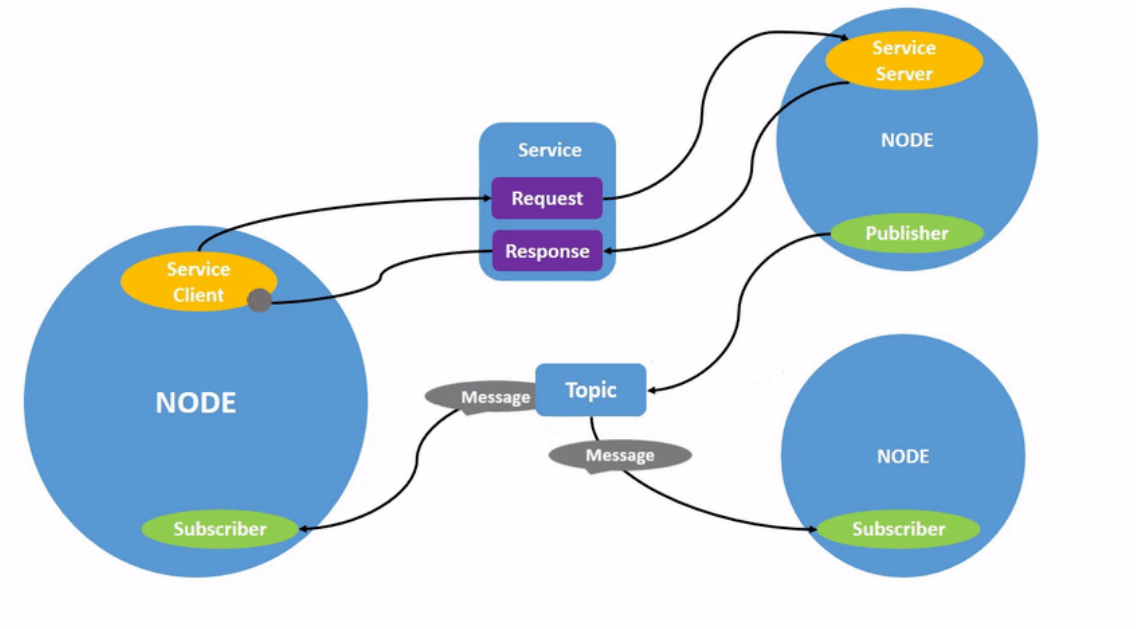


Figure 3: ROS Nodes

3.2 Topics

Topics are intermediate for connection between multiple nodes. Nodes associating with a topic can either publish or subscribe it. A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics. Topics can establish one-to-many, many-to-one or many-to-many relationships.

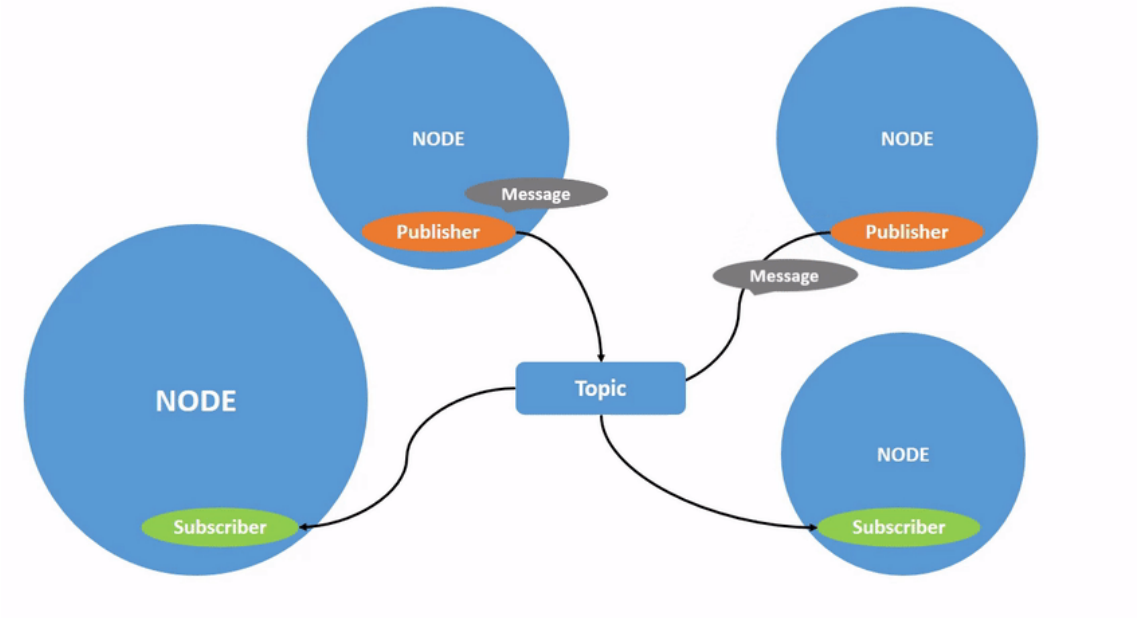


Figure 4: ROS Topics

3.3 Services

Service is another method of communication between nodes. Its works on Request-Response model. By the use of service the nodes provide data only on request from other nodes.

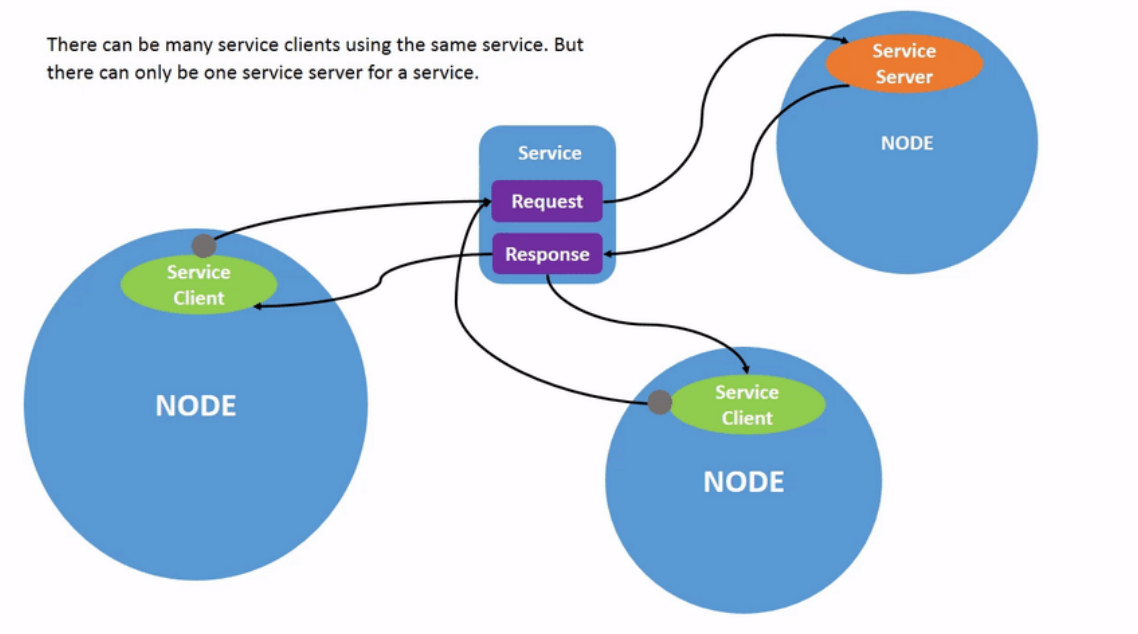


Figure 5: ROS Service

3.4 Parameters

A node is associated with its individual parameters. Parameter is used to configure nodes at startup without changing the code of the Nodes. The lifetime of a parameter is tied to the lifetime of the node.

