IMPLEMENTATION OF ADAS FEATURES ON ONE TENTH SCALE OF AN AUTONOMOUS VEHICLE

by

Yogitha Davuluri

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Indianapolis, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Lingxi Li, Chair

Department of Electrical and Computer Engineering

Dr. Brian King

Department of Electrical and Computer Engineering

Dr. Yaobin Chen

Department of Electrical and Computer Engineering

Approved by:

Dr. Brian King

This work is dedicated to my parents and my sister for their endless love, support and encouragement.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and honor the support and guidance of my thesis advisor, Dr. Lingxi Li. Without his support, this job was not easy to be done. I would also like to thank Dr. Brian King and Dr. Yaobin Chen for serving on my thesis committee. Likewise, I want to thank Ms. Sherrie Tucker who sincerely and selflessly helped me through every stage of my graduate studies.

Besides all my faculty, I would like to acknowledge LHP Engineering solutions for providing guidance and research resources for my thesis work.

Last but not least, I am highly indebted for the support and fortitude of my family, for helping me get through all the difficulties during these years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

m mass

v velocity

ABBREVIATIONS

ADAS	Advanced Driver Assistance Systems
BEV	Bird's eye view
BGR	Blue Green Red
DC	Direct Current
ED	Edge Drawing
GPIO	General Purpose Input/Output
HSV	Hue Saturation Value
HT	Hough Transform
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
IP	Internet Protocol
IMU	Inertial Measurement Unit
KF	Kalman Filter
LIDAR	Light Detection And Ranging
LIPO	Lithium Polymer
OPENCV	Open Source Computer Vision
OS	Operating System
OSRF	Open Source Robotics Foundation
PI	Proportional Integral
PWM	Pulse Width Modulation
RANSAC	Random Sample Consensus
RC	Radio Controller
ROI	Region of Interest
ROS	Robot Operative System
Rviz	ROS Visualization
SAE	Society of Automotive Engineers
SD	Secure Digital
SPI	Serial Peripheral Interface

TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
2D	Two Dimensional

ABSTRACT

An autonomous car is a self-driving vehicle, that operates without human intervention and has the capability of sensing the environment around it. To achieve this, the autonomous vehicle mostly depends on multiple Sensors, Actuators, Machine learning, complex algorithms and processors for software execution. Developed Software, at that point, processes all the information obtained from sensors, plans the path, and the instructions are passed to the vehicle's actuators, which are capable of controlling acceleration, steering, and brake systems. The rules that are hard-coded, algorithms for detection of object and obstacle avoidance, and predictive modelling control algorithms assist the software with observing traffic guidelines and navigate the vehicle accordingly. Free driving is anything but a simple assignment, and to make independent driving game plans is an extraordinarily critical capacity in the current programming planning field. Engineers and Researchers have been keeping huge endeavors to develop safe and precise algorithms to be incorporated in autonomous vehicles.

ROS is a flexible and perfect middle ware tool for robotic applications. ROS offers the necessary tools to effortlessly get the sensors information, process that information, and produce a suitable response to actuators of the vehicle. This thesis work plans to exhibit how ROS could be utilized as a middle- ware tool to make the vehicle move autonomously by examining the surroundings and taking decision.

The main focus of this thesis is to develop a one-tenth scale of an autonomous Race car equipped with Jetson Nano as the on-board computer, ROS based software architecture, sensors, and a PWM driver and implement ADAS features such as Emergency Brake system, Lane Detection and Lane change on the autonomous Race car vehicle. At last, by following the strategies introduced in this thesis work, it is possible to build and develop an autonomous vehicle that uses ROS framework.

1. INTRODUCTION

1.1 Background

The Automotive industry has worked tremendously in different perspectives with the revelation of better than ever innovations, whether it is in the efficiency of fuel consumption, driver assistance, design and safety factor in the last few decades. Safety factor has been a significant objective of interest, all through these years in the scientific world as guaranteeing the safety of drivers and pedestrians is still a great challenge [1].

As most of the vehicle collisions are caused by drivers because of lack of attention, drowsiness at wheel and belittled safety systems, technologies ensuring road safety and help drivers are of current interest these days. According to [2], around 90% of vehicle collisions are because of driver negligence, the result of which 1.35 million individuals are died every year as per World Health Organization [3].

To resolve this issue, many have intended systems that would automatically drive a vehicle most securely, confirming the safety of every human being involved. Consequently, self-driving cars are likewise designated as autonomous vehicles. Autonomous vehicles are envisioned as a forthcoming transportation system in cities where the individuals will be dropped off to their destination without any pedal press and coordinating the wheel [4].In the current days autonomous driving is not an unreasonable idea as a huge number of these cars have begun its work in cities around the world with phenomenal accomplishment. Nevertheless, these systems are of gigantic intricacy, with a lot of work still to be done until the autonomous vehicles are cruising on roads with full of consciousness of surroundings encompassing them is accomplished. To address above safety problems, Advanced driver-assistance system (ADAS) [5] has come into play.

The international automotive organization SAE International (Society of Automotive Engineers) determined 6 levels of driving mechanization to get the concept of autonomous driving[1]. These levels decide how independent an autonomous vehicle is and the necessities and precautionary measures of the human driver [6].

Level 0, No automation. This level asserts that no automation is available, where driving just depends on human activities in each circumstance and the framework just gives alerts or transient help. Lane Departure Warnings is an example of this level which is present in old generation of vehicles.

Level 1, Driver assistance. This level is liable for helping the driver with a particular undertaking. Instances of it are Lane Keep Assistance (LKA), cruise control, since the framework is equipped for taking responsibility for an activity, such as steering or slowing down.

Level 2, Partial automation. The framework can finish at least two or more programmed tasks, working close by to help driver, but it isn't yet viewed as an independent vehicle and the driver is needed to keep up with the vehicle taken control of and take the important actions if essential. The vehicle performs activities, for example, speeding up or decelerating and steering simultaneously. Instances of are same of level 1, but they work close by and the driver is mindful to direct.

Level 3, Conditional automation. Up from this level, the driving capabilities are taken by vehicle. The framework is liable for the typical activities in driving and just requires the individual to take necessary measures, if the framework needs support or an emergency situation shows up. That being said, albeit the vehicle is driven automatically, the driver should examine traffic and street conditions and be ready and prepared to take control when the framework requests. For example, these highlights can be found in autonomous highway driving or in traffic congestion.

Level 4, High automation. From this level, the vehicle can screen the climate and can work completely independently in a wide range of driving situations. Additionally, as opposed to the level 3, the framework is fit for shielding its travelers from accidents, regardless of whether it required and the driver doesn't as expected intercedes on time. If the framework finds that not all conditions are met, at that point the driver must assume control, until autonomy is met again.

Level 5, Full automation. Equivalent to level 4, the system can drive the vehicle in every driving situation and conditions. It can duplicate human activities and may surpass at it.

All emergency circumstances are managed independently and never requires the human to make moves or to manage. No pedals and guiding are important to be installed.

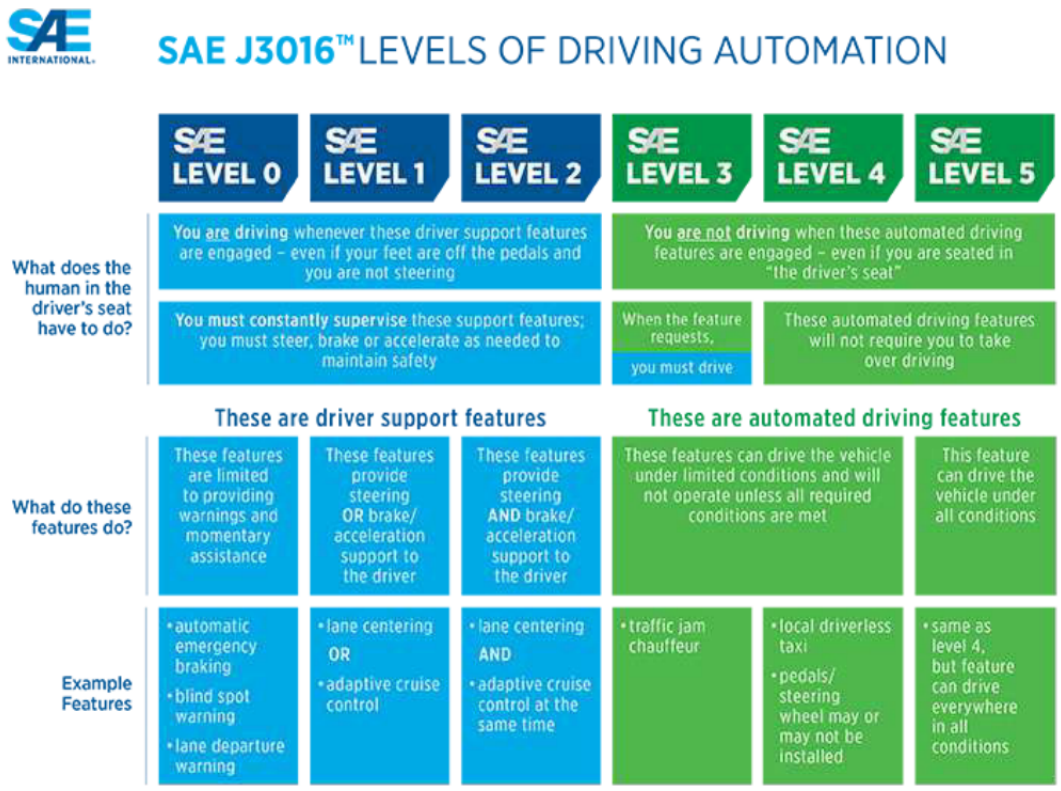

Figure 1.1. Summary of SAE 6 levels of Driving Automation [6]

To sum up, as it tends to be found in Fig. 1.1, one can assert that from level 0 to level 2, the driver is the person who is in full control of the vehicle, person should be totally mindful of each circumstance out of road and is answerable for the well-being of its travelers and other drivers out and about. From level 3 to level 5, the system is the one liable from nearly to all controls of the vehicle. The driver in the front seat should examine all conditions and take responsibility for the vehicle if vital, but it should be ready to make a move in emergency circumstances.

1.2 Autonomous vehicle architecture

There is no concurrence on the right useful architecture for the autonomous vehicle among industry specialists. Regardless, we can comprehensively divide the fundamental components of the independent vehicle, similar to some other machine, into Hardware and Software. Additionally, these two classes are partitioned to extra subsets. Hardware parts are extensively split into sensors, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, and actuators. Software parts are comprehensively split into Perception, Planning, and control. An architecture of an autonomous vehicle is an analogy to a of the anatomy map of human Body.

