REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineer

DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS VEHICLE ENHANCED BY ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS) USING ML

Mustafa Oudah Hani ALSAEDI

Master's Thesis

Supervisor

Assoc.Prof. Dr. Çağdaş ATİLLA

İstanbul, 2024

DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS VEHICLE ENHANCED BY ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS) USING ML

Mustafa Oudah Hani ALSAEDI

Electrical and Computer Engineer

Master's Thesis

ALTINBAŞ UNIVERSITY

2024

The thesis titled DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS VEHICLE ENHANCED BY ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS) USING ML prepared by Mustafa Oudah Hani ALSAEDI and submitted on 28/06/2024 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineer.

Assoc.Prof. Dr. Çağdaş ATİLLA
Supervisor

Thesis Defense Committee Members:

Assoc. Prof. Dr. Çağdaş ATİLLA	Department of Electrical-Electronics Engineering, Altınbaş University	_____
Asst. Prof. Dr. Muhammad ILYAS	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Peren Jerfi CANATALAY	Department of Electronics and Automation, İstinye University	_____

I hereby declare that this thesis meets all format and submission requirements of a master's thesis.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Mustafa Oudah Hani ALSAEDI

Signature

ABSTRACT

DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS VEHICLE ENHANCED BY ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS) USING ML

ALSAEDI, Mustafa Oudah Hani

M.Sc., Electrical and Computer Engineer, Altınbaş University,

Supervisor: Assoc. Prof. Dr. Çağdaş ATİLLA

Date: 28/06/2024

Pages: 91

This thesis discusses the design and implementation of an autonomous vehicle enhanced with advance driver assistance systems (ADAS) using machine learning. This vehicle can be classified as an educational platform suitable for researchers and specialists in the field of autonomous vehicles. Its structure can be easily modified to meet the needs of researchers, and it can be reprogrammed with ease. The work details the construction of the vehicle, including the chassis structure, suspension system, steering system, brakes, and the anti-lock braking system (ABS). Control of the vehicle is achieved through a mobile phone using a control program developed with MIT App Inventor, allowing wireless Bluetooth communication for driving. The thesis also covers the vehicle's key tasks, such as path planning and navigation using a specialized algorithm for selecting the shortest path to the destination. The vehicle is equipped with ultrasonic sensors distributed around it to detect both stationary and moving obstacles. Additionally, a LIDAR sensor is used for obstacle detection. A machine learning model is created to sense obstacles, trained on data collected from various sensors and scenario, and used to implement autonomous driving in simulation and augmented reality. The results demonstrate the vehicle's ability to navigate obstacles during its journey. Finally, the vehicle can recognize different traffic signs, trained using machine learning on a dataset of over 50,000 samples of 43 classes of German traffic signs.

The model is tested for visualization and through the vehicle's camera, enabling it to recognize and respond to all traffic signs appropriately.

Keywords: Autonomous Vehicle, ADAS, LIDAR, Machine Learning, Convolutional Neural Network, Artificial Neural Network, Traffic Sign Recognition.

ÖZET

ML KULLANILAN GELİŞMİŞ SÜRÜCÜ DESTEK SİSTEMLERİ (ADAS) İLE GELİŞTİRİLMİŞ OTONOM BİR ARAÇ TASARIMI VE UYGULAMASI

ALSAEDI, Mustafa Oudah Hani

Yüksek Lisans, Elektrik ve Bilgisayar Mühendisi, Altınbaş Üniversitesi

Danışman: Doç. Dr. Çağdaş ATİLLA

Tarih: 06/2024

Sayfa: 91

Bu tez, makine öğrenimini kullanan gelişmiş sürücü destek sistemleri (ADAS) ile geliştirilmiş otonom bir aracın tasarımını ve uygulamasını tartışmaktadır. Bu araç, otonom araçlar alanında çalışan araştırmacılara ve uzmanlara uygun bir eğitim platformu olarak sınıflandırılabilir. Yapısı araştırmacıların ihtiyaçlarını karşılayacak şekilde kolaylıkla değiştirilebilir ve kolaylıkla yeniden programlanabilir. Çalışmada şasi yapısı, süspansiyon sistemi, direksiyon sistemi, frenler ve kilitlenmeyi önleyici fren sistemi (ABS) dahil olmak üzere aracın yapısı ayrıntılarıyla anlatılıyor. Aracın kontrolü, MIT App Inventor ile geliştirilen ve sürüş için kablosuz Bluetooth iletişimine olanak tanıyan bir kontrol programı kullanılarak cep telefonu aracılığıyla sağlanıyor. Tez ayrıca, hedefe giden en kısa yolu seçmek için özel bir algoritma kullanarak yol planlama ve navigasyon gibi aracın temel görevlerini de kapsamaktadır. Araç, hem sabit hem de hareketli engelleri tespit etmek için etrafına dağıtılmış ultrasonik sensörlerle donatılmıştır. Ayrıca engel tespiti için LIDAR sensörü kullanılıyor. Engelleri algılamak için bir makine öğrenimi modeli oluşturuluyor, çeşitli sensörlerden ve senaryolardan toplanan verilerle eğitiliyor ve simülasyonda ve artırılmış gerçeklikte otonom sürüşü uygulamak için kullanılıyor. Sonuçlar, aracın yolculuğu sırasında engelleri aşma yeteneğini gösteriyor. Son olarak araç, 43 sınıf Alman trafik işaretinden oluşan 50.000'den fazla örnekten oluşan bir veri kümesi üzerinde makine öğrenimi kullanılarak eğitilen farklı trafik işaretlerini tanıyabiliyor. Model, görselleştirme

açısından ve aracın kamerası aracılığıyla test edilerek tüm trafik işaretlerini uygun şekilde tanıması ve bunlara yanıt vermesi sağlandı.