3.5 Actions

Actions are another types of communication which are used for long running tasks. It consists of 3 parts: a goal, feedback, and a result. Actions are based on topics and services. Actions use a client-server model, similar to the publisher-subscriber model. An "action client" node sends a goal to an "action server" node that acknowledges the goal and returns a stream of feedback and a result.

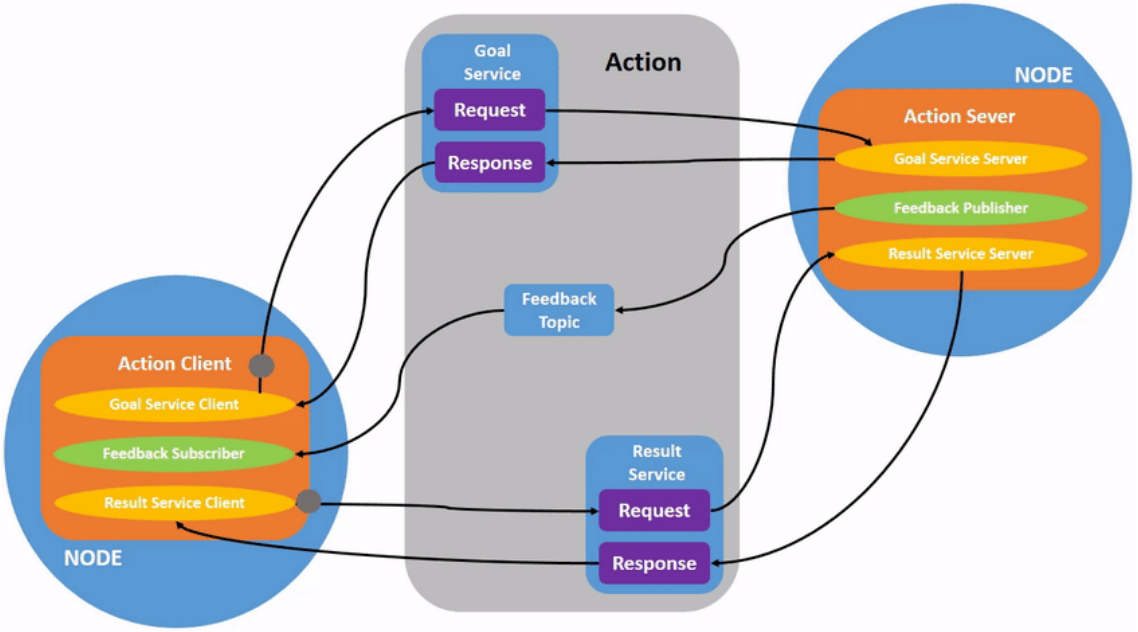


Figure 6: ROS Actions

3.6 Packages

A package is a bundle of ROS codes. It consists of multiple node and the nodes outside the package are connected through various means describes above with a specific message format enabling the package to operate for specific task in the system. The packages can also be released to allow others to build and use it easily.

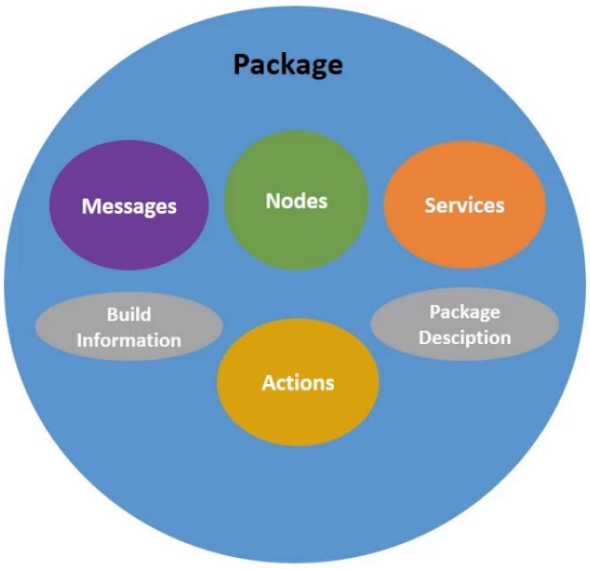


Figure 7: ROS package

4 Methodology

4.1 Scale of the Project

We have developed a small electric toy car with dimensions 100cm, 55cm, and 45cm (l, b, h). The vehicle is equipped with DC motors to drive the rear wheels, while the front wheels is connected through a bar linkage system controlled by a servo for steering. To power the motors, we had used the Lippo batteries, securely positioned in the central area of the chassis, specifically in the back seat of the car.

The vehicle design takes a lot of research and work. The difficult aspect before any mass production is creating the vehicle itself. We are not looking to mass-produce this vehicle and neither we want to spend time on any aesthetics of the system. So taking that out of the equation, still the main aspect of any vehicle is the chassis. Initially, the chassis contained a steering mechanism issue, and the existing chassis was equipped with a small low speed DC motor with no feedback. To address this, we replaced the current motor with a 775 motor with seperated 3D printed gear attached to the optical rotary encoder for the feedback. Additionally, we designed a steering mechanism to be controlled by servo motor.

As listing our tasks, we have completed the tasks from Printed Circuit Boards(PCB) designing and circuit layout to the intricate aspects of sensor placement, communication design, and embedded firmware development. The project further entailed expertise in real-time operating systems, proficiency in CAN protocols, and the intricate process of system integration. Moreover, our scope extended into the realms of Visual SLAM (Simultaneous Localization and Mapping), leveraging technologies such as Hector SLAM. The integration of mapping techniques and the utilization of ROS(Robot Operating System) frameworks was pivotal in orchestrating a cohesive and efficient self-driving car system. It is sure that this is a very engaging project and we populated most of our effort in it.

4.2 Autonomous Region and Environment

For the vehicle to navigate within the confines of our campus specifically along the road encircling the Robotics Club block, we accounted for various environmental factors. This includes negotiating shaded areas created by trees, sunlight stretches, and the presence of students strolling along the road, with bicycles parked on one side. Operating autonomously during the daytime, our SDC heavily relied on Camera data, incorporating visual slam technology to ensure precise navigation. We

addressed crowded scenarios, particularly when roads are densely populated with pedestrians, our SDC will implement strategic U-turn maneuvers. Our approach encompasses robust solutions to effectively navigate these diverse environmental conditions, ensuring the safe and reliable operation of the autonomous vehicle.

4.3 Absolute and Relative Positioning of the System

We need to keep track of our vehicle in order to command it to where to go in real-time. For this, we need to know the position of the vehicle. By fusing information from the IMU and rear wheel encoders and applying the Unscented Kalman Filter [Mat18], we achieve precise and instantaneous positioning feedback. In the long run, utilizing Lidar and Camera data, we will implement Visual SLAM technology to establish and maintain the vehicle's position accurately over time. But instead of doing this, we integrate local planning techniques like road segmentation, and obstacle detection using YOLOv8 and Lane-Keeping to generate a map which will be used for the navigation of the robot.