The core skills of self-driving vehicle hardware framework [7] can be comprehensively arranged into three categories namely sensors, V2X technology and Actuators as portrayed in Fig. 1.1. The hardware components empower the autonomous vehicle to wrap up such responsibilities as seeing (through sensors), communicating (through V2V communication), and moving (through actuators).

The core skills of self-driving vehicle software framework [7] can be comprehensively arranged into three categories namely Perception, Planning and Control, with the communication among these competencies and interaction of vehicle with the environment portrayed in Fig. 1.1 Moreover, to accomplish further enhancements in Perception and Planning, Vehicle to Vehicle (V2V) communication can be utilized.

Sensors are the parts that permit the autonomous vehicle to take in raw data of the surroundings. Sensors resemble eyes of the human body, which empower to comprehend whats happening in the surrounding environmental elements. The principle sensors in autonomous vehicles incorporate GPS/Inertial Measurement Unit (IMUs), camera, LiDar and radar. All of these sensors have their particular benefits and impediments. LiDar, for instance, is extraordinary at catching data in different sorts of encompassing light (regardless of whether night or day), though cameras might experience issues in dealing with specific impediments brought about by shadows or other low lighting conditions. Likewise, most independent vehicles consolidate the readings of numerous sensor types to add additional repetition and make up for the shortcomings of the different sensors in a cycle called sensor fusion [8].

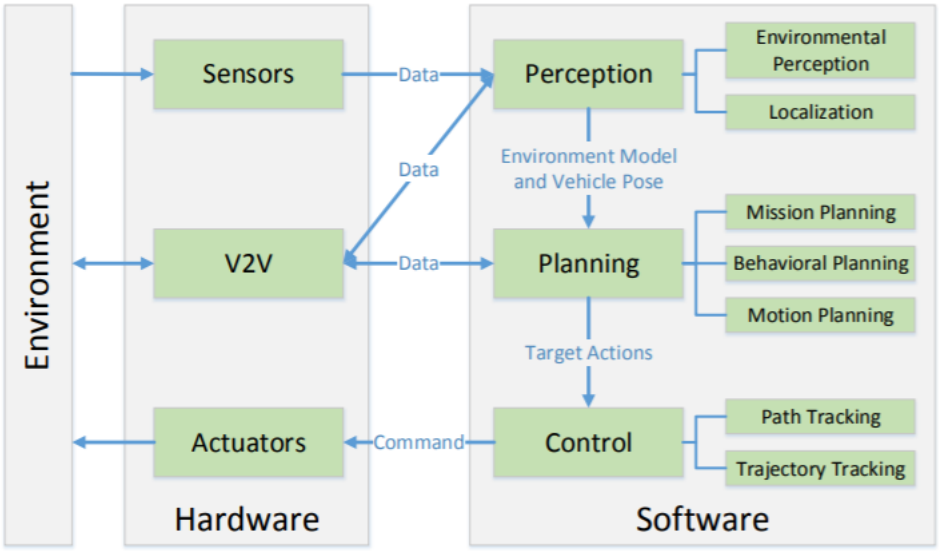

Figure 1.2. Conceptual diagram of Autonomous Vehicle System [7]

Communication technologies such as V2V and V2I empower the independent vehicle to send and get data from other vehicles and infrastructures in the surroundings, for example, sent data from a city light that it has become green or warnings from an approaching vehicle. We can consider V2X communication similar to human parts such as mouth and ears. Mouth permits us to convey information to different people, and ears permit to get what different people are imparting to us [8].

Actuators are the parts of a machine liable for controlling and moving the vehicle. Actuators resemble muscles of human body, reacting to electrochemical signs from the cerebrum so we move arms or legs [8].

Perception alludes to capacity of an autonomous system to gather data and obtain significant information from the environment. Environment Perception alludes to create a context-oriented understanding of environment such as location of objects, road marking/signs detection, furthermore, sorting information by their semantic significance [7]. This interaction is comparable to how our brain cycles the data we acquire through sight into importance. Localization refers to capacity of autonomous vehicle to decide its position with regard to environment.

Planning points to most common way of settling on purposeful choices to accomplish autonomous vehicle's higher order objectives, by making the vehicle move from its start point to final point.Planning framework works by joining the handled data about the climate (for example from the sensors and V2X parts) with built up approaches and information concerning how to explore in the surroundings (for example try not to run over people on roads, dial back when moving toward a stop sign, and so forth) so the vehicle can figure out what move to make (for example surpass another vehicle, how to arrive at the destination, and so forth). Comparably, very much like the planning framework in the independent vehicle, the cycles in the frontal projection of the human cerebrum empower us to reason and decide [7].

Lastly Control category alludes to vehicle's capacity to execute planned activities that are being produced by other elevated level cycles. Here the control framework tells the hardware component (the actuators), the fundamental sources of info that will prompt the specified motions.This is analogous to the role played by cerebellum of human body [7].

In order to design fully autonomous vehicle, large number of challenges must be considered during the development phase of software and hardware of the vehicle. The initial step is to totally change over the driving task into software: The vehicle should have the ability to plan its own roue(Navigation), take along the path (Path Following), and balance out itself(Control) based on the current environmental conditions. Second step includes the fruitful presentation of these algorithms into vehicles which require incredible power and self-diagnosis abilities to deal uncommon circumstances securely [9]. Finally, to guarantee the secure use of vehicle in the real time environment software needs to be tested.

To bridge and test all the requirements of autonomous vehicles, considerations were highlighted on small, minimal expense platforms, which helps in execution of similar approaches applied in full sized autonomous cars.

1.3 Context

In the context of Introduction to Autonomous Mobility, this thesis work has been built and developed at LHP Engineering Solutions.

This test bed gives a self-driving vehicle that grants developing, testing and endorsement of various technologies from simple detection of object and its distance to vision techniques and so on that ensures the safety for Advanced Driver Assistance Systems(ADAS).

1.4 Purpose of Research

The main objective of this thesis work is to develop and build an autonomous race car to create real time scenarios like Object detection, Emergency Brake system, Lane Detection and Lane Change algorithms. The main intention is to build a race car platform to test the ADAS functionalities and other required hardware that are used in Autonomous Driving.

1.5 Research Contributions

The essential contributions of this work are:

- Assembling and adjusting the Traxxas remote controller vehicle into an autonomous vehicle.
- Obtaining the data from various sensors like LiDAR and Camera.
- Implementing different algorithms which uses information from sensors to test on the race car.
- Testing the ADAS features on the race car.

1.6 Thesis Organization

The thesis is organized as mentioned below:

- Chapter 2: Literature Review will be outlined in this chapter.
- Chapter 3: All the software technologies and hardware tools utilized in here will be outlined in this chapter.
- Chapter 4: Both Hardware and Software architecture will be outlined in this chapter.

- Chapter 5: Algorithm Implementation and final results will be outlined in this chapter.
- Chapter 6: All the future work and further improvements will be outlined in this chapter.

2. LITERATURE REVIEW

The paper [10] describes the static Region of Interest (ROI) first for the purpose of Lane detection and Lane Keeping. Within the obtained ROI, using simple filter technique, feature extraction is executed. Based on the thickness level of lanes, the filter has been built and the level of thickness I decided offline along the whole path in terms of perspective view. Then, in order to detect the lane Hough Transform(HT) is applied by taking into consideration, the slopes of the paths. In conclusion, to track the HT parameters, Kalman Filter (KF) has been used.

The work in [11] uses stereo camera to detect the road and then ROI is determined. The features are being extracted by the application of adaptive threshold. For that activity, based on the location of camera, different window size has been considered. Moreover, upon feature extracted image, edge filter image processing technique is applied to estimate the vertical and diagonal edges. At last, to detect the lanes dynamic programming has been utilized.

In [12], after capturing the perspective image the authors transformed the image with a predefined static ROI to a Bird's eye view(BEV) image. From that point onward, to work on the quality of the worn-out lanes temporal blurring technique is utilized and successively first order Gaussian kernel is carried out to smoothen the image. Thereafter, to upgrade the lane features, a second order Gaussian kernel is horizontally convoluted. Afterward, by utilizing the filtered image pixel values the pathway is organized into four parts: unclear path, clear path, non-lane and obstacle path. To determine these paths, a window more modest than the dashed sort of path locale is taken advantage of through the image. In the wake of indicating the parts and also to extricate the path pixels, an adaptive value of threshold is applied to every region of window size in the image. From that point, Steerable filters are used to compute the angles of the candidate lane pixels. Then, to detect the lanes feedback Random Sample Consensus (RANSAC) is performed, which utilizes the angles from the last step and takes the feedback from the last frame which is the curvature of the lanes.

In paper [13], the authors focus on determining dynamic Region of Interest dependent on gathered measurable information of the location of the paths and on vanishing points. From that point on wards, utilizing the grayscale image and the YUV color model, line segments are extricated through ED Lines. Then, noise is removed through the technique of slope filters that are parallel to the left and right paths. A while later, all the line segments that had equal slopes in approximately are combined and then, at that point, the combined line segments are tested with the characteristics of low-high-low intensity qualities of the paths to identify them. Finally, KF is applied for following the inclines and the convergence point of the paths at the base boundary of the image.

In [14], first and foremost converts the captured image gray-scale image model and depending on the road model prediction, dynamic Region of interest(ROI) is determined. Then, at that point, each row of matrix is investigated separately to decide whether the pixels in the specified row are forefront pixels or not. For the specified pixel point, a threshold value has been computed by utilizing measurable information and mean intensity values that are encompassing the region of the pixel. Binarization technique is applied on the image, in the ROI after the computation of threshold value for every pixel. Then, frontal area pixels are inspected by taking morphological properties and chose if they are path markings or not. Finally, places of the Bots Dot type path are followed by KF.

The paper [15] initially extricates features of lane through LSD technique. Then, at that point, direction priority search is applied for disposing of non-path highlights. Direction Priority search just associates line fragments having similar slants in close locale. Line fragments having modest number of pixels are killed after direction priority search. Then, region of interest through vanishing point assessed by the convergence of outstanding line portions. A while later, line fragments out of the ROI is sifted through, too. Finally, single lane pair is distinguished by a score work picking the line fragments converging disappearing point intently in the ROI.