Anahtar Kelimeler: Otonom Araç, ADAS, LIDAR, Makine Öğrenmesi, Evrişimsel Sinir Ağı, Yapay Sinir Ağı, Trafik İşareti Tanıma.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

ABS	:	Anti-Lock Braking System
ACC	:	Adaptive Cruise Control
ADAS	:	Advance Driver Assistance System
AEB	:	Automatic Emergency Braking
ALBS	:	controller and Anti-Lock Braking System
ASBS	:	Anti-Slip Braking System
AVs	:	Autonomous Vehicles
CAS	:	Collision Avoidance System
CNN	:	Convulsion Neural Network
DWA	:	Dynamic Window Approach
ESC	:	Electronic Stability Control
GPS	:	Global Positioning System
GSM	:	Global System for Mobile Communications
HDC	:	Hill Descent Control
LIDAR	:	Light Detection and Ranging
MPC	:	Model Predictive Control
SVSF	:	Smooth Variable Structure Filter
TSRB	:	Traffic Sign Recognition Benchmark

1. INTRODUCTION

1.1 AUTONOMOUS VEHICLES

Autonomous vehicles (AVs), also known as self-driving cars, are a new type of transportation that can drive themselves without needing a human to control them. They use special technology such as various sensors such as radar, LIDAR, ultrasonic sensors, cameras, and GPS to see and understand the world around them. Self-driving vehicles have the potential to make driving safer and easier, reduce traffic congestion, and help people who can't drive get around more easily [1]. However, there are still challenges to overcome before autonomous vehicles can be used everywhere, such as making sure they are safe and follow the rules of the road. Despite these challenges, autonomous vehicles are expected to change the way we think about transportation in the future.

A programmable machine is generally not considered a robot unless it has the following five features. First it must have a body, which is a mechanical device such as a platform with wheels, arms, or similar device that allows it to interact with the environment Secondly, it must sensors are placed on or around its body that can sense the environment and giving useful feedback to the device In order to Thirdly, the robot must have a brain, which is conscious processes the incoming data in the current state of the machine and instructs it to perform actions in response to the situation and fourth, it is necessary to design actuators to move or perform actions on the robot or its attachment. These are usually mechanical or hydraulic pistons. Finally, the power source provides the robots with the energy needed to drive the brain, sensors, and actuators. This can be in the form of batteries, fuel tanks, or other means [2].

1.2 LEVELS OF AUTONOMY

First paragraph. Society of Automotive Engineers (SAE) levels of autonomy range from 0 to 5. These levels are standards that are used uniformly to identify and categorize vehicles' autonomous capabilities, as explained in Figure 1.1: SAE Levels of Automation [3]. [3].

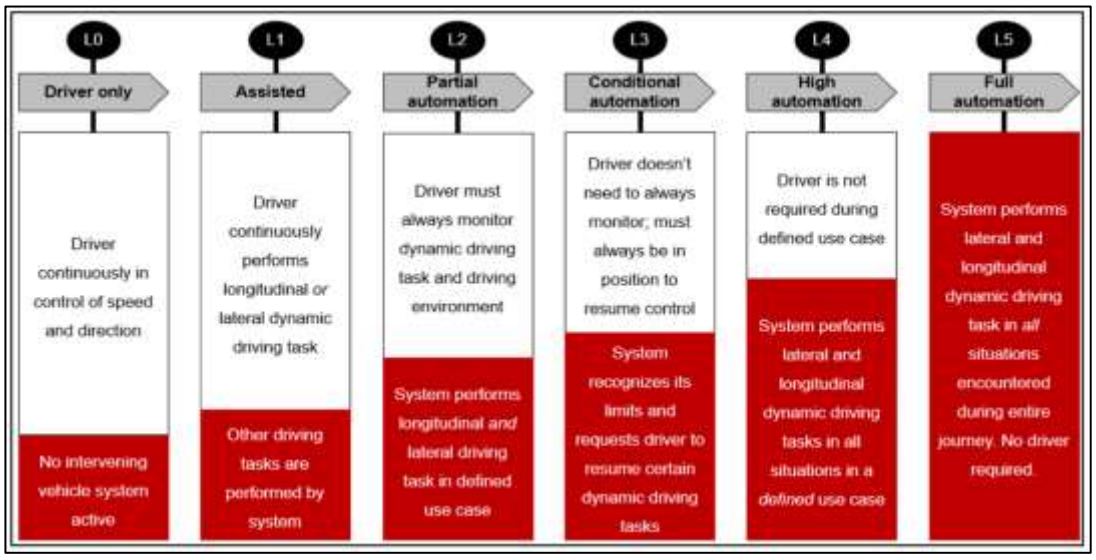

Figure 1.1: SAE Levels of Automation [3].

Level 0 (no automation): The driver is in full control of all aspects of driving, including steering, acceleration, braking, and monitoring the environment.

Level 1 (Driver Assistance): The vehicle can assist with either steering or acceleration/deceleration, but not both simultaneously. Examples include adaptive cruise control or lane-keeping assistance.

Level 2 (Partial Automation): The vehicle can assist with both steering and acceleration/deceleration simultaneously, but the driver must remain engaged and always monitor the environment. Examples include Tesla's Autopilot or GM's Super Cruise.

Level 3 (Conditional Automation): The vehicle can manage all aspects of driving in certain conditions, but the driver must be ready to take over when requested by the system [4]. The driver can disengage from driving tasks but must be available to intervene when necessary.

Level 4 (High Automation): The vehicle can perform all driving tasks under certain conditions and environments without human intervention. However, there may be some conditions where the driver may need to take over.

Level 5 (Full Automation): The vehicle can perform all driving tasks under all conditions without human intervention just passengers [5]. The vehicle is designed to operate autonomously in any environment a human driver can navigate.

1.3 ADVANCED DRIVER ASSISTANCE SYSTEMS

In autonomous vehicles, the advanced driver assistance systems (ADAS) perform autonomous functions such as cruise control adaptation, parking assist, lane departure and median, blind spot warning, and driver alerts such as collision warning. The purpose of these systems is to enhance the safety of the vehicle by helping the driver. Enabled or disabled security systems for ADAS. Passive safety systems inform the driver, for instance, by giving a forward collision warning when a stationary object is detected ahead. Active security systems, however, take an action only when they receive an alert. As an illustration, when forward collision warning is triggered, the active safety system will take charge to control braking and thus prevent a collision. ADAS employs different types of sensors like radar, cameras, LiDAR, infrared, and ultrasonic sensors to capture data that is used to evaluate the situation and determine the right response or the warning.