4.4 Team and Work Division

- Mechanical Work

  Each and every modification of hardware chassis of the car was accomplished with the tools and resourcees available in the Robotics Club. Junior members from the Robotics Club offered assistance in the system modification process.

- Electronics and Embedded Systems

  This part mostly consisted of electronics hardware design like PCB designing and fabrication. Our team used PCB Design software: Altium for PCB Designing. Fabrication of PCB will be done by outsourcing it to PCBWay. Component placement on the printed board will be done with the tools available in the Robotics club.

- Control System Design

  The Control System Design was used in the velocity tuning of the motor used in the wheels of the car for vehicle control and stability.

- Perception Implementation

  Perception problem requires training of the model and data selection to suit our environment and system. Optimization of the model is crucial for fast

inference. We collected different datasets of the track around the Robotics Club and annotated the datas and trained to the yolov8 model and this model will be used for detecting the obstacle and segmentation of the road for finding the drivable region.

4.5 Equipment, Tool and Devices

- **Microcontroller** We used STM32F407 microcontroller based on ARM Cortex-M processors having best performance levels and peripherals for our needs of different applications.

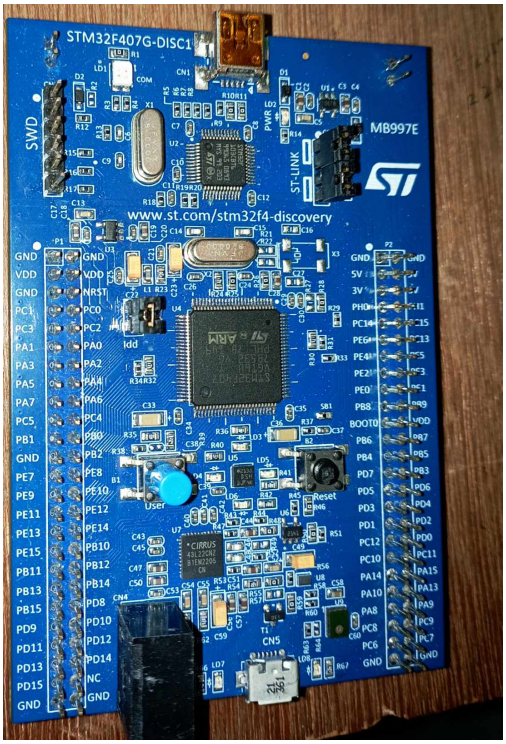


Figure 8: STM32F407 Discovery

- **Raspberry Pi**

  We used Raspberry Pi 4 model B, a single-board computer used for most of the major operations of the system like perception of the car and supporting role for estimating the pose of the vehicle.

Figure 9: Raspberry PI

- **Inertial Measurement Unit**
  We used Inertial Measurement Unit (MPU6050) for the estimation of the vehicle. In our setup, we have IMU sensors such as MPU6050.

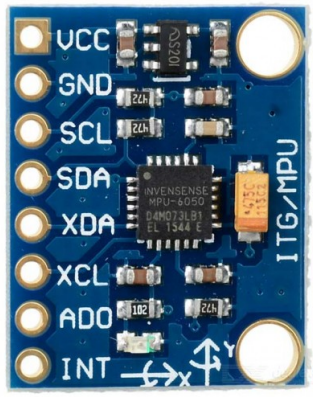


Figure 10: MPU6050

- **Camera**
  For perception in a self-driving car (SDC), we choosed the Logitech C270 HD Webcam. This webcam is capable of recording videos at 720p resolution with a frame rate of 30 frames per second (fps).

Figure 11: Logitech C270 Camera

- **Lidar**

  For perception in an SDC, we have chosen the A1 RPLidar. The A1 RPLidar is a lidar sensor known for its reliability and cost-effectiveness in providing essential environmental data for autonomous vehicle systems.

Figure 12: A1 RP Lidar

5 System Design

5.1 System Overview

The system is designed around two controllers: a main controller based on Raspberry Pi, and a sub-controller based on an STM32 Microcontroller. The controllers are connected to each other via a UART bus. The main controller hosts the processes for perception, localization, and navigation. The sub-controller manages the actuators and sensors.

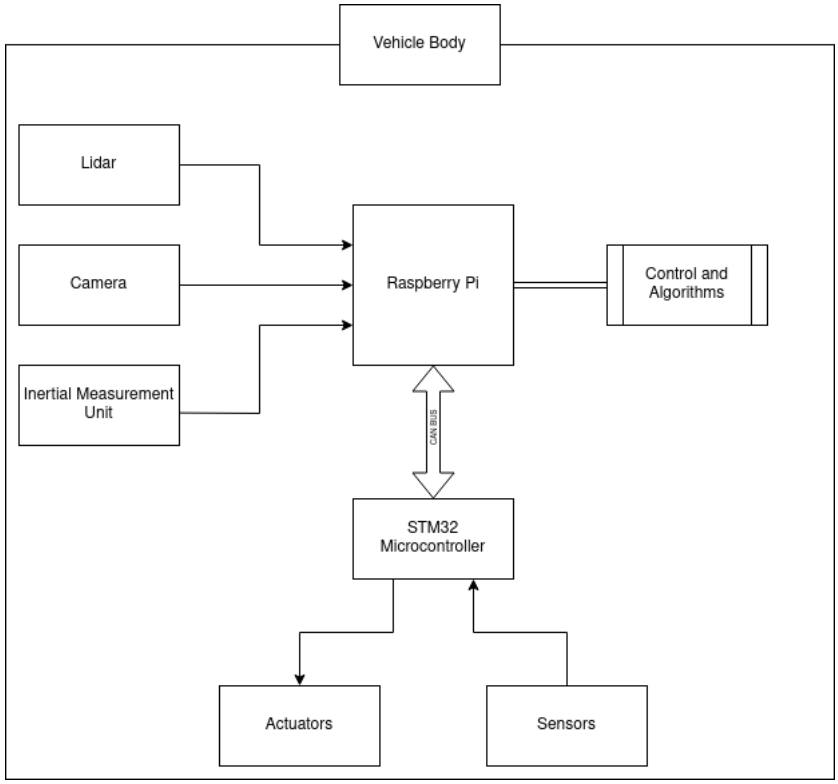


Figure 13: System Overview

The high-level design of the system is inspired by Stanley [Thr+06]. The system is composed of 4 layers: Sensor layer, Perception Layer, Control layer, and Monitor layer. These layers are logical, and each is made of a number of independent, concurrently running nodes. The life-cycle of these nodes, and communication between them, is facilitated by ROS [Mac+22]. ROS acts as a middleware for these nodes to communicate, while also providing a standard programming model for creating and running nodes. Nodes can be written in a variety of languages, but Python and C++ are popular.