The paper [16] proposes a methodology assessing path positions dependent on CNN network alluded to as DeepLanes utilizing the pictures from down-confronting cameras on the left and right sides of the vehicle rather than forward looking cameras. The network is trained utilizing physically marked and artificially produced pictures utilizing genuine street

pictures with no path markings as foundation. In conclusion, the network yields column list of the path checking which is nearest to the vehicle on the left and right side.

3. TECHNOLOGIES AND TOOLS

This chapter presents software technologies and hardware tools which were embraced to implement the required platform and test the ADAS features.

3.1 Robotic Development Framework

3.1.1 Robotic Operating System

Robotic Operating System(ROS) is an adaptable middle-ware which provides libraries, tools and convention mechanisms that normalizes the implementation of complex projects [17]. William Garage3 in collaboration with more than twenty institutions developed ROS in a Unified model and kept up by Open Source Robotics Foundation (OSRF). So, around the world the active community for ROS is greater than any organization which might be set up. As explained in [18], ROS has three levels of concepts which are: Filesystem Level, Computation Graph Level, and Community Level.

The Filesystem level ideas principally cover ROS assets that you experience on disc such as Packages, Meta-packages, Repositories, Message types, Service types which are as shown in Fig. 3.1. In ROS, for organizing software Packages acts as a main unit. Packages also contain information about ROS runtime Processes also called as nodes, config files, dependent libraries of ROS. Packages are the foremost nuclear construct item and release thing in ROS [19]. Message types define the structure of data for the messages that are being sent in Ros [20]. Service types defines the request and gives response about the structure of data for services in ROS [21].

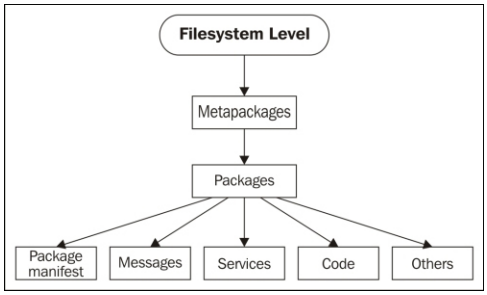

Figure 3.1. FileSystem Level of ROS [18]

The Computation Graph is the distributed organization of ROS measures that are preparing information together. The basic concepts of the ROS Computation Graph level are as below and are shown in Fig. 3.2:

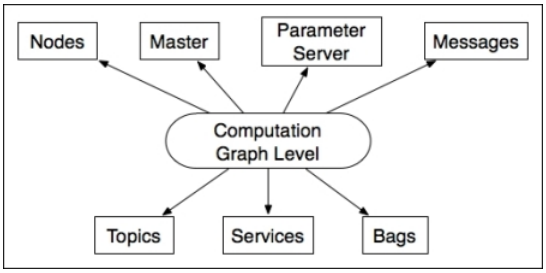

Figure 3.2. Computation Graph level of ROS [18]

- Nodes: Nodes are responsible for performing the process of computation. Nodes are consolidated into a graph and speak with each other utilizing topics, services, and Parameter Server. These nodes are implied to function at a fine-grained scale [22].
- Master: The ROS Master is responsible for providing names and registration services to all other remaining nodes in ROS framework. It chases down the Publisher and the Subscriber to remaining services and topics. The primary role ROS Master is to engage every ROS node to discover each other. At the point when the nodes have discovered one another, they talk with each other distributed [23].
- Parameter Server: A Parameter server is a common, multi-variate word reference that is available through network APIs. Nodes utilize this server to store and recover configuration parameters during run-time [24].
- Messages: A message is a basic information structure, involving composed fields. Nodes uses Messages to communicate with one another [25].
- Topics: Messages are steered by means of a transport system with the help of Publish/Subscribe semantics. Nodes convey the message by distributing to the required topic. Topics are named transports over which Nodes trade Messages. A node that is inquisitive about a certain kind of information will subscribe to the suitable Topic.

Generally, Publishers and Subscribers are not mindful of each other's presence. The thought is to decouple the generation of data from its utilization [26].

- Services: The publish/subscribe model is a truly adaptable correspondence worldview, however its many-to-many, single direction transport isn't suitable for request/reply cooperation's, which are frequently needed in a peer to peer framework. Request/reply is carried with the help of services, which are characterized by a couple of message structures: one is for request and the other is being for reply. Providing node provides service on one name and client utilizes the service by sending the request message and anticipating the answer [27].

- Bags: Bags are a configuration for saving and playing back ROS message information. Bags are a significant system for storing information, like sensor information, that can be hard to gather yet is vital for creating and testing algorithms [28].

The ROS Master acts as a central node in the entire ROS framework. It stores all the information related to Topics and Service registration data for the ROS Nodes. Nodes communicate with Master to give their registration data and to get information about other nodes. Ros uses the most common protocol called TCPROS, which makes use of TCP/IP sockets.

The Community level, which is the last level in ROS, shown in Fig. 3.3 is capable for conveying and giving resources, providing information and software with specific systems. It likewise gives dissemination, discussions and furthermore responsive sites for replying dedicated questions of ROS. In my thesis work, I have utilized ROS Melodic distribution.

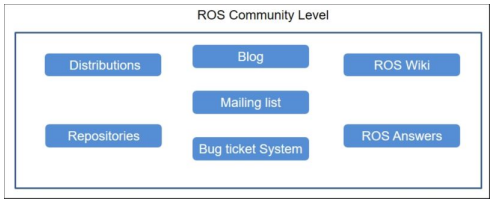

Figure 3.3. Community Level of ROS [18]

3.1.2 RVIZ

ROS Visualization, Rviz, is an incredible 2D/3D representation device from ROS. Rviz gives different components to the user to picture a robot model while following its development, noticed sensor data and numerous different perspectives that are observed to be pertinent for the project being developed. By giving this information that can be shown from real minute or through logged data, utilizing ROS Bags, Rviz states as incredible instrument for troubleshooting a robot application, since it permits the client to show just the data that the client requires [29].

With respect to sensor visual representation, Rviz permits to show 3D sensor information from stereo cameras, lasers, energy and other 3D gadgets information from point clouds or depth images. Similarly, 2D sensors information can be obtained from webcams or RGB cameras and laser range locators. Rviz running machine communicating with ROS working, will show the robots current arrangement on the virtual robot demonstrate. This model communicates with sensors, through TF changes that will get the information from the sensors and applying to virtual demonstrate. Fig. 3.4 portrays an illustration of a rivz window, which thusly gives the current design of a virtual bot.

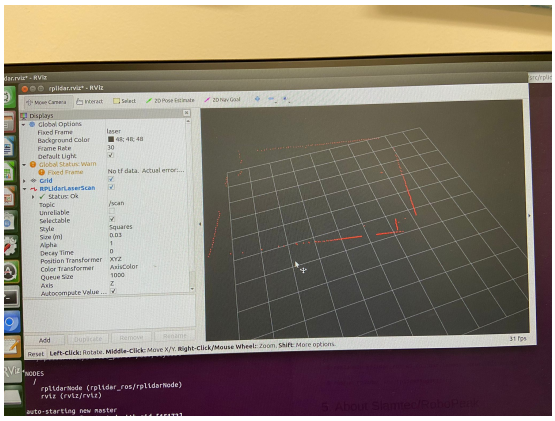

Figure 3.4. Rviz window

3.2 Embedded Computing Platforms

3.2.1 NVIDIA Jetson Nano

Nvidia Jetson Nano Developer Kit was the on-board computer that was utilized in this project which is displayed in Fig. 3.5. This was being used because of its computational power and its compactness and is also ideal for the improvement of software utilizing Linux Operative System (OS), and has standard connectors that give an adaptable and expandable interfaces with different modules.

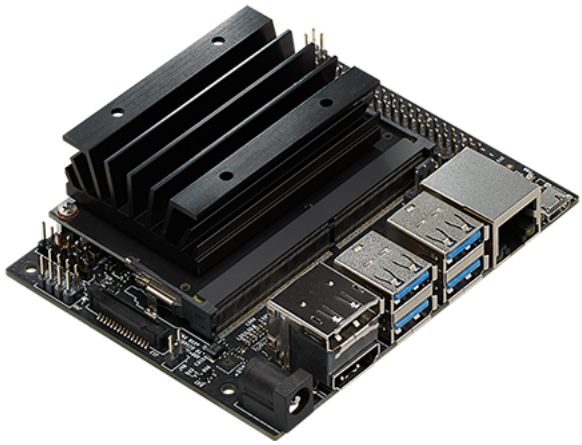

Figure 3.5. Nvidia Jetson Nano

The development carrier board provides an assortment of peripherals which includes [30]:

- Gigabit Ethernet M.2 Key E connector
- USB: 4 x USB 3.0 , USB 2.0 Micro-B
- Storage: microSD card slot for main storage
- Display port connector and an HDMI output slot;
- Expansion header which includes 40-pin headers with I2C, GPIO, SPI, UART, I2S
- DC barrel jack for 5 V Power Input

The fundamental target of the board is to handle all the essential data of the surroundings, that are received from sensors associated with its peripherals and control upon the vehicle.

3.3 Sensors

To get data approximately of the surroundings and the positioning of platform, it is important to utilize sensors which gives us the necessary data with the foremost accuracy possible. The ones that are used over here are RPLiDAR A1[31] and Logitech C270 Desktop Webcam - USB Camera [32].

3.3.1 LiDAR

Lidar, abbreviated as Light Detection and Ranging, not just provides likelihood to scan 360 degrees of the environment from the placed position, contingent upon sort of sensor utilized, yet additionally to get distance of a specific obstacle. Lidar is also great for robot applications that is necessary for Localization and detection of obstacle. Its functionality is equivalent to a simple range finder, a light is transmitted and at whatever point it reflects on a surface and is gotten by the sensor, it computes the distance taking in thought the time it has elapsed since it was transmitted. Fig. 3.6 presents RPLidar A1 utilized in this work.

Figure 3.6. RPLIDAR A1

RPLIDAR A1 relies upon laser triangulation expanding standard and uses quick vision getting and taking care of hardware. The rate at which it estimates the range of distance is in excess of 8000 times each second. It runs clockwise to play out a 360-degree omni

directional laser range filtering of its general environment around it and afterward create a layout map for the same. We can change the rate of scan values from 2 - 10 Hz and it is perfect device for Localization and Navigation of robot. Below are the specifications of RPLIDAR:

- Measuring Range: 0.15m - 12m
- Sampling Frequency: 8K
- Rotational Speed: 5.5Hz
- Angular Resolution: $\leq 1^{\circ}$
- Dimensions: 96.8 $\times$ 70.3 $\times$ 55mm
- System Voltage: 5 volt
- System Current: 100 milliampere
- Power Consumption: 0.5W
- Output UART Serial (3.3 voltage level)

3.3.2 Camera

Cameras basically acts as Human Visual System. The camera can see and perceive the tones and textures. These benefits of the camera permit the autonomous vehicle to recognize distinctive Traffic signs, path markings, vehicles, walkers, and so on. Fig. 3.7 presents the Logitech C270 Desktop Webcam- USB Camera used in this thesis work.