1.4 FEATURES OF ADVANCED DRIVER ASSISTANCE SYSTEMS

Advanced Driver-Assistance Systems (ADAS) cover the wide range of technologies integrated into cars as passenger and driver safety, as well as comfort improvement. Here are some of the most common ADAS features:

a. In the ADS system, ACC (Adaptive Cruise Control) is the main component. Adaptive Cruise Control (ACC) can modify the car's speed or slow down to ensure a safe space between the vehicles ahead of one. Given the ACC system approves and validate its safety measures by using either of these: radar, cameras, or some architectures.

b. The electronic stability function (ESC) is the control system installed in modern vehicles. It tracks cars positioning and takes over if they slide out of control or lose sight of. Through ESC, it is possible for the system to apply brakes to certain wheels, when necessary, on slick roads and reducing the engine power to help with manoeuvring. It gives stability and security of the vehicle by stopping from an accident and staying in control under challenging condition.

c. Parking sensors are the tiny instruments which are normally mounted on the vehicle's bumper or rear panel that assistance in parking and avoiding obstacles. They use ultrasonic or electromagnetic technology that is installed on the vehicle and will alert the driver about objects within its vicinity through visual or audible signals. This

supports the pilot-less collisions with the obstacles, signalizing parking as safer and more convenient by transmitting in real time data concerning the distance from objects (behind and front of the vehicle).

d. Automatic Parking Assistance is an advanced feature that provides an easy life when it comes to parking, making it practically stress free. By means of sensors the system can find parking spots and with almost no human contribution it can obliquely or parallelly park the car depending on the situation. The operator activates the automatic parking system, and the car starts to guide itself into the parking space. Along the way, the driver puts in the brakes and acceleration. It allows you to fit in tight areas and in a way, can make parking more precise and comfortable, especially for the people who have a problem with parking.

e. The Hill Descent Control (HDC) is a vehicle function that helps drivers to negotiate in downhill climbs or slopes. It automatically controls the braking system to maintain a proper speed allowing for a safer ride thus eliminating skidding or loss of control. HDC is especially directed for off-road and difficult situations when deceleration is applied smoothly and safely.

f. The Lane Change Assistant helps the driver make safer lane changes. This technology is equipped with sensors and cameras to search for dangers in the same lane, say cars in the blind spot, or the ones coming fast from behind, when a driver switches lanes with the turn signal on. It encourages a safer lane-changing procedure by virtue of issuing warnings either visually or verbally to the driver. However, if there is a need to change lanes, it will be safer and more convenient, but drivers should exercise caution and be sure to observe their surroundings before such a move.

g. A Collision Avoidance System (CAS) is the most significant security dimension that utilizes instruments, cameras, and complicated technology to determine possible collisions with other vehicles, pedestrians, and obstacles on the road. If danger is detected, the system will give warnings to the driver, and it may do things such as braking or steer to avoid or minimize the collision. The CAS (Collision Avoidance System) significantly boosts driver safety through early signals and drivers' guidance in emergency situations to avoid accidents and enhance general road safety.

h. The automotive navigation system is the technology that is installed in vehicles to give live guidance and directions. It is one of the systems which enables the vehicle to attain the goal location. Such systems use GPS (Global Positioning System) to navigate to the desired destination.

i. Blind spot monitors, equipped with sensors normally installed on the side mirrors and rear bumper, will constantly scan the driver's blind spots to ensure his/her safety. When another vehicle enters the blind spot area with an indication of visual or auditory which for example illuminated icons on the side mirrors or vibrations of the steering wheel. This technology improves the safety of driving by informing drivers about vehicles that are not visible to them and it decreases the danger of accidents that happens when changing lanes or making turns.

j. AEB or Automatic Emergency Braking is a safety feature in modern vehicles that utilizes sensors, cameras, or radar to scan the front of the vehicle for imminent collisions. In case the AEB detects an upcoming collision with another vehicle, pedestrian or any obstacle and the driver is unable to break or avoid the collision then the AEB actuates the brakes. This emergency braking technology is geared at reducing accidents, lowering the severity of collisions, and increasing road safety by providing emergency braking assistance when necessary.

k. Rain sensors are common devices on current cars that switch on windshield wipers when sensors detect rain or moisture on the windshield.

l. Automotive HUD (Head-Up Display) is a transparent screen that displays the driving information which is in the driver's line of sight directly onto the windscreen. With such innovation the drivers can easily access important information like speed limits, navigation directions, and the vehicle conditions through their smartphones, and at the same time the drivers never have to take their eyes off the road. HUDs help driving safety and convenience because the information is given as real-time, and the driver can look at the road while at the same time the driver understands the surroundings better and thus minimizes distractions.

m. The Driver Emergency Stop Assistant is one of the aids for drivers in cases of extreme situations. In the situation the emergency system is activated, it can take the control of

the vehicle and safely bring it to a stop in this case of the driver being incapacitated, unresponsive or unable to control the car. This is attained by controlling steering, braking, and acceleration so that the car comes to an accurate stop either at the side of the road or in a safe place. This decreases the chance of an accident for the driver and other road users, thus keeping everyone away from harm.

n. Driver's Drowsiness Detection is a driver's safety features having ability to sense and detect the drowsiness and fatigue of the drivers. The technology is battery operated and consists of sensor and camera systems which, among others, detect drivers' eye movements, head position and navigation patterns. When the system recognizes features like driver falling asleep or irregular driving habits, it issues warnings and/or shakes the driver gently to wake him/her up. Such warnings could be visual, acoustic, or even haptic (seat vibration). Driver Drowsiness Detection makes the road free from crashes by controlling the driver's drowsiness, which relies on the alertness as well as driving breaks, when necessary, on long trips.

o. Lane Centring Assist is a driver's assistance feature that helps vehicles to drive within their lanes. By means of sensors and cameras, it is designed to track the lane markings and to gently steer the vehicle back to the centre of the lane in case of unintended drifting. Such technology improves the driving safety as it lessens the possibility of vehicle leaving the lane and provides the atmosphere of serene driving during a long journey or in a busy traffic. It cannot be used as a replacement for attentive driving and demands the driver to stay focused and hands on the wheel.

p. TSR or Traffic Sign Recognition is an automotive technology that uses cameras and image recognition software to recognize and classify traffic signs like speed limit signs, stop signs, and no-entry signs.

q. Vehicle communication system allows vehicles to transmit information to each other as well as to infrastructure elements which can be traffic lights or road signs. This form of communication which is commonly known as V2X (Vehicle-to-everything) uses wireless connections to send information about road conditions, traffic, and dangers. V2X technology is characterized by provision of real-time updates and warnings to

drivers with a view of making informed decisions on the road and proper navigation through different traffic situations.