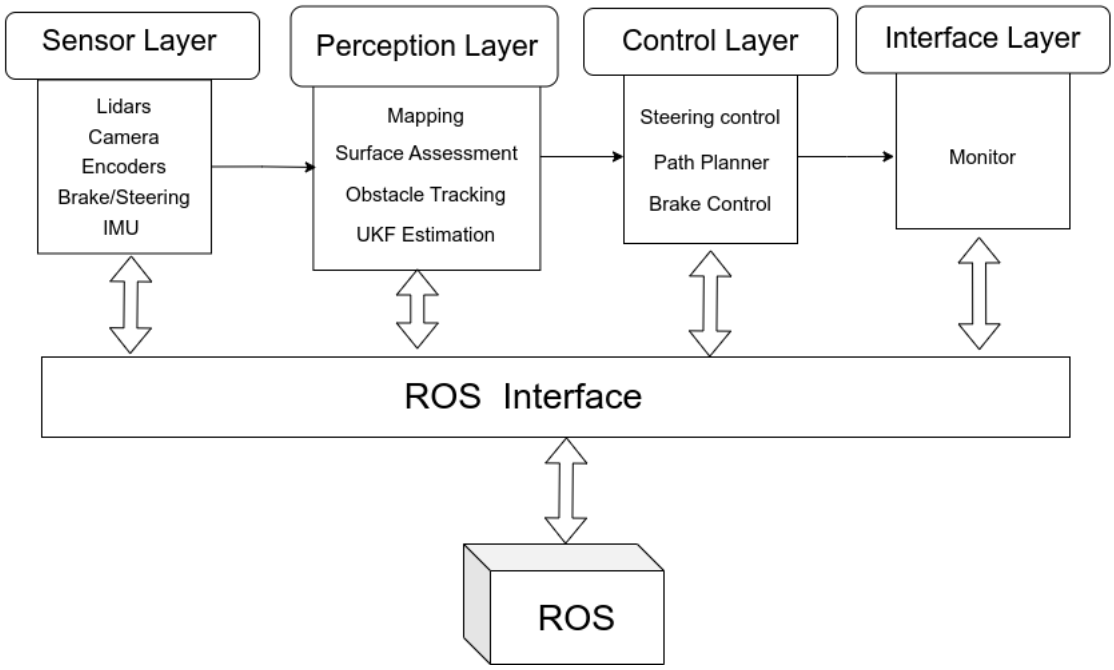


Figure 14: System Layers

5.1.1 Main Controller

The System incorporates Raspberry Pi as the main controller. It runs Robot Operating System (ROS) for managing and monitoring the overall operation of the system. Devices like Lidar, Camera and Inertial Measurement Unit (IMU) are connected with to the Raspberry Pi. Their outputs are consumed via ROS drivers, which publish the received data to topics using standard ROS data types. For perception tasks that require more computation, we will supplement the Pi with a laptop computer with a GPU.

The ROS nodes can be as granular as required. Some of the parts of the system that will be implemented as nodes are:

- Localization
- Mapping
- Perception and Scene Understanding
- Obstacle Detection and Tracking
- Path Planning
- Control Algorithms
- Sensor fusion Algorithm

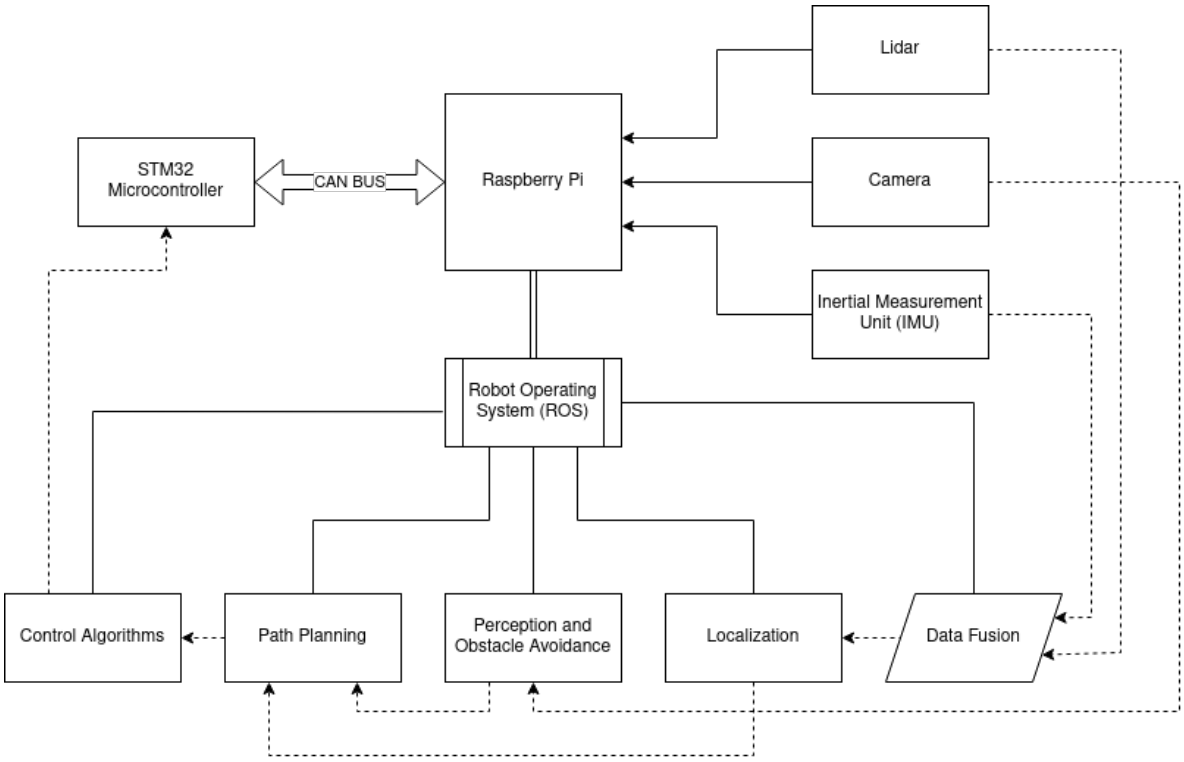


Figure 15: Raspberry Pi as Main Controller

5.1.2 Sub-Controller

STM32 Microcontroller is used to control drive-train actuators for the motion of the system. The motor is connected to the motor driver controlled by STM32 using a PID controller which takes feedback from the encoder and additional inputs from the UART bus. The FreeRTOS system is implemented in STM32 for task management. Individual algorithms are threaded in order to create a stable and responsive real-time system.