Figure 3.7. Logitech C270 Webcam

This USB camera has a resolution of 30 fps and the other technical specifications are as below [32]:

- Max Resolution: 720p/30fps
- Camera mega pixel: 0.9
- Focus type: fixed focus
- Lens type: plastic
- Built-in mic: Mono
- Mic range: Up to 3 ft (1 m)
- Diagonal field of view (dFoV): 60°

3.4 Vehicle Platform

To create a physical test platform, a vehicle model was important to be modified and adjusted to have every one of the parts put and coordinated. As stated earlier, this test platform is developed at LHP Engineering Solutions. The vehicle model is a TRAXXAS Slash 4 X 4 Platinum truck, a one -tenth size of a vehicle, and is as shown in below Fig. 3.9. The flexibility of the RC model, permits it to be changed and be based upon it, making an all-around organized stage to test various situations. Other than the parts that come mounted on the vehicle, a Lipo battery is important to control up the vehicle, in here 4000 maH 7.4V Traxxas Lipo battery was utilized [33].

Figure 3.8. Vehicle platform

3.4.1 Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface (PCA9685)

A simple and basic strategy to control Servo Motors is through I2C Communication protocol, which is achieved by connecting the Jetson Nano and PWM driver on the NVIDIA Jetson Nano Developer Kit. I have utilized Adafruit 16-Channel 12-bit PWM/Servo Driver in the Race vehicle. The best benefit of this micro controller is that, just by using two

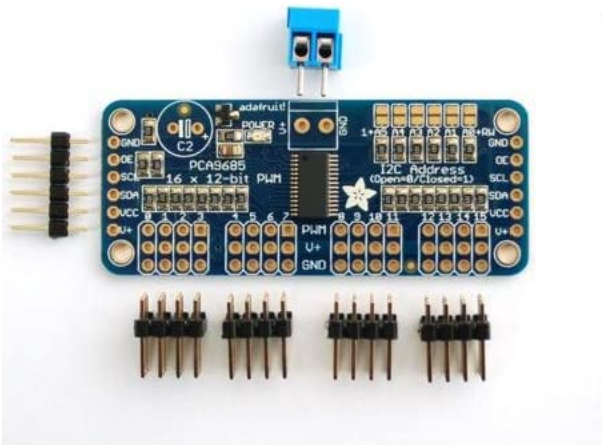

Figure 3.9. Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface

pins, we can handle 16 free-running PWM outputs. The PCA9685 breakout board is a fair application to unequivocal the scaled down scale controller for servo control [34].

3.4.2 Buck Converter

A buck converter is utilized to change over the voltage from LIPO battery (11.1v) to Jetson Nano(5v). It is a stage down DC-to-DC power converter which steps down voltage from its feedback (supply) to its yield (load). SING F LTD Buck Converter DC to DC 5A 75W Input 4-36V Output 1 25-36V Step Down Regulator Power Module is the converter that we utilized on our RACECAR [35].

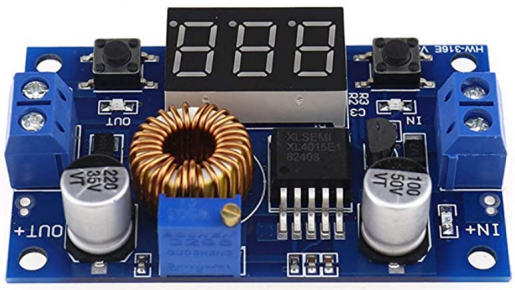

Figure 3.10. Buck Converter

3.4.3 LIPO batteries

LiPo batteries give the punch and on-request ability to arrive at the maximum velocities that Traxxas models are worked to convey. LiPo batteries have more conspicuous essential thickness compared to other batteries, which implies they provide more voltage and breaking point than other batteries would of a comparable volume. Below are the specifications used [36].

Table 3.1. LIPO Battery Specifications

Type	Total Capacity	Voltage	C Rating	Length	Height	Width
LIPO	6400mAh	11.1V	25	135mm	42mm	45mm

Figure 3.11. LIPO Battery

To bolster each component within the vehicle, changes were important to be made to the design. Because of it, a couple of transformations were made, such as altering the stature of NVIDIA Development board to fit an alternate battery, as well as penetrating new openings in it, to integrate it to the new chassis made. Also the wiring for the vehicle was revamped to fit the requirements of this vehicle.

4. SYSTEM ARCHITECTURE

This chapter depicts the framework of overall system, describing how every part is associated with one another and the advantage of carrying out this automated framework.

4.1 Hardware

This autonomous race car takes upon the below addressed hardware architecture as shown in Fig. 4.1. All the components on the cart are connected to Nvidia Jetson Nano, which acts as an on-board computer and does all vital calculations. The components are connected to Jetson Nano using communication protocols such as USB and I2C. Expanding it further, Camera and Lidar are connected through USB and PWM driver module is connected using I2C communication protocol. To control each electrical part of the automated vehicle, 12V DC power supply was essential and for that LIPO batteries were utilized.

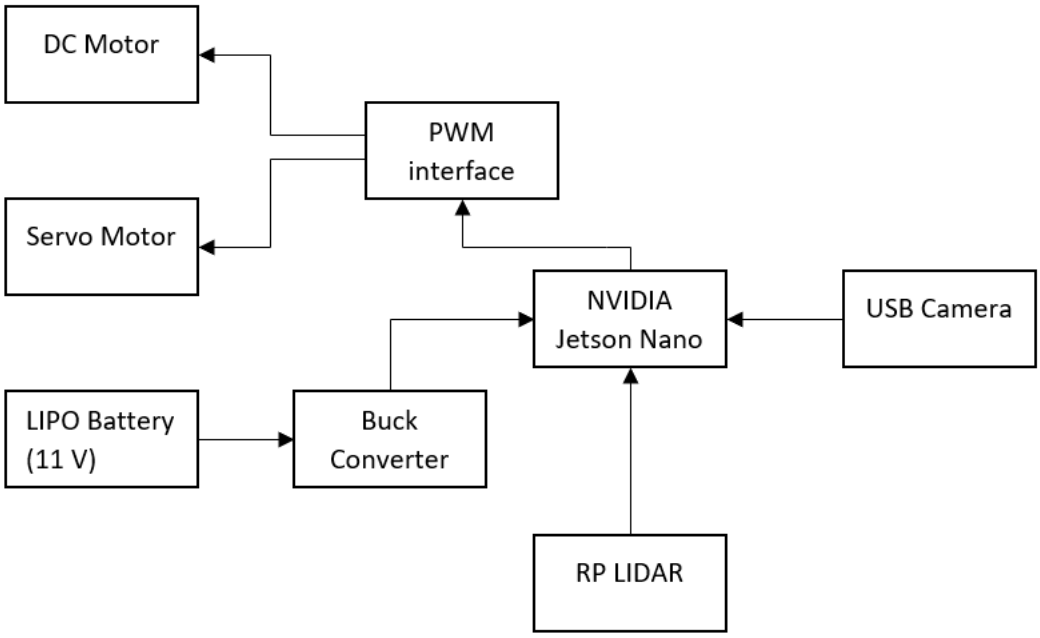

Figure 4.1. Hardware System Architecture

4.2 Software

Software architecture can be explained as shown in below Fig. 4.2

An operating system is required for NVIDIA jetson Nano, which is an on-board computer. The most commonly used operating system for automated applications is Linux Ubuntu. Hence, Linux Ubuntu 16.04 has been used as operating system on Jetson Nano to implement the race car application. In this PC, the Robot Operative System of Melodic Distribution has been installed which in turn provides bundles of packages, workspace, to build and develop the required framework. Different packages which are ready to utilize software are being provided by Robotic Operating System (ROS) by taking the dependency on configuration of parameters. In here, packages that were being used are LIDAR and USBCAM.

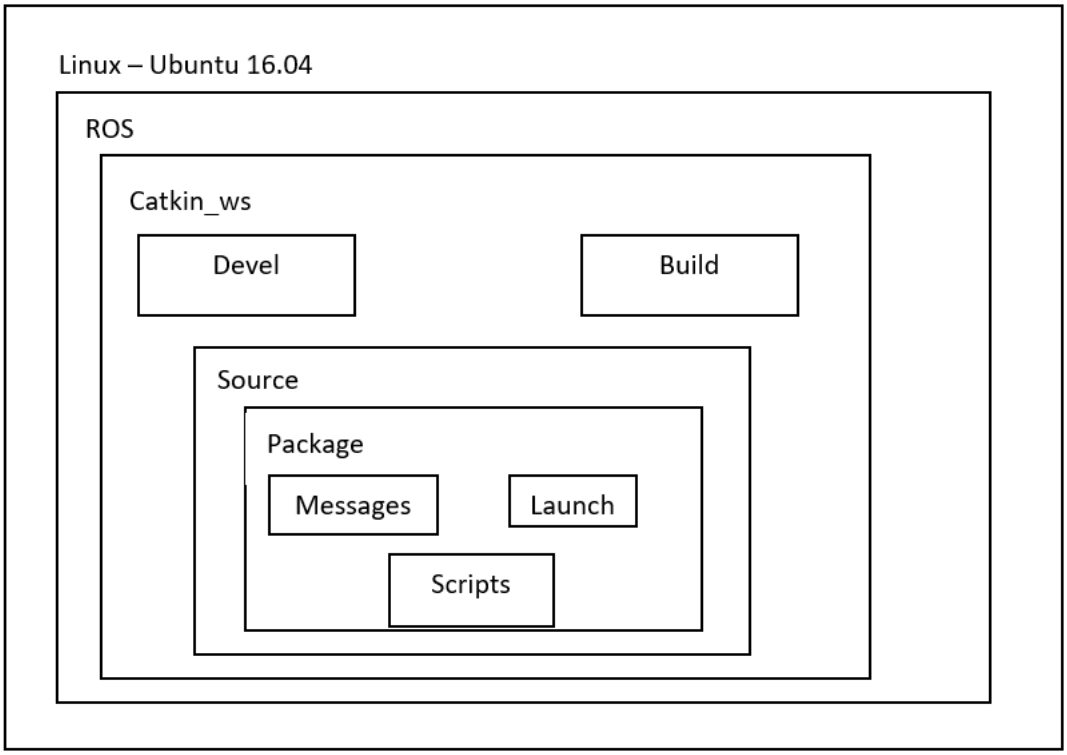

Figure 4.2. Software System Architecture

Catkin workspaces comprises of three directories which are Build, Devel and Source, and all the written scripts, Message folder and Launch folder will be saved in the last directory which is Source which can be seen in the below Fig. 4.3. Along with written scripts it also incorporates the Message folder, which consists of the message types that went through topics. Furthermore, launch folder permits to make custom launch which are responsible for initiation of multiple nodes, Configuration files and other files simultaneously.