1.5 MOTIVATION AND CONTRIBUTION

1.5.1 Motivation

One of the most important causes of traffic accidents is due to human errors while driving, where quick responses and high concentration are among the most important priorities in driving. For this reason, autonomous vehicles have been put forward as a solution to this problem, especially those featuring Advanced Driver Assistance System (ADAS). The computer works as an assistant to the driver in collecting data from the road through a group of different sensors, such as radar, LIDAR, ultrasonic, thermal cameras, and others, so that this data is processed using different algorithms and appropriate decisions are made in a period not exceeding a fraction of a second. Accordingly, we have great motivation to design and implement an ADAS autonomous vehicle. This vehicle performs several tasks, the most important of which are drawing a road map and choosing the shortest path while heading to the target point. It also performs a comprehensive survey around the vehicle, bypassing obstacles, and recognizing traffic lights using machine learning and other characteristics that will be explained later in the upcoming chapters of this thesis.

1.5.2 Contribution

This thesis contributes to the field of autonomous vehicles and advanced driver assistance systems (ADAS) by designing and implementing an educational platform in the form of an autonomous vehicle. The vehicle is enhanced with ADAS functionalities, making it programmable and customizable for researchers to conduct experiments and research in autonomous navigation and driver assistance systems.

The key contributions of this work are as follows:

a. Design and Implementation of an Autonomous Vehicle: The primary contribution of this work is the design and implementation of a fully autonomous vehicle prototype. This includes the selection and integration of hardware components, development of control algorithms, and implementation of software systems necessary for autonomous operation.

b. Integration of ADAS: The thesis demonstrates the integration of various ADAS functionalities into the autonomous vehicle, including path planning, obstacle detection, and traffic sign recognition. By using machine learning algorithms, the vehicle can perform these functions autonomously, enhancing its safety and efficiency.

c. Implementation of Wireless Control Interface: Another significant contribution is design and implementation of a wireless control interface for the autonomous vehicle. This allows users to remotely control the vehicle's movements and monitor its status using MIT App Inventor as a mobile application, providing flexibility and convenience in operation.

d. Machine Learning for Obstacles Avoidance and Object Detection using LIDAR and ultrasonic sensors: The thesis showcases the application of machine learning techniques for obstacle avoidance during the vehicle's journey, utilizing data acquisition, preprocessing, and training the classifier. Additionally, it demonstrates the use of LIDAR and ultrasonic sensors for object detection, detailing the acquisition of data, preprocessing steps, and training of the classifier. These implementations highlight the feasibility and effectiveness of machine learning in enhancing the vehicle's ability to detect and avoid obstacles during maneuvers.

e. Traffic Sign Recognition Model: A machine learning model is trained using the German Traffic Sign Recognition Benchmark (GTSRB) dataset to recognize traffic signs. This model is integrated into the vehicle's control system, enhancing its ability to interact with traffic signs.

f. Testing and Validation: Various testing scenarios are designed to evaluate the vehicle's performance, including navigation, obstacle avoidance, and traffic sign recognition. The results demonstrate the effectiveness and reliability of the implemented system.

g. Contribution to Education and Research: Overall, this research contributes to the advancement of education and research in autonomous vehicles and ADAS. The designed vehicle provides a practical platform for students and researchers to explore and experiment with autonomous driving technologies, fostering innovation and learning in this rapidly evolving field.

Overall, the contributions of this thesis advance the field of autonomous systems by providing a comprehensive framework for designing, implementing, and testing autonomous vehicles enhanced with ADAS using machine learning techniques.

1.6 THESIS OUTLINE

This thesis consists of six chapters. The descriptions of all chapters shown as follows:

CHAPTER ONE: This chapter provides a comprehensive introduction to automated vehicles and vehicles equipped with advanced driver assistance systems and explains the characteristics of these systems and levels of automation. It also provides a comprehensive literary review of previous research and dissertations related to automated vehicles and those equipped with advanced driver assistance systems, and a review of the application of machine learning in advanced driver assistance systems and in the extensive section. In the chapter, we discussed the most important motivations that made us do this work and what contribution our research makes to this field.

CHAPTER TWO: This chapter covers the theoretical foundations relevant to the thesis, including the path planning problem and the A* algorithm. It also discusses the vehicle navigation system, Convolutional Neural Networks (CNNs), Anti-Lock Braking System (ABS), Slide Mode Controller (SMC), and the hardware components of the proposed vehicle model.

CHAPTER THREE: This chapter details the design and methodology of the system architecture, focusing on path planning and obstacle avoidance using the A* algorithm. It also discusses the hardware system design for the vehicle robot, including chassis design, suspension system, braking system, steering system, motor and motor driver, sensors integration, power supply system, battery charger, vehicle control, communication modules, horn, and indicators. Additionally, it covers vehicle control using a mobile platform and the design of object detection and traffic sign recognition systems using machine learning.

CHAPTER FOUR: This chapter presents the practical results of the thesis, including system requirements, scenarios of the vehicle model, practical path tracking, path tracking including obstacles, simulation and practical implementation of ABS, practical implementation of object detection, and practical implementation of traffic sign recognition. It includes performance evaluation, visualization results, and practical experimental results.

CHAPTER FIVE: Discusses the conclusion and future work of this project.