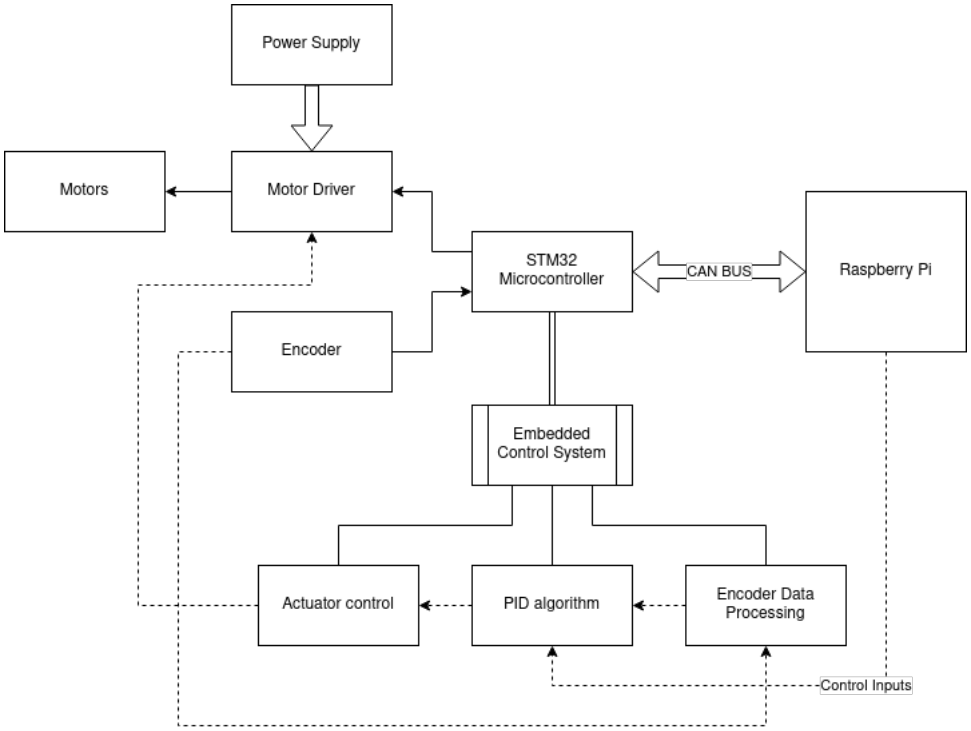


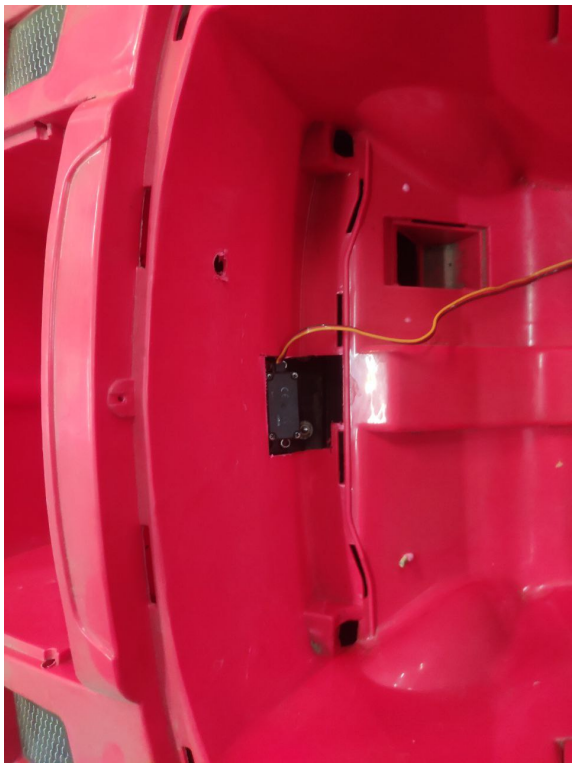
Figure 16: STM32 as Sub Controller

6 Tasks Completed

6.1 Hardware

6.1.1 Steering Mechanism

We implemented the Ackermann Steering Mechanism, customizing it for a readily available basic toy car. This required modifying the steering system to incorporate a servo motor, which is controlled by a microcontroller to provide signals.



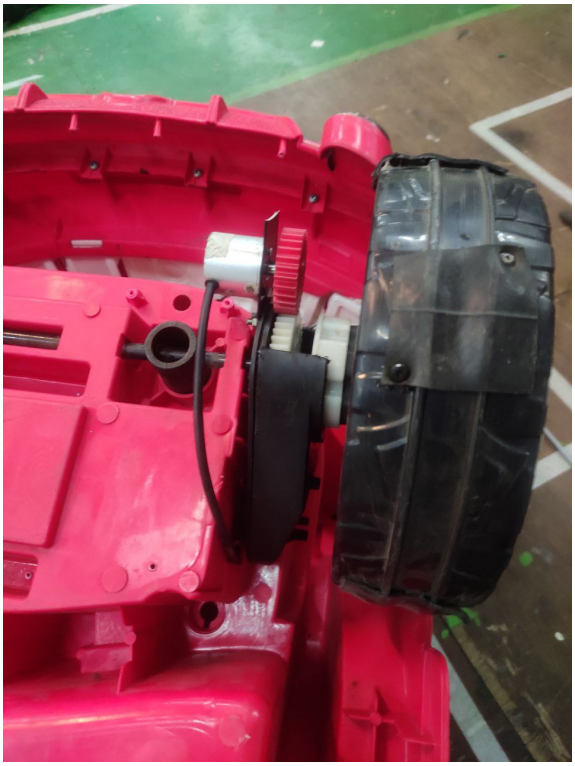
(a) Steering with Servo

(b) Steering with Servo

6.1.2 Drive-shaft Repair

The drive shaft and gears required repair due to their unusable state. We used 3D printed parts for the repair and added encoders to the drive shaft for additional functionality.

(a) Steering with Servo



(b) Steering with Servo

6.1.3 Circuitry and Wiring

Circuit Design: We designed a PCB housing multiple integrated circuits (ICs) to fulfill our needs, including an STM32 IC alongside several IMU sensor chips. Nevertheless, owing to prolonged fabrication times in China, we opted for off-the-shelf components for testing and initial implementation phases.