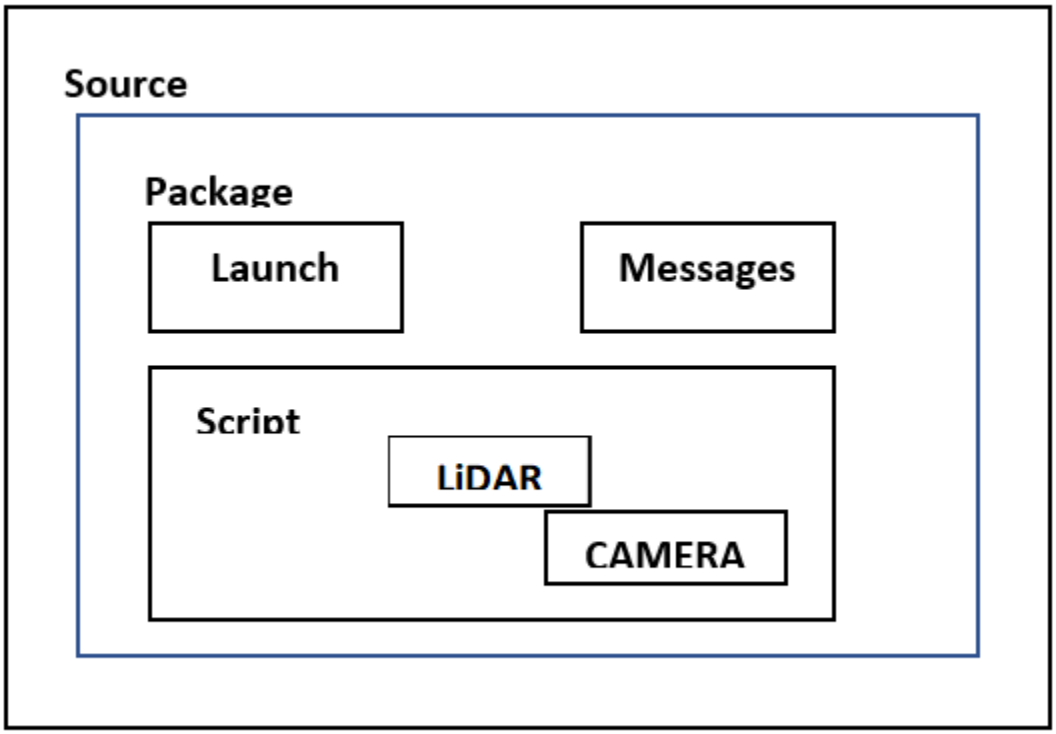

Figure 4.3. Source Directory of Catkin Workspace

4.3 PWM Interface

A clear technique to control DC and Servo motor of autonomous race car which is using Jetson Nano is, to connect them using I2C communication protocol to PWM Driver, as there is no possible means of direct association between them. In this way, the PWM Driver module, takes the control of steering angle and furthermore, accepts the input commands of

throttle values from the Jetson Nano and controls the motors of the race vehicle based on the provided input throttle and steering values.

Using the PWM driver, an expansion can be set up among Jetson Nano and the electronic speed controller. The PCA9685 uses Pulse Width Modulation technique, which sends digital pulses to servo and DC motors, dependent upon the information it got from the Jetson Nano. Here, the information is sent utilizing I2C communication from Jetson Nano to PWM module PCA9685.

The library adafruit servokit must be installed, which consists of python commands as mentioned below in the equation form, to convert the input commands of steering and throttle, to its relating PWM signals. The throttle values are the integer values, where the 0 addresses the halting value of the vehicle, value 1 represents the forward movement of vehicle and -1 represents the backward movement of the vehicle. The steering values going from 43 degrees to 140 degrees of which 43degrees being extreme left angle and 140 degrees being extreme right angle, 43 - 90 degrees indicate left turn, and 92 – 140 degrees indicate right, and 91 degrees is going straight. Below are the python commands for throttle and steering angle [37].

```
import time
from adafruit_servokit import ServoKit
kit = ServoKit(channels=16)
kit.continuous_servo[0].throttle = 1
kit.continuous_servo[0].throttle = -1
kit.continuous_servo[0].throttle = 0
kit.servo[1].angle = 43
kit.servo[1].angle = 140
```

Here, the numbers 0 and 1 mentioned inside the square brackets in the above command indicates the pin numbers of PWM driver.

4.4 Getting started with Jetson Nano kit

To prepare for the setup, the items that are required for getting started are microSD card and micro-USB power supply.

- Jetson Nano development kit utilizes microSD card for the purpose of booting and also as the main storage device(32GB is used here).
- To power the Nano, a micro-USB power supply of 5V and 2.5A is required. I have utilised Adafruit's 5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable (GEO151UB-6025).
- Ethernet cable.

4.4.1 Write image to microSD card

To prepare the SD card, a computer with good internet connection and should have the capacity to read and write SD cards. Download the Jetson Nano Developer Kit SD Card Image from NVIDIA'S developer website [38]. For the windows operating system, I have followed all the instructions step-wise mentioned in the NVIDIA website[38] for flashing the image to the microSD card.

4.4.2 Initial setup and Boot

After flashing the microSD card, insert the SD card into the slot on the underside of the Jetson Nano Developer Kit. Then, power on the computer and connect USB keyboard, mouse and micro USB power supply. Jetson Nano Developer Kit will be powered and boots automatically.

After the Jetson Nano Developer kit powers on, a green colour LED which is present next to the micro-USB connector is will light and after booting for the first time the screen looks as shown in the below Fig. 4.4.

Figure 4.4. Output screen of NANO after the Initial Boot

5. LIDAR AND CAMERA INTERFACE

This chapter depicts how the data can be captured by LIDAR and camera. Further it is expanded to explain how the gathered information can be used to implement the ADAS features such as emergency braking (on detecting an object) feature, lane keep assist feature and lane changing feature (on detecting an object while traveling in two lane roads). The below drawn flowchart can be used to explain the algorithm used to build the code:

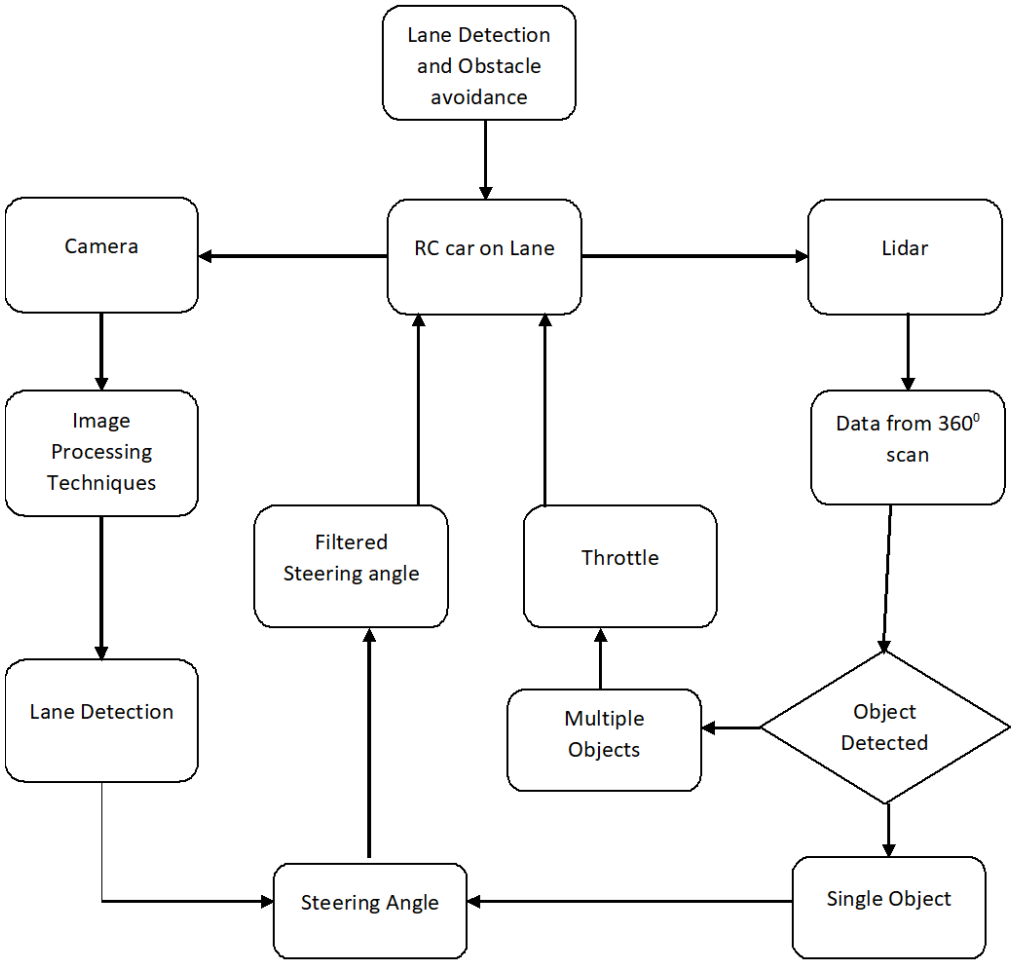

Figure 5.1. Flowchart for LIDAR and Camera interface

Along with other hardware, race car is integrated with LIDAR and camera sensors. The Camera is used to capture images. Once the images are captured, camera used image processing techniques to detect the lanes. After the detection of lanes, using mathematic formulas, I have calculated the slope and the angle at which it is deviated from midpoint and calculated the steering angle. PI controller is then used to get the filtered steering angle. LIDAR scans from 0 to 360 degrees to detect the objects. Based on the number of objects on the lanes, the cart will decide to stop or to change to the other lane.