2. LITERATURES REVIEW

2.1 REVIEW OF AUTONOMOUS VEHICLES

Pepy and al. (2006) focused on the problem of the vehicle path planning applying a dynamic vehicle model which had been implemented as an improvement on the work done by the authors who used only simple kinematic models previously. In parametric models, speed accuracy until a low regime was good; however, they diverged so much when the speed regime was too high that consequently they provided paths that were unfit for completing the mission effectively. To address this shortcoming of the recommended navigation algorithm which requires static vehicle model the authors developed realistic path planner based on dynamic model of a vehicle. They are also characterized based on their modularity and adaptability as it can be easily adjusted to a specific car class or make. In addition to that, future work will be done by applying more effective and dynamic models, which are based on Pacejka's or Dugoff's formulas, by replacing the size of the vehicle with the current dynamic model and putting it in line with the new-state function. Notwithstanding, studies could also involve pursuing accuracy and dynamic model realism in place following as well as reduction of computation time to make it real time [6].

Naderi et al. (2010) came up with solutions of vehicle stability during turnaround and braking named Anti-Slip Braking System (ASBS) controller and Anti-Lock Braking System (ALBS) controller respectively. This was done to enhance the integrity of the braking system for the safety of the passengers during such critical braking scenarios, like on U-split or slippery road surfaces where such intense pressure on the brake pedal can cause wheel lockup and while in turn lead to vehicle instability and unwanted lane changes. With the aid of these features, ASBS and ALBS ensure a more stable driving by correcting the position of the car. This method is based on referencing a yaw angle to predict a stable vehicle path and then controlling the sideslip of each wheel using Anti-Slip fuzzy controllers, for each individual wheel. Thanks to this regulator the number of mistakes is significantly reduced in each wheel improving stability of the vehicle during braking and turning on different road surfaces [7].

Zidek et al. (2012) proposed the integration of search rules and satellite map image data with edge detection for trajectory planning. The algorithm applied was depth-first resulting in an

optimal trajectory path between the initial stage and the final stage. GPS data of start and end points were used to set the base point and establish the location of the device on satellite map. Satellite image data were used for obstacle detection through edge detection algorithms which detected buildings. A virtual grid map was filled with unreachable positions next using the data gathered by the edge detector analyzer. For the real-world deployment of the algorithm on the real platform sensors (example, infrared or ultrasonic sensors) would be necessary for obstacles detection during movement as well as re-calculating the path. This algorithm is specifically designed for use in intelligent devices for exploration in unfamiliar areas, particularly in highly urban places [8].

Khan, G. T. et al. (2013) pointed out the solution to the problem of a lack of geographic information and navigational system for a ground-based mobile robot by using GPS system. The robot made use of a GPS system for the navigation part, sensors - LV-Max Sonar Ez 4 – for obstacles detection and GSM Modem for the communication process. The robot would be guided by these waypoints sent using the GSM Modem and would proceed to move around obstacles on its route. This experiment was done on the floor of a university campus with a testbed robot navigating in a planned fashion using GPS. According to the research, the GPS was identified as an efficient and accurate system of mechanisms guiding robot navigation. The instruments showed that the GPS navigation was easy, highly accurate and working with bad weather filtering, and hence, it was superior in comparison with different options [9].

Zhou et al. (2014) introduced a navigation strategy based on autonomous driving mode that was developed for an outdoor mobile robot running in actual scenarios. Dead-reckoning system of the robot had the encoder wheel, electronic compass, and GPS to assist. To improve the navigation accuracy, they combined the extended Kalman filter (the method of data fusion) for the encoder disk and the electronic compass. The GPS received was indispensable to the dead reckoning error accumulated and corrected data reduction. In addition to that, a laser range finder was applied as an obstacle detection method and environment mapping tool. Authors used the Field D* algorithm and DWA for obstacle avoidance, as well as the path-planning method. Practical outcome confirmed the capability of outdoor mobile robot for autonomous driving with the complex road situations [10].

Oroko et al., 2014 ascertained the diverse approach towards mobile robot navigation relating to path planning and obstacle avoidance. The paper discussed three methods for obstacle avoidance and path planning: Bug Algorithms, Potential Field methods, and Vector Field Histogram techniques which are active sensor-based techniques. The author emphasized a more optimized self-navigating system that could be constructed through a combination of methods utilizing their strong points instead of their weak points [11].

Lin, et al. (2014) proposed a fuzzy logic control method for the ABS (Anti-lock Braking System) in an EV (electric vehicle) that uses electromagnetic brakes. It had been a way to reduce the acceleration and deceleration of the car's wheel. The researchers used Simulink and AMEsim to create a model of the EV braking performance on low adhesion roads providing the model for co-simulation. The fuzzy logic ABS controller simulation obtained that it actually raises the braking effectiveness and stability of EVs. The authors concluded that adjusting the model parameters based on the factors under investigation is a direction of further study, which would make the model more applicable to ABS studies in EVs [12].

Al-Faiz et al. (2015) suggested an approach of mapping out a path between a given start and goal point in a satellite image with the use of the A* algorithm. This operation identifies buildings in the satellite image by looking at their roof color then splits the search area into cells using a cell decomposition grid method that makes the estimates. The optimal routes determined by the software are transported through the WIFI receiver to the autonomous robot as a collection of set points. The robot, possessing 5 sharp IR sensors for obstacle detection and avoidance, GPS, and compass, navigates its path exactly as it was described in the map. To achieve smooth motion with minimal deviation from the goal, the PID (Proportional-Integral-Derivative) controller which will control the speed of mobile robot's motors according to readings of the digital compass [13].

Wang (2018) proposed a model predictive control (MPC) approach for autonomous four-wheel steering (4WS) vehicles focusing on low-speed parking maneuvers and high-speed lane changes trajectory planning. The study compared FWS vehicles with 4WS ones, noting that FWS vehicles are better in maneuverability at low speed but less stable when driving fast. The study employed the curvilinear coordinate frame (Frenet frame) in the kinematic double-track model of a 4WS vehicle to lower the side displacement of the MPC optimal trajectory planner. The planner was designed to keep heading and velocity errors at

minimum as well. The simulations using the specified trajectory planner proved that 4WS could steer various paths with decreased the heading error, shorter longitudinal distance and smaller lateral deviations compared to FWS [14].