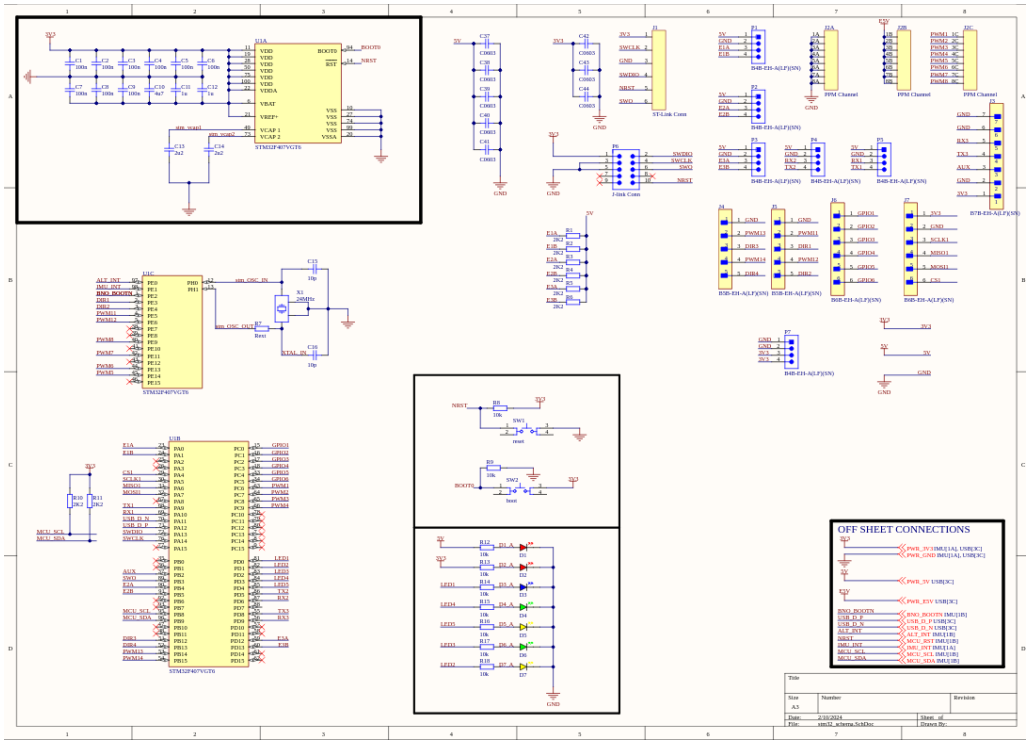


Figure 19: STM32 Schematic

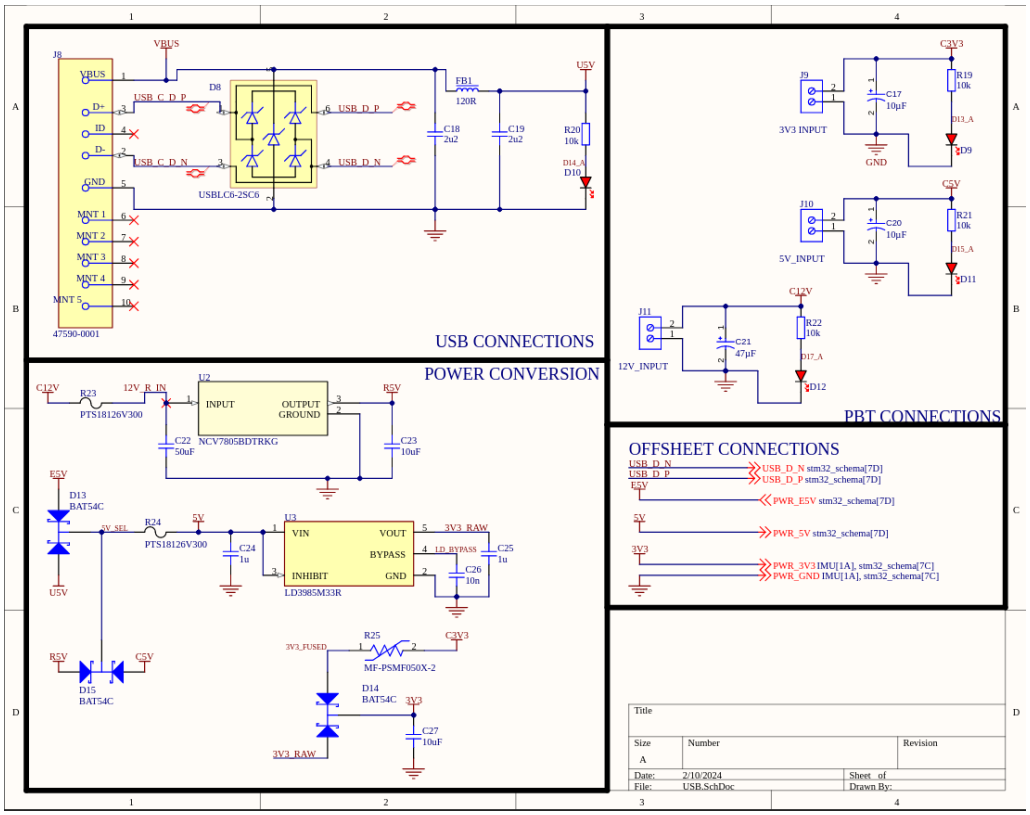


Figure 20: USB and Power Schematic

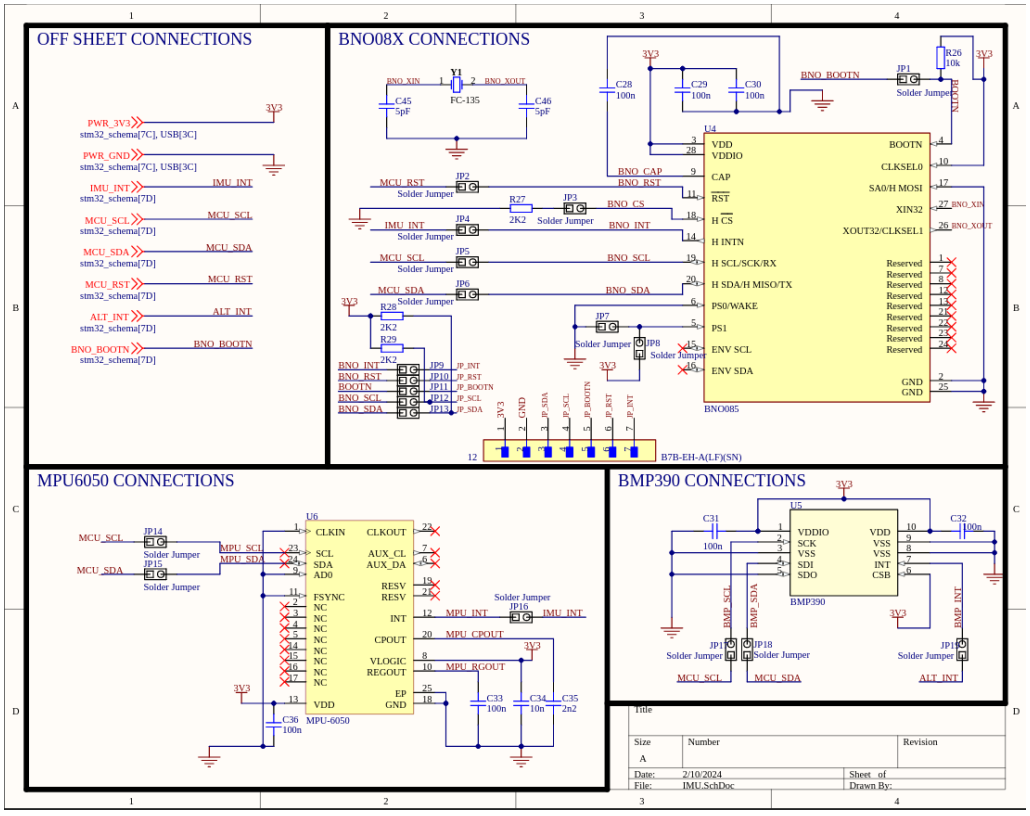


Figure 21: IMU Schematic

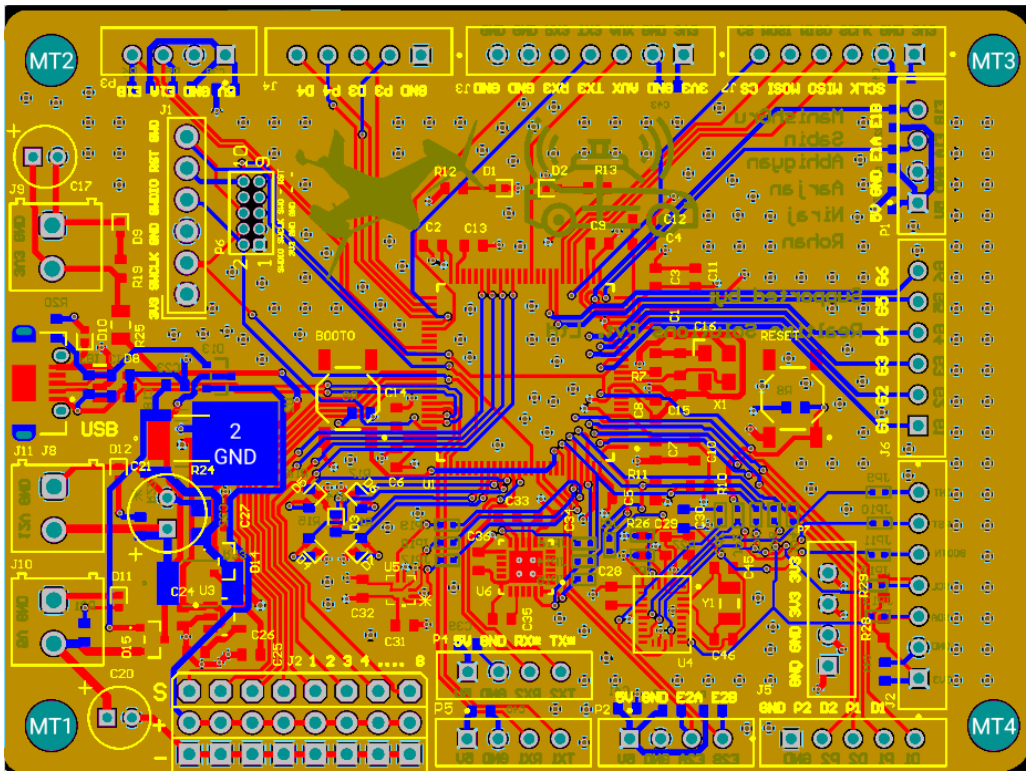


Figure 22: PCB Design

Wiring: For an interim measure, the system is configured using a pre-existing PCB featuring an STM32 microcontroller linked to peripherals such as encoders and motor drivers. This microcontroller interfaces with a Raspberry Pi via UART bus. Additional sensors such as cameras and Lidar for advanced system operations are connected to the Raspberry Pi using their respective buses.

6.2 Software

6.2.1 Navigation Stack

We tried controller models for both Ackermann geometry and Bicycle geometry model. While the Ackermann model is closer to the actual geometry of the drivetrain we have in our car, we found it to be much more sensitive to parameters as compared to the Bicycle model. The bicycle model is additionally simpler to understand.

We also established a ros_control system for the control of the car. ros_control provides a unified pipeline for commanding actuators, and gathering feedback from them.

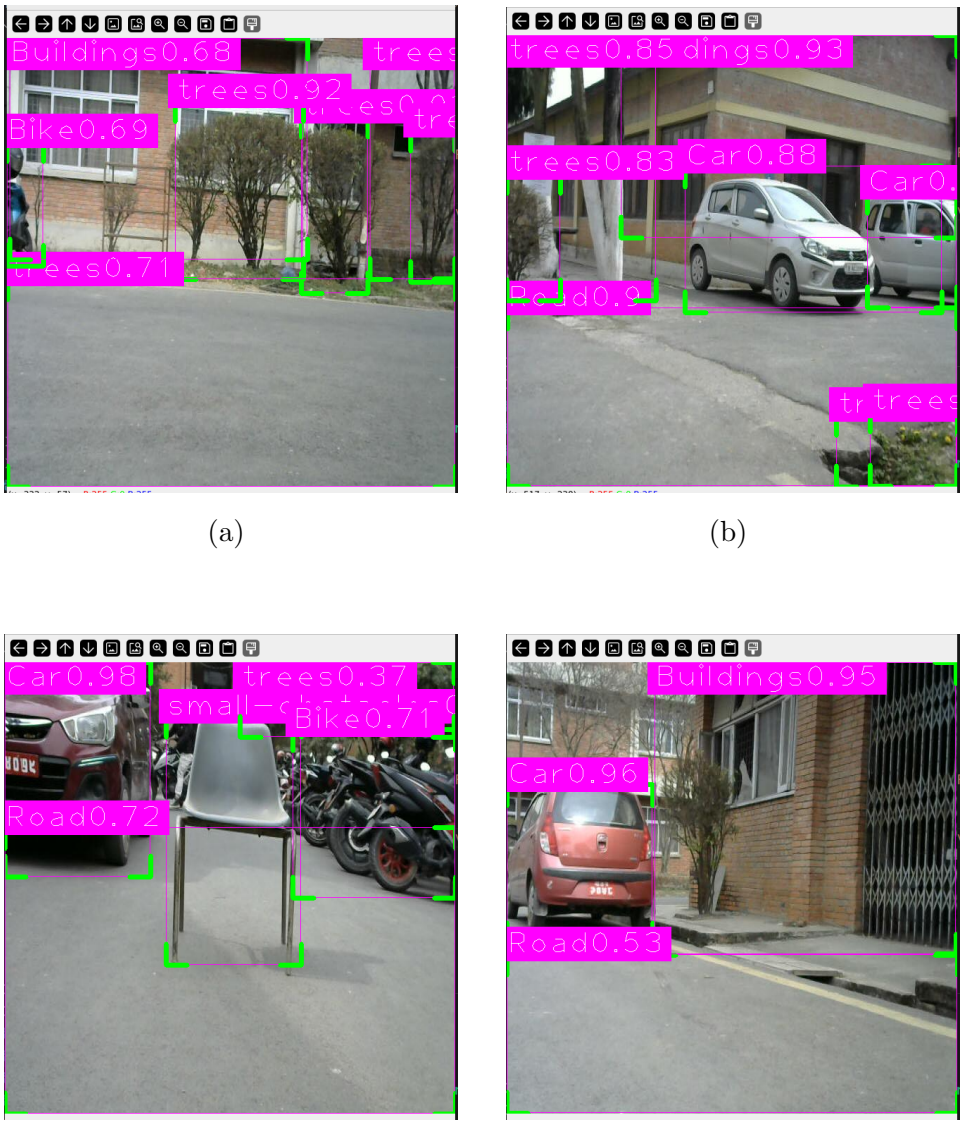
Finally, we completed preliminary testing of occupancy grid map calculation using SLAM-TOOLBOX. We also tried to implement different ways to treat obstacles detected by the lidar and camera.

6.2.2 Visual Perception Models

We have been fine-tuning a YOLOv8 model for object detection and semantic segmentation using a custom dataset collected within the college premises. The process involves iterative adjustments to the model architecture and hyper-parameters. The goal is to optimize detection and segmentation performance on our specific domain. The initial models are ready, but this is an iterative process and we'll keep curating the dataset and run more training runs to refine the motor. Some example predictions are shown in 23.

6.2.3 Motor Control

The entire navigation stack running on ROS2 depends on the car being able to track reference wheel velocities and steering angle closely. Tracking steering angle is not a huge problem because we use a servo motor. Tracking the reference velocity for the two rear motors is therefore of our primary interest here. We have set up PID controllers for motor speed control. An encoder is coupled to each of the two rear wheels so that they can be controlled independently in a closed loop fashion. We



(a) (b)

(c) (d)

Figure 23: Example Predictions on Dataset Taken in the College

have completed tuning the PID controllers for each of the wheels for the current load distribution in the car. As the load distribution shifts throughout the project, we'll have to fine-tune the PID parameters.