5.1 LIDAR

5.1.1 Implementation

LIDAR has a scan capability of 0 to 360 degrees. It rotates in clockwise direction while scanning all the angles from 0 to 360 degrees and it rotates in anti clockwise direction to give the data. For mounting the LIDAR with the right configuration, LIDAR's front side which is its 0 degrees, should be determined by placing an obstacle in front of the LIDAR. By validating the data published by the LIDAR, we can determine whether the obstacle was detected or not by the sensor. The data published by the LIDAR should contain the value "inf" as shown in the Fig. 5.2 when an obstacle is detected.

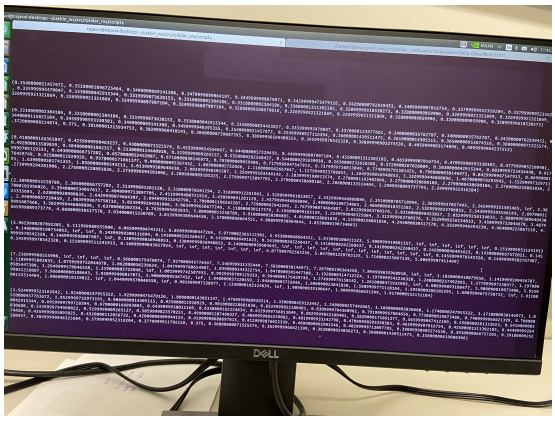

Figure 5.2. Data published by LIDAR

By following the above method, front of the Lidar has been determined and is mounted accordingly on the racecar as shown in the Fig. 5.3. All the required Ros drivers and packages related to RPLIDAR, I have referred to [39] GitHub repository.

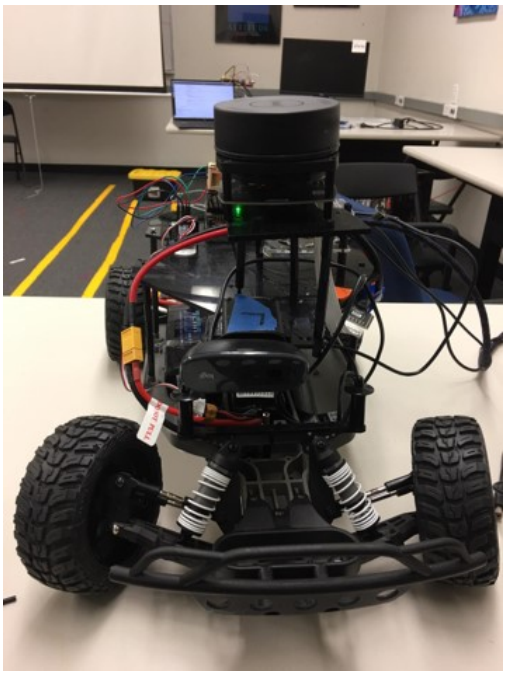

Figure 5.3. LIDAR mounted on Race car

The RP LiDAR A1 publishes the data in ROS environment as a sensor_msgs/LaserScan.msg topic, which contains information such as range, intensities, minimum angle, maximum angle etc. as displayed in Fig. 5.4. By subscribing to LaserScan topic [40], we can get the distance for the specified range of angles and the algorithm is as shown in below Fig. 5.5.

Figure 5.4. LaserScan topic [40]

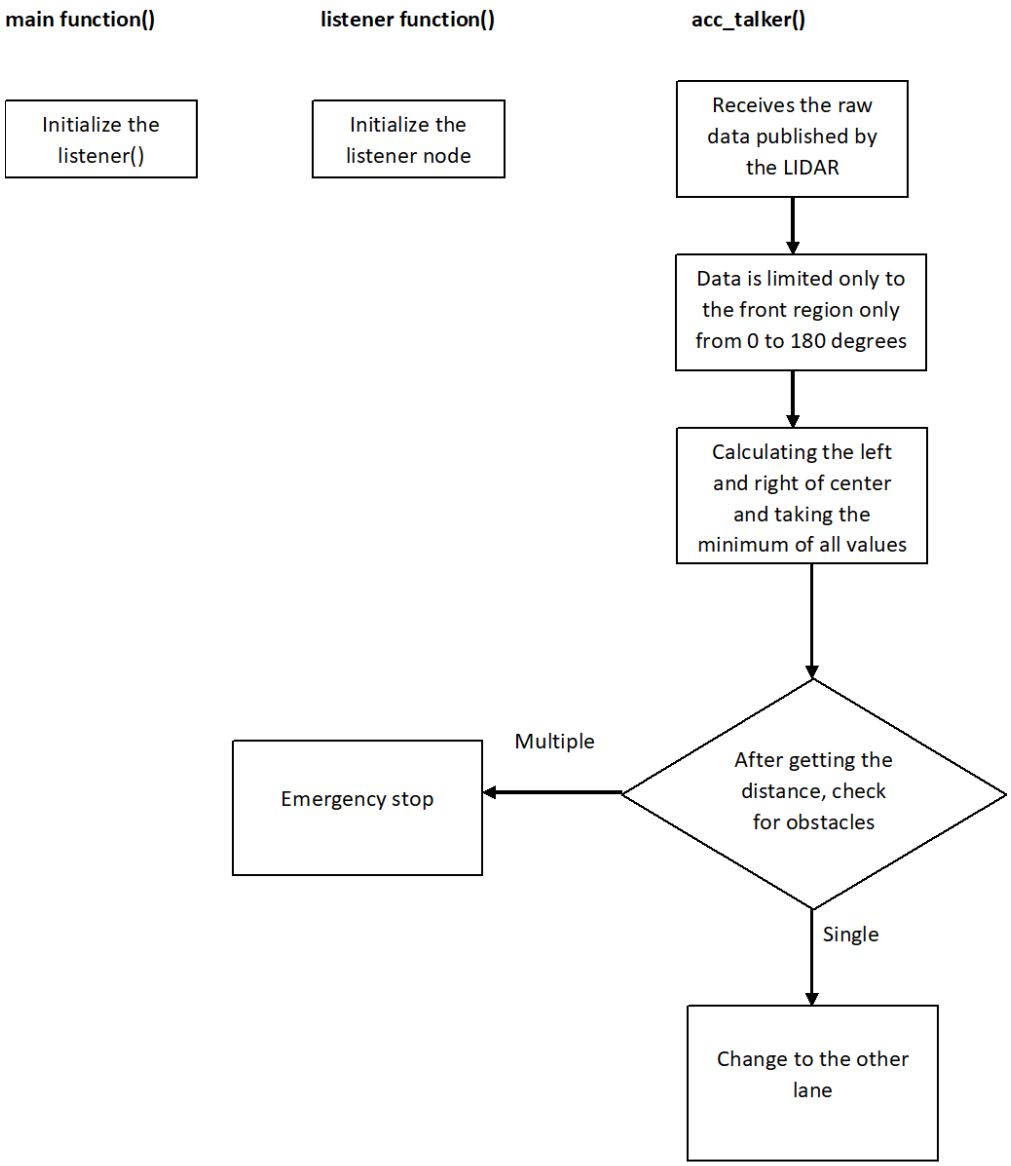

Figure 5.5. LIDAR subscriber Algorithm

5.2 Camera

The Camera integrated on the racecar is used for Lane Detection and have used USB C270 Logitech camera. Computer vision and image processing techniques have been utilized for developing an algorithm for lane detection and lane keeping. On ROS platform, OpenCV library along with python interface have been utilized. All the required packages for open cv library are downloaded from GitHub repository [41]. The flowchart algorithm for camera is as shown in Fig. 5.6.

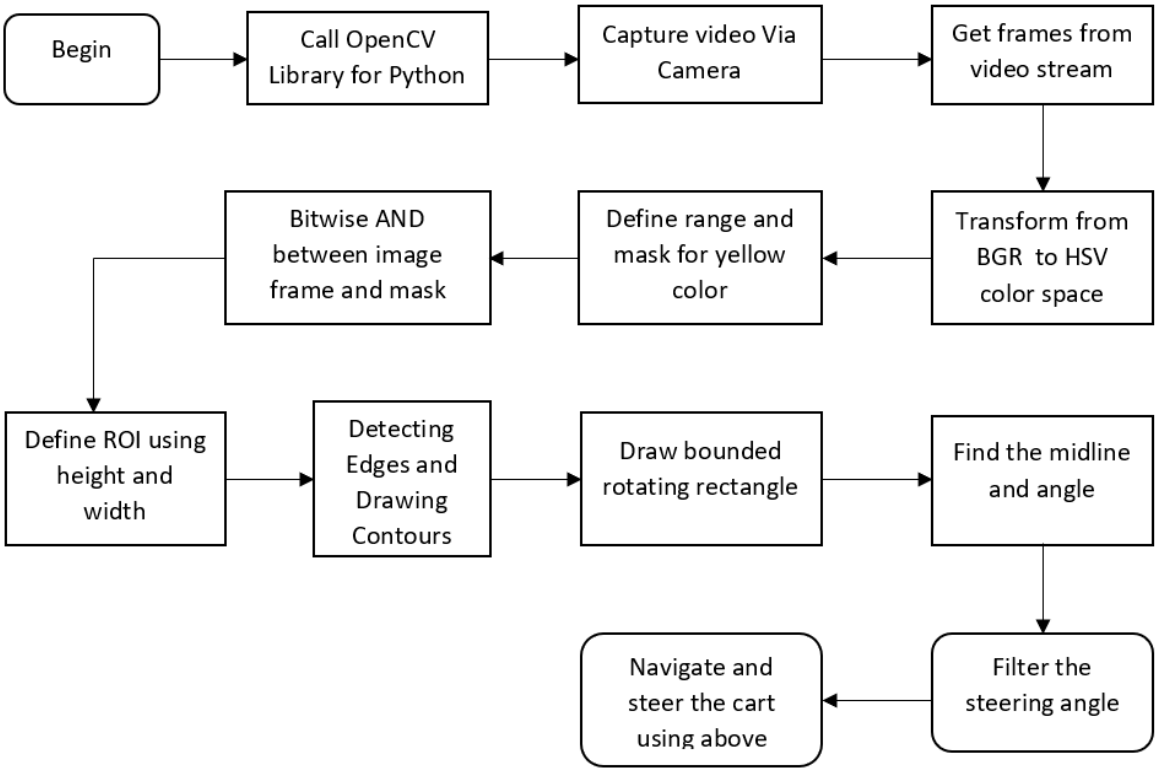

Figure 5.6. Flowchart of camera

Initially we need to import all the required libraries such as open cv and rospy for the program to execute in python environment.