2.2 REVIEW OF CURRENT ADAS ARCHITECTURES

Zhongzhen (2017) put forward the design of LIDAR (Light Detection and Ranging) perception system, consisting of ground detection, object detection, object classification and object tracking. These subtasks are necessary for many autonomous vehicles uses. To realize ground segmentation, a probability occupancy grid map method was implemented to deal with problems such as over segmentation, under-segmentation, and slow segmentation on non-planar surfaces. Object detection method used a point cloud clustering algorithm based on RBNN on a static Kd-tree. We have applied a supervised learning approach using the LiDAR sensor to classify objects into "Sedan", "SUV", "Van" and "Truck" respectively. In order to handle disturbances and motion uncertainties, we have established a Smooth Variable Structure Filter (SVSF) that involves the integration of Hungarian algorithm (HA) and Probability Data Association Filter (PDAF The thesis also includes a detailed experimental evaluation of each task to confirm the robustness and efficiency of the perception system that will be developed [15].

Viktor (2017) outline the development and deployment of an experimental ADAS platform used for automotive engineering. The thesis discusses the most significant automotive concepts concerning ADAS viz. Electronic Control Units (ECU), communication protocols in cars, and the OS that is developed for automotive system. It also explains the origin of ADAS technology and the current technology status. he presents platform architecture at the micro level: hardware specification and reasons for the decision. The architecture encompasses various ECUs mutually interconnected via Ethernet network with several switches. The platform could incorporate an actual car with actuators and sensors, which would recreate driving scenarios from the real world. Rather than that, sensor data reading, actuator control and the power of the interface for car control have been already implemented. The formed system represents a proof-of-concept version which can be utilized as a basis and developed for different areas of application in the context of ADAS functions and accident engineering [16]

Sachin, in (2023), presents the detailed Test framework for Unit, Integration and System Testing levels focusing on speed, and modularity to resolve the issues through the rapid execution and a prior test plan. At the Unit Test level, we recommend two major shifts with the update of actionable items for new developers that could be considered during code reviews. University of Waterloo, the Structured Testing Framework (UW-STF) is a way that is integrated, and system-level tested to supply some benefits and the detailed implementation of ROS-based development. This is doing via the integration of data/results from the CARLA simulator for end-to-end assessment. The test framework is used to our purpose about codebase UWAFT, related to connected and autonomous vehicles. The findings have demonstrated that there is improved readability and understanding at the unit test level, there is a robust automated testing at the integration level, and there is also transparency in the performance of the algorithm (average F1-score of 0.77 and average OSPA of 2.42). Besides, UW-STF performed same test suite as the standard ROS integration test framework, but with close to 60% reduction in code lines and with the significant cuts in CPU and memory utilization [17].

Meng (2023) shows a method for constructing an open framework for the estimation of distance between a vehicle carrying sensors and different road objects through the fusion of data from a camera and LiDAR. It is focused on the integration of ADAS device which have sensor attributes for real-time road occupancy and the obstacle distance detection purposes in conjunction with the automated braking systems. The architecture involves a low-level approach based on geometrical projection for a 3D mapping using a point cloud to feed to the 2D camera images. The obtained fused data from yolov7-detector is then used to estimate the last digit value of the number. My custom ROS pipeline, which is implemented in ROS, consists of a sensor calibration method, a yolov7 detector, point cloud down- sampling and point cloud clustering, and 3D to 2D transformation between frame of references. The pipeline means for the data association and to estimate the distance of identified road objects. For the evaluation of the framework's accuracy and performance real-world scenarios with specific sensors data are simulated and analyzed. With such a system, the pipeline made on an embedded Nvidia Jetson AGX Xavier AI Vehicle Computer manages well the task of recognizing objects and measure the distance from objects, maintaining the rate of 5Hz. Through this theorem a framework that provides for a pliable and resourcing effective data

associating approach using the conventional automobile sensors is presented, as an enhancement for the environment perception of the assisted driving systems [18].

Teck et al. (2019) focuses on the earlier integration of cameras and radars sensors, an area that has been more neglected. They suggest given the minimally processed radar signal together with the camera frame to the deep learning architecture to enhance the perception module in terms of accuracy and robustness. The assessment, which is carried out with real-world data, shows that the combination of radar and camera signals can decrease lateral error by 15% in the detection of objects when applied. This possibly means that blending the radar data into the perception pipeline early makes the system to stand out in the overall performance [19].

2.3 MACHINE LEARNING TECHNIQUES IN ADAS

Omar (2023) is centered on the area of autonomous driving and the application of Machine Learning (ML) methodologies, with special emphasis on Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN). The thesis attempts to train a neural net for autonomous navigation within the Car Simulator environment provided by Udacity. The design stage comprised data gathering and treatment as well as image optimization which greatly determined the performance of the model. The outputted model displayed significant robustness. This, then, was merged into a simulation, allowing for real-time interactive application and effective car management. The paper emphasizes the contribution of machine learning in the development of autonomous vehicles, and proves the fact that neural networks can help in effective autonomous navigation .

Hampus et al. (2018) present a prediction method for surrounding objects of byway cars using RNN, as recurrent neural network. The system employs the raw LiDAR data directly for producing the final occupancy grid so as to finally give way to an end-to-end solution. The authors utilize the sub-meter level accuracy of modern LIDARs by taking into an account the LiDAR as the ground truth for unoccluded scenes during the training phase. Via looping sensor inputs and comparing the response with an actual scan, the network gains the ability to track hidden objects without the labels. LiDAR point clouds are casted into a 2D occupancy grid in which each channel just has one value. Hence, machine learning algorithms can be applied, for example, by using convolutions to do feature extraction. Created networks encompass such components as convolutional and long short-term

memory layers and were examined on real-world data. We evaluated five different network architectures, and the best one displayed a low variance performance. Output data show that this network is able to correctly localize an object in a scenario of occlusion, such as in a scene on a highway when a car is overtaking another one. The network obtains an accuracy score of around 90% during the first 1.1 seconds of the blackout after driving through a highway scenario and considering all the cells [20].