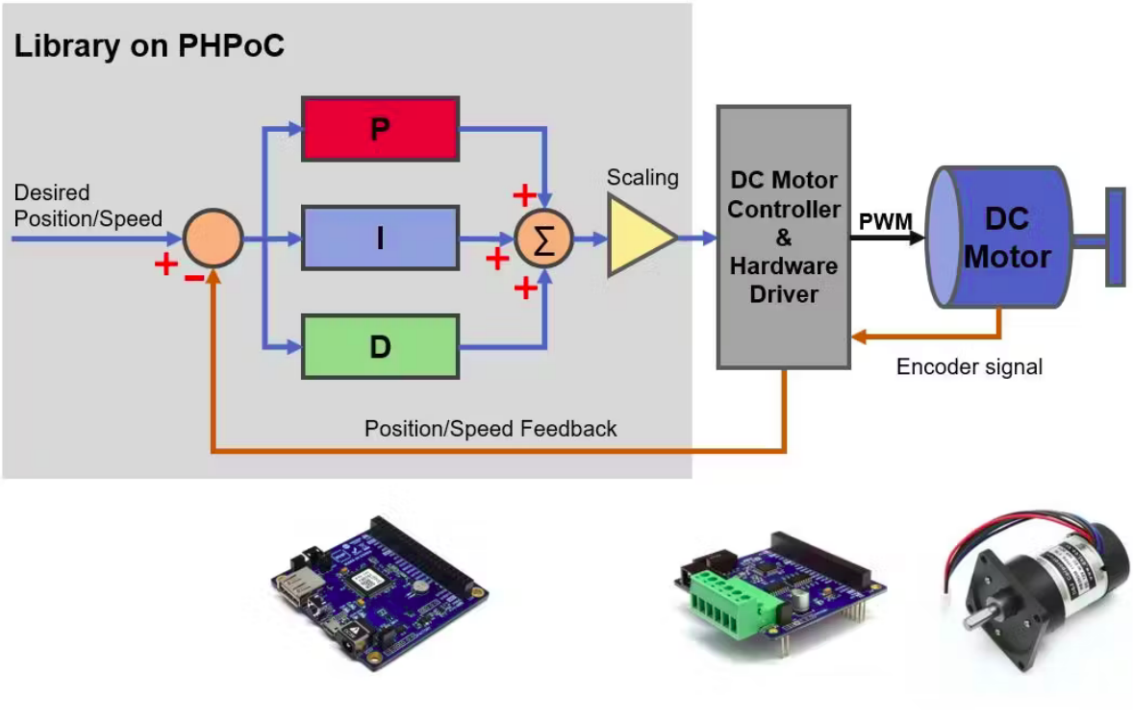


Figure 24: Motor Control PID Block Diagram

7 Tasks Remaining

7.1 Mapping

Our mapping and localization strategy involves a visual map for global localization. But because we only have a single monocular camera, we can recover the global map up to a scale term. We compensate for this by using externally computed odometry, which allows us to use two views from the same camera to simulate a stereo pair.

Furthermore, using the 2D Lidar, we also create a global occupancy map of the world. We can then process this map to remove non-static parts of the map, such as cars. At driving time, we can localize the car using the visual map, and then use the occupancy grid for global planning.

7.2 Odometry

We have many sensors that give some form of odometry. This list includes an Inertial Measurement Unit (IMU), Lidar, camera, and wheel encoders. We need to fuse the information from all these sensors into a good odometry estimate using a sensor fusion algorithm, such as the Extended Kalman Filter [23]. The fake-stereo setup for mapping also depends depends on good odometry, so this step is crucial.

7.3 Fusion into local occupancy grid

We need to create nodes for producing each of the costmaps that will influence driving. This includes a projection system that creates a drivable region costmap in bird's eye view, a costmap to make the vehicle prefer the left side of the road, and a costmap based on the occupancy grid generated by the lidar. Then these costmaps will need to be fused into a consistent local representation for local planning.

7.4 Perception Models

Three important tasks, viz. obstacle avoidance, road keeping, and lane preference, depend on good perception models. Efforts here will need to be split into two main areas: object detection, and semantic segmentation. Training and tuning models is an iterative process, and the goal here is to push the model performance as much as possible to the end.

7.5 Navigation Testing and Tuning

Each of the components above will need to be integrated with the Nav2 stack [**macenski2023survey**]. Furthermore, the whole navigation system will need to be tuned and tested. Transforms and models will need to be calibrated to the end, as the physical model of the car will change and shift throughout the project, and wrong parameters will lead to subpar results.

References

[20] *Oral-History:Ernst Dickmanns*. en. Dec. 2020. URL: https://ethw.org/Oral-History:Ernst_Dickmanns (visited on 12/20/2023).

[23] *Extended Kalman filter*. en. Page Version ID: 1185257110. Nov. 2023. URL: https://en.wikipedia.org/w/index.php?title=Extended_Kalman_filter&oldid=1185257110 (visited on 12/20/2023).

[Bad+19] Claudine Badue et al. *Self-Driving Cars: A Survey*. arXiv:1901.04407 [cs]. Oct. 2019. DOI: 10.48550/arXiv.1901.04407. URL: http://arxiv.org/abs/1901.04407 (visited on 12/20/2023).

[BSS12] Hari Om Bansal, Rajamayyoor Sharma, and PR Shreeraman. "PID controller tuning techniques: a review". In: *J. Control Eng. Technol* 2.4 (2012), pp. 168–176.

[CMU] CMU. *Navlab 5 Details*. URL: https://www.cs.cmu.edu/~tjochem/nhaa/navlab5_details.html (visited on 12/20/2023).

[Com21] On-Road Automated Driving (ORAD) Committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Apr. 2021. DOI: https://doi.org/10.4271/J3016_202104. URL: https://doi.org/10.4271/J3016_202104.

[DAR] DARPA. *The Grand Challenge*. URL: https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles (visited on 12/20/2023).

[GPM89] Carlos E. García, David M. Prett, and Manfred Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), p. 335. ISSN: 0005-1098. URL: https://www.academia.edu/6160090/Model_predictive_control_Theory_and_practice_A_survey (visited on 12/19/2023).

[Hes+16] Wolfgang Hess et al. "Real-Time Loop Closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[Kal60] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[LLL22] Fei Liu, Zihao Lu, and Xianke Lin. *Vision-Based Environmental Perception for Autonomous Driving*. arXiv:2212.11453 [cs]. Dec. 2022. DOI: 10.48550/arXiv.2212.11453. URL: http://arxiv.org/abs/2212.11453 (visited on 12/20/2023).

[Mac+22] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[Mat18] Fredrik Matsson. *Sensor fusion for positioning of an autonomous vehicle*. 2018.

[McL+22] Scott McLachlan et al. *The Self-Driving Car: Crossroads at the Bleeding Edge of Artificial Intelligence and Law*. arXiv:2202.02734 [cs]. Feb. 2022. DOI: 10.48550/arXiv.2202.02734. URL: http://arxiv.org/abs/2202.02734 (visited on 12/20/2023).

[Min22] N. Minorsky. "DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES". In: *Journal of the American Society for Naval Engineers* 34.2 (1922). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1559-3584.1922.tb04958.x, pp. 280–309. DOI: https://doi.org/10.1111/j.1559-3584.1922.tb04958.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1559-3584.1922.tb04958.x.

[MMT15] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015). Publisher: Institute of Electrical and Electronics Engineers (IEEE), pp. 1147–1163. ISSN: 1941-0468. DOI: 10.1109/tro.2015.2463671. URL: http://dx.doi.org/10.1109/TRO.2015.2463671.

[RF18] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. arXiv:1804.02767 [cs]. Apr. 2018. DOI: 10.48550/arXiv.1804.02767. URL: http://arxiv.org/abs/1804.02767 (visited on 12/20/2023).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Lecture Notes in Computer Science. Cham:

Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

[Thr+06] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge." In: *J. Field Robotics* 23 (Jan. 2006), pp. 661–692.

[WV00] E.A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Oct. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463. URL: https://ieeexplore.ieee.org/document/882463 (visited on 12/19/2023).