5.2.1 Capturing Video

Using the VideoCapture() function in Open CV, the live video will be captured. Fig. 5.7 shows the captured RGB yellow lane image.

Figure 5.7. RGB Yellow Lane Image

5.2.2 Get frames from video stream

Once the video has been captured, frames are decoded from the video to perform all the required image processing techniques.

5.2.3 BGR to HSV color space

As the captured frames are in BGR color space , they should be converted to HSV color space. The principle explanation for converting BGR color space to HSV color space is, in BGR color space, the object color might be illuminated with the different light conditions and can appear as darker or lighter version of it. In HSV color space, as the hue represents the color, this color space will convey the object color in one tone regardless of its overshadowing. Fig. 5.8 shows the image converted from BGR color space to HSV color space.

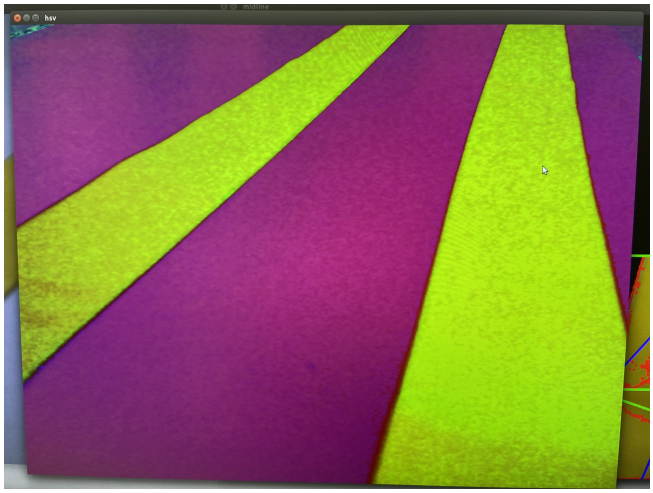

Figure 5.8. HSV Image

5.2.4 Define the range and Mask for Yellow Color

After converting to HSV color space, upper and lower boundaries for the lane color are specified, so that the lane color is highlighted. After determining the range, masking is done so that the defined lane color is lifted in white color and the remaining is in black color. Fig. 5.9 shows the masked image.

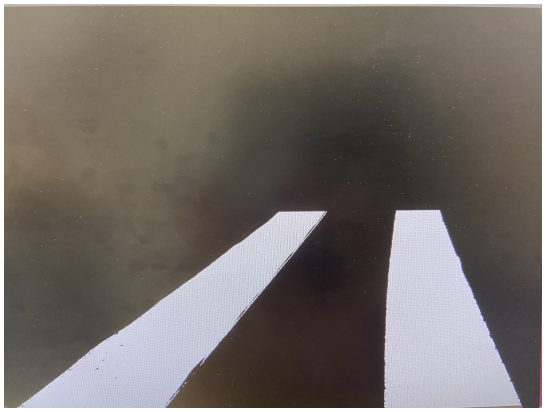

Figure 5.9. Masked Image

5.2.5 Bitwise AND between image frame and mask

After masking operation is done, stitching is done between original image and masked image using Bitwise AND operator. Fig. 5.10 shows the stitched image of original image and masked image using Bitwise AND operator.

Figure 5.10. Bitwise AND between image frame and mask

5.2.6 Define ROI using height and width

The captured image dimensions (height and width) can be obtained using the shape() function. While camera is detecting the lanes, we just focus on the closest path and the rest of the path can be overlooked. So, the top part of the screen is cropped out by concentrating only on the bottom part of the screen. Fig. 5.11 shows the isolated image with the bottom part of screen and removing the top part of the screen.

Figure 5.11. ROI Image

5.2.7 Detecting Edges and Drawing Contours

On the masked image, edges are detected using canny edge detector. Open CV library provides the function directly. Contours are drawn by joining all the points on the boundary of an image. We can draw the contours using the open CV function drawContours(). Fig. 5.12 shows the contours drawn on the image.

5.2.8 Draw bounded rotating rectangle

For all the contours drawn, using the function minAreaRect() we can get the bounded rotating rectangle. Using the function boxpoints() we can get the four vertices (x1,y1), (x1,y2), (x2,y1) and (x2,y2). Fig. 5.13 shows the bounded rotating rectangle drawn on the lane.

Figure 5.12. Contours on Image

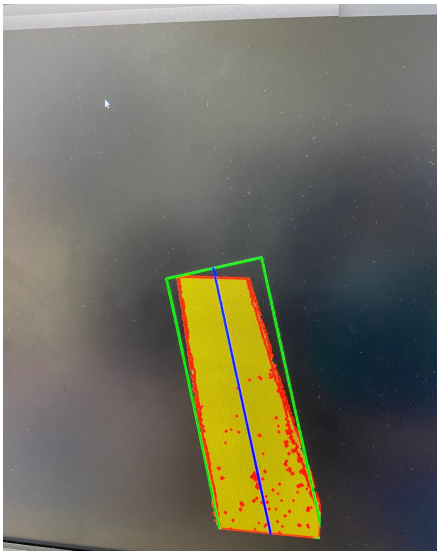

Figure 5.13. Bounded Rotating Rectangle

5.2.9 Find the midline and angle

As we are having the four co-ordinate points, we can get the midpoint for top and bottom of the rectangle, and then we can draw a midline between the points using the function line(). Based on the obtained points, we can find the angle by using the below formula Theta =Tan(a/b)

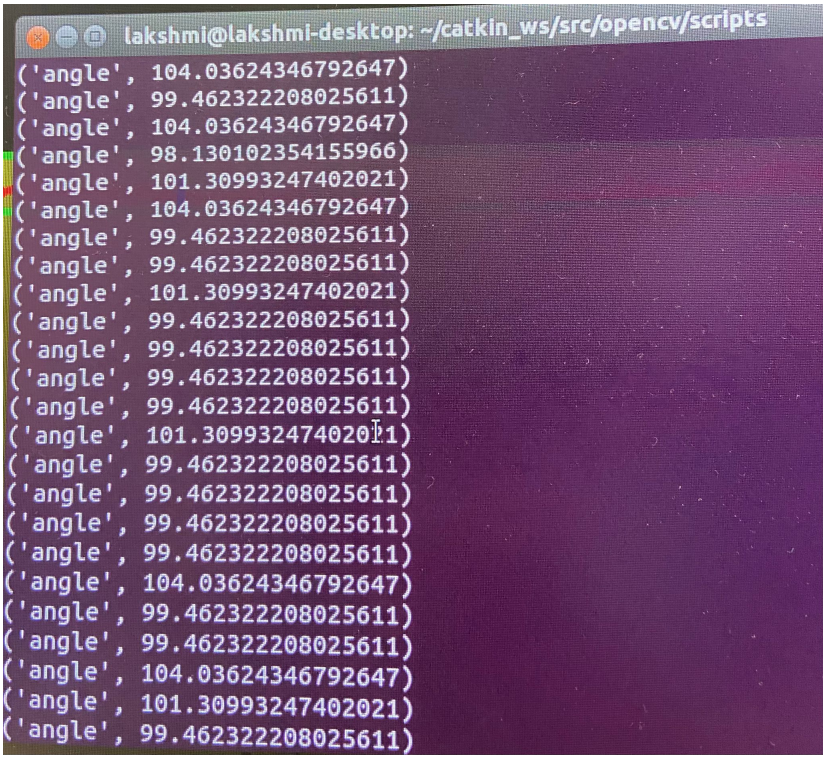

Figure 5.14. Recorded angles

5.2.10 Filter the steering angle

PI controller has been utilized to tune the steering values based on the steering values that are calculated in the above step.

5.2.11 Navigate and steer the cart

Based on the steering angle and the lane detection from the above steps, the cart will be navigated along the lane with the heading directions as shown in Fig. 5.14.

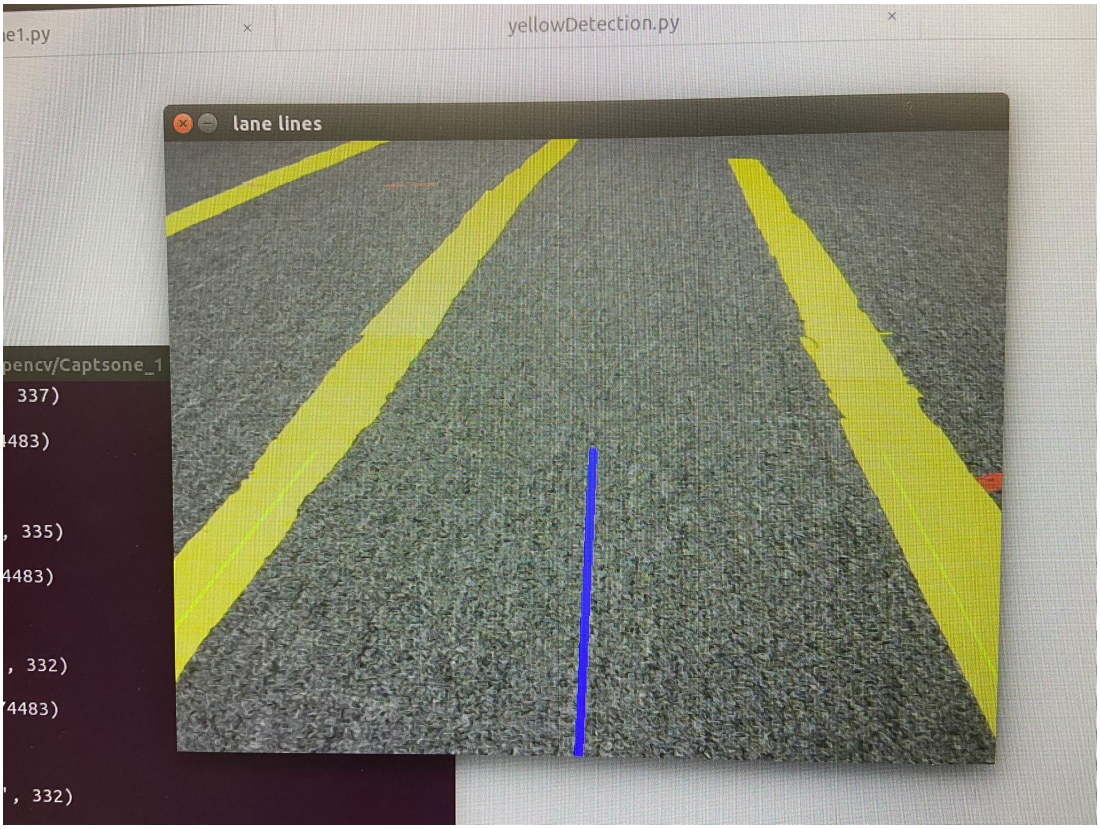

Figure 5.15. Image with Heading direction

5.3 Object Detection

After LIDAR detects the objects, based on the lanes ADAS features such as Emergency Brake and Lane change are implemented.