Maryam (2019) is a study that aimed to integrate Artificial Intelligence (AI) network types with Advanced Driver Assistance Systems (ADAS), especially neural networks (NNs), to perceive distance between vehicles. The research examines, which NN architecture is the most accurate for this task, considering the complexity of NNs. The objective of the research is to establish if AI can be a reliable processor of autonomous vehicles (AVs) through the incorporation of ADAS into AI. The ADAS developed in thesis shall be using monocular vision and machine learning. A CNN model was trained on a dataset of 200,000 images which achieved 96.75% accuracy in determining whether the image has a valid license plate or not. A sliding window technique is deployed to identify a license plate sub-section of an image, and if found, the algorithm is going to keep the image sub-section. In this context, a heatmap threshold is applied to the sub-images to reduce number of false detections. Upon recognition of a license plate, the algorithm determines the distance of the vehicle to the license plate plate which can be up to 1-meter distance accuracy. The ADAS is developed to be user-friendly and can be incorporated systematically in future AV platforms [21].

Jesudara (2022) developed a platform for autonomous driving research and education. The study aimed to create a comprehensive platform with a drive-by-wire system and sensor packages suitable for testing and deploying machine learning algorithms on a scaled vehicle. The platform addresses issues related to reproducibility, onboard processing capability, vehicle scale, and system cost. It integrates drive-by-wire and an autonomous system with sensor packages in an easy-to-implement format. The platform includes flight controller programs with a GPS module for mid to high-level motor controls, and a laptop with a GPU for handling advanced algorithms. The platform was validated through a deep-learning-based behavioral cloning study. Its affordability and adaptability make it beneficial for broader research and the education community in the field of autonomous driving [22].

Ilias et al. (2021) introduce a new traffic sign dataset comprising the Carla Traffic Sign Detection (CTSD) and Carla Traffic Sign Recognition Dataset (CATERED) for detection and recognition tasks. They use this dataset to train and evaluate a deep Auto-Encoder algorithm, which shows high accuracy in detecting and recognizing distorted traffic signs. Additionally, they extend the system to a federated learning environment, demonstrating its suitability for modern decentralized architectures [23].

2.4 SUMMARY OF LITERATURES

Our review of the literature highlights both the strengths and the gaps in current research on autonomous vehicles and advanced driver assistance systems (ADAS). The studies reviewed present a range of approaches and technologies, from dynamic vehicle models for path planning to sensor integration for obstacle detection and navigation strategies for outdoor mobile robots.

The strength of our work lies in its comprehensive examination of various aspects of autonomous vehicle technology, including path planning, obstacle detection, navigation, and control systems. By synthesizing findings from multiple studies, we provide a detailed overview of the state of the art in ADAS and autonomous vehicles.

However, our review also reveals several areas where further research is needed. For instance, while some studies have made significant advancements in path planning and obstacle avoidance, few have provided robust algorithms that can be generalized for real-world applications. Additionally, the lack of standardized datasets for benchmarking purposes remains a challenge in the field of traffic sign recognition.

Overall, our review highlights the need for more research and practical implementation of the technologies discussed to enhance the reliability and efficiency of ADAS and autonomous vehicles. The studies reviewed demonstrate the potential of machine learning algorithms and sensor integration in improving the performance of autonomous vehicles, but further research and practical implementation are needed to realize these advancements fully.

3. SYSTEM DESIGN AND IMPLEMENTATION METHODOLOGY

3.1 INTRODUCTION

In Chapter Three, titled "System Architecture Design and Methodology," the focus is on designing a comprehensive system for Advanced Driver Assistance Systems (ADAS). The chapter begins with an introduction highlighting the importance of system architecture design and outlines the subsequent sections. It then delves into the path planning and obstacle avoidance aspects using the A* algorithm, detailing the general system architecture and the hardware system design for the vehicle robot. This includes discussions on chassis design, suspension system, braking system, steering mechanism, motor and motor driver, sensor integration, power supply system, and communication modules. Additionally, the chapter covers the development of an object detection and recognition system using machine learning, specifically focusing on traffic signs, traffic lights, and lane keeping. Throughout the chapter, the integration of various components and technologies is emphasized to ensure the effective functioning of the ADAS.

3.2 PATH PLANNING PROBLEM

The challenge of path planning in mobile robotics is when a robot has to navigate from a starting point to a destination point quickly. The two components of the path planning problem and its solution are discussed next. The first subproblem, that we call generate-graph, aims to change the continuous environment into a graph (for instance, the workspace is the area where the agents travel). The find-path subproblem is the second subproblem. The goal of the find-path subproblem is to compute the shortest weighted path from the initial point to the target point.

3.2.1 Path Finding Algorithm

Developing the method of describing the surroundings is the first step, then we have to go through calculating the route which is the best one through that representation. The path planning cell is providing with both graph search method and cell breakdown. Several disciplines, including problem solving, computer networking, artificial intelligence, and mechanical manipulation are involved in the retrieval [24]. From easy to complex, the list of algorithms continues to grow daily. They all are accepted and many are quite popular with

a lot of people. Criterion A: The easiest strategy is just to keeping walking straight forward until you encounter any difficulties encountered. Any item which comes straight towards you will turn away; you take the path around it, as an option to avoid it. While this algorithm is based on the ignorance of the path details, it can be considered as a static algorithm example of "blind search". Dijkstra's Algorithm, Depth-First Search, and Breadth-First Search are known as dynamic algorithms for "blind search". The point of device generally is that algorithms plan the course for the entire way before the train starts. Best-first way will be applied towards the target with computation of the node cost using a heuristic estimate. Initially , profitable structures as predicted by profit are extended. A* algorithm, which is generally used as a blend of best-first and Dijkstra's algorithm, stands out as the most common algorithm.

3.2.2 A Star Algorithm

In 1968, the shortest path algorithm was presented for the first time in the paper, written by Bertram Raphael, Nils Nilsson, and Peter Hart. The global search space with path finding is just one of the numerous problems which this technique will be able to solve. This algorithm is actually the most popular and widespread; its implemented into several real-time strategy games. The A* algorithm operates under the same principles as the best-first and Dijkstra algorithms; however, the values of the nodes are computed in a rather unusual manner. The sum of original cost of each node from the very beginning and the synthetic cost estimates from the node towards the optimum becomes the value of that node. It alternatively performs the best initial search and Dijkstra's algorithm.