5.3.1 Multiple Obstacles

If the autonomous race car is going on a double lane and has objects on both the lanes, as soon as the LIDAR detects the objects the race car will be stopped featuring the EMERGENCY BRAKING feature of ADAS system.

Figure 5.16. Lane with multiple obstacles

5.3.2 Single Obstacle

If the autonomous race car is on the double lane and has only one obstacle on one of the lanes, then after the object detection, the race car goes to the other lane featuring LANE CHANGE feature of ADAS system.

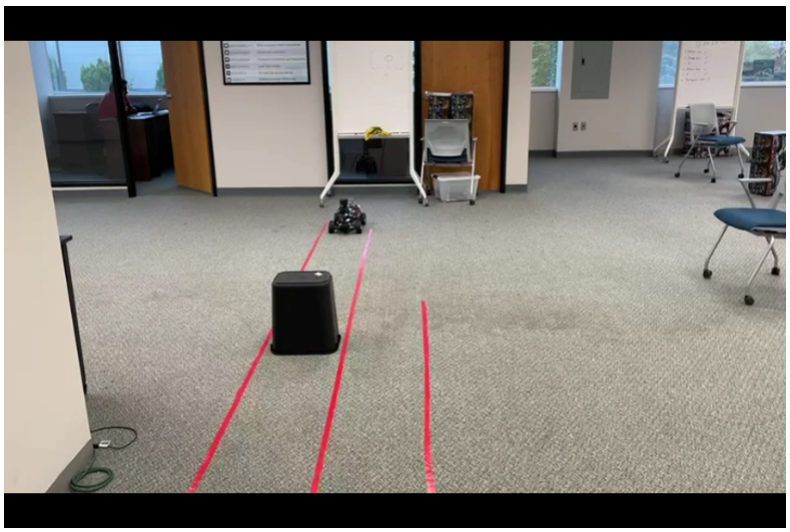

Figure 5.17. Lane with single obstacle

6. CONCLUSION AND FUTURE WORK

This thesis shows the implementation of ADAS features such as Emergency Brake system, Lane Keeping and Lane changing using Robot Operative System (ROS) environment. With the race car assembled, by utilizing the ROS along with other packages and tools, algorithms were developed to achieve the implementation of the mentioned ADAS features. Relying on the USB camera and LIDAR, obstacle distance and identification of lanes has been achieved. Upon interfacing the LIDAR and camera data the ADAS features have been successfully implemented on the race car which has been tested in an inside environment with constant slow speed.

In future, besides testing in an indoor environment with constant speed, outside environment with variable high speed and sensor fusion can be considered for better processing time as an optimistic approach. USB camera can be upgraded to stereo camera to obtain better processing times. Conversion of 2D lidar data and sensor fusing it with stereo camera data can be considered for future development and implementation.

REFERENCES

[1] D. d. W. K. A. Brookhuis and W. H. Janssen, "Behavioural impacts of advanced driver assistance systems–an overview," *European Journal of Transport and Infrastructure Research*, vol. 1, 3 2001. DOI: https://doi.org/10.18757/ejtir.2001.1.3.3667.

[2] Y. Dong, Z. Hu, K. Uchimura, and N. Murayama, "Driver inattention monitoring system for intelligent vehicles: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 596–614, 2011. DOI: 10.1109/TITS.2010.2092770.

[3] T. Pietrasik. "World health organization." (2021), [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[4] K. T. P. Czech and J. Barcik, "Autonomous vehicles: Basic issues.," *Scientific Journal of Silesian University of Technology. Series Transport*, vol. 100, 2018. DOI: https://doi.org/10.20858/sjsutst.2018.100.2.

[5] B. O. Oviedo-Trespalacios O Tichon J, "Is a flick-through enough? a content analysis of advanced driver assistance systems (adas) user manuals," *PLOS ONE*, vol. 16, 2021. DOI: https://doi.org/10.1371/journal.pone.0252688.

[6] WARRENDALE, "SAE international releases updated visual chart for its "levels of driving automation" standard for self-driving vehicles," 2018. [Online]. Available: https://www.sae.org/blog/sae-j3016-update, Last Accessed: 08/15/21.

[7] S. D. Pendleton, H. Andersen, X. Du, *et al.*, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017, ISSN: 2075-1702. DOI: 10.3390/machines5010006. [Online]. Available: https://www.mdpi.com/2075-1702/5/1/6.

[8] T. W. of Sam Huang, "How the autonomous car works: A technology overview," 2018. [Online]. Available: https://thewordofsam.medium.com/how-the-autonomous-car-works-a-technology-overview-5c1ac468606f.

[9] J. Betz, A. Wischnewski, A. Heilmeier, *et al.*, "A software architecture for an autonomous racecar," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–6. DOI: 10.1109/VTCSpring.2019.8746367.

[10] D.-K. Lee, J.-S. Shin, J.-H. Jung, S.-J. Park, S.-J. Oh, and I. S. Lee, "Real-time lane detection and tracking system using simple filter and kalman filter," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, pp. 275–277. DOI: 10.1109/ICUFN.2017.7993792.

[11] J.-G. Kim, J.-H. Yoo, and J.-C. Koo, "Road and lane detection using stereo camera," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018, pp. 649–652. DOI: 10.1109/BigComp.2018.00117.

[12] Y. Son, E. Lee, and D. Kum, "Robust multi-lane detection and tracking using adaptive threshold and lane classification," *Machine Vision and Applications*, vol. 30, Feb. 2019. DOI: 10.1007/s00138-018-0977-0.

[13] C. Lee and J.-H. Moon, "Robust lane detection and tracking for real-time applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 4043–4048, 2018. DOI: 10.1109/TITS.2018.2791572.

[14] S. Gehrig, A. Gern, S. Heinrich, and B. Woltermann, "Lane recognition on poorly structured roads-the bots dot problem in california," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, 2002, pp. 67–71. DOI: 10.1109/ITSC.2002.1041190.

[15] T. Youjin, C. Wei, L. Xingguang, and C. Lei, "A robust lane detection method based on vanishing point estimation," *ScienceDirect*, vol. 131, pp. 354–360, 2018. DOI: https://doi.org/10.1016/j.procs.2018.04.174.

[16] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "Deeplanes: End-to-end lane position estimation using deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 38–45. DOI: 10.1109/CVPRW.2016.12.

[17] M. QUIGLEY, "Ros : An open-source robot operating system," *ICRA 2009 WOrkshop on Open Source Software*, 2009. [Online]. Available: https://ci.nii.ac.jp/naid/20001641901/en/.

[18] *Ros/concepts - ros wiki*. [Online]. Available: http://wiki.ros.org/ROS/Concepts, Last Accessed: 08/15/21.

[19] *Ros/concepts - ros packages*. [Online]. Available: http://wiki.ros.org/Packages, Last Accessed: 08/15/21.

[20] *Ros/concepts - ros msg*. [Online]. Available: http://wiki.ros.org/msg, Last Accessed: 08/15/21.

[21] *Ros/concepts - ros services*. [Online]. Available: http://wiki.ros.org/srv, Last Accessed: 08/15/21.

[22] *Ros/concepts - ros nodes*. [Online]. Available: http://wiki.ros.org/Nodes, Last Accessed: 08/15/21.

[23] *Ros/concepts - ros master*. [Online]. Available: http://wiki.ros.org/Master, Last Accessed: 08/15/21.

[24] *Ros/concepts - ros parameter server*. [Online]. Available: http://wiki.ros.org/Parameter20Server, Last Accessed: 08/15/21.

[25] *Ros/concepts - ros messages*. [Online]. Available: http://wiki.ros.org/Messages, Last Accessed: 08/15/21.

[26] *Ros/concepts - ros topics*. [Online]. Available: .%20http://wiki.ros.org/Topics, Last Accessed: 08/15/21.

[27] *Ros/concepts - ros services*. [Online]. Available: http://wiki.ros.org/Services, Last Accessed: 08/15/21.

[28] *Ros/concepts - ros bags*. [Online]. Available: http://wiki.ros.org/Bags, Last Accessed: 08/15/21.

[29] *Ros/concepts - ros rviz*. [Online]. Available: https://wiki.ros.org/rviz, Last Accessed: 08/15/21.

[30] N. DEVELOPER, *Jetson nano developer kit*. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit, Last Accessed: 08/15/21.

[31] S. Slamtec, *Rplidar a1*, 2016 - 2021. [Online]. Available: https://www.slamtec.com/en/Lidar/A1.

[32] *Logitech c270 webcam*. [Online]. Available: https://www.logitech.com/en-us/product/hd-webcam-c270#specification-tabular.

[33] Kanglow, *Racecar/j platform preparation*. [Online]. Available: http://www.jetsonhacks.com, Last Accessed: 08/16/21.

[34] *16-channel 12-bit pwm/servo driver - pca9685*. [Online]. Available: https://www.adafruit.com/product/815, Last Accessed: 08/16/21.

[35] T. Instruments, *Dc/dc buck converters*. [Online]. Available: https://www.ti.com/power-management/non-isolated-dc-dc-switching-regulators/step-down-buck/buck-converter-integrated-switch/overview.html, Last Accessed: 08/16/21.

[36] *Traxxas guide to batteries and chargers*, 2020. [Online]. Available: https://traxxas.com/battery-basics, Last Accessed: 08/16/21.

[37] *Adafruit servokit repository*. [Online]. Available: https://github.com/JetsonHacksNano/ServoKit, Last Accessed: 08/16/21.

[38] *Jetson nano*. [Online]. Available: https://www.nvidia.com, Last Accessed: 08/16/21.

[39] *Rplidar github repositry*. [Online]. Available: https://github.com/Slamtec/rplidar, Last Accessed: 08/26/21.

[40] *Rplidar laserscan topic*. [Online]. Available: https://github.com/ros-perception/laser, Last Accessed: 08/26/21.

[41] *Usbcam github repositry*. [Online]. Available: https://github.com/opencv/opencv, Last Accessed: 08/26/21.