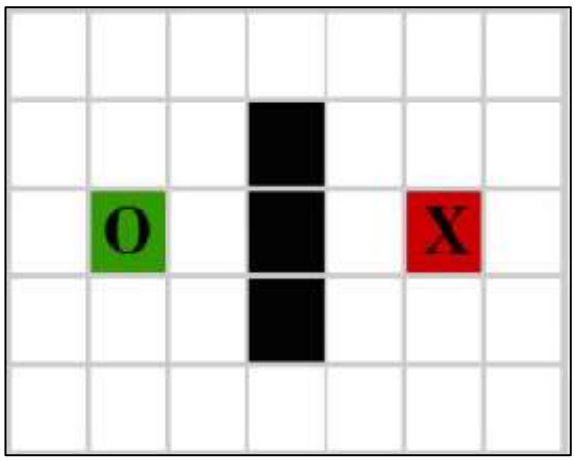

Figure 3.1: Starting and Destination Points in A* Search Algorithm **[24]**.

In fact, the heuristic for Dijkstra search stays 0 every time, as it is an A* search. Moreover, no other algorithm: if it applies the same heuristic, will explore less nodes or find the optimal path, which shows that this method exploits the heuristic function effectively. When a desired path is availabke, A* algorithm creates it using the starting point and the destination point. There are two list arrays in this algorithm: an open ballot and a closed ballot. Closed list locates the cell to be saved whereas the open list does this operation. The cell in Figure 3.1 designated as "O" is marked the starting place or node, and the cell identified as "X" represents the destination. Black squares denote barriers, whereas white squares are walking routes [25]. The following is the flowchart of A* algorithm, as shown in Figure 3.2.

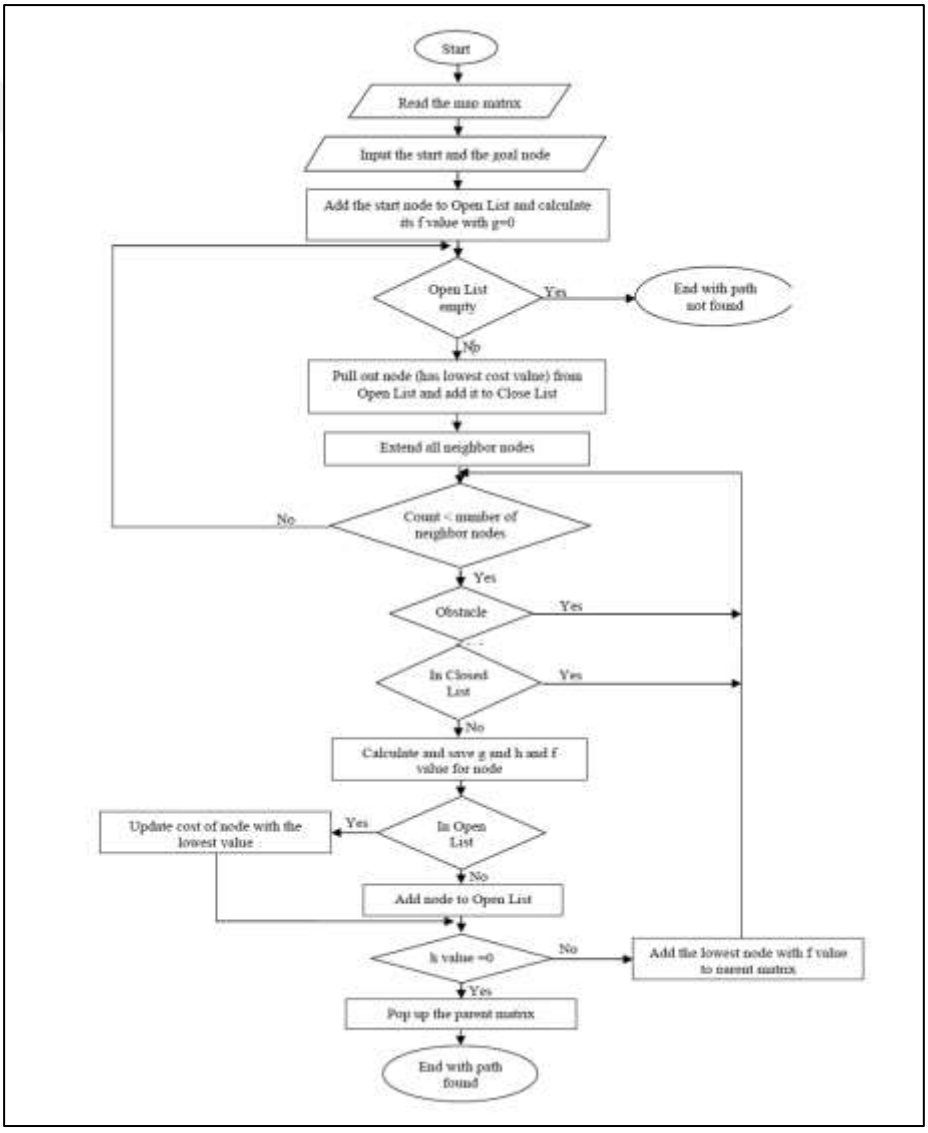

Figure 3.2: Flowchart of A* Algorithm [24].

3.3 VEHICLE NAVIGATION SYSTEM

Automated vehicles, like animals, need to perceive and understand their environment. This requires information and data about the world, which is gathered by sensor devices. These sensors act as transducers, converting physical quantities in the environment into electrical signals that describe the environment's state. The automated vehicle is designed to take into account this information and then produce its output that helps it to reach its goals. In the field of vehicle navigation two pioneering sensors – for identifying a vehicle's position and detecting obstacles – are indispensable. There are 2 types of sensors that are usually encountered: the absolute and the relative. Navigational systems featuring sensors of absolute position using well-known locations in the environment to determine the location of vehicles enables the car to situate itself. However, the sensors that indicate positions relatively, such as zoom inertial flow and odometry, determine the vehicle's position relative to its original point and the movements made. Unrelated, odometry with encoders or a speedometer helps to record wheel rotation, and the car can obtain the heading and position data for prediction based on the model. This technique might seem easy and cheap but as a result, it could result in a low level of precision. In contrast inertial navigation amounts to the integration of different elements such as gyroscopes and accelerometers which are used to measure the rates of acceleration as well as the rotations. But, it is less accurate than ones which merely measure the ones not change over time and whose prices are lower. There are many types of magnetic sensors that can be used for heading. Examples of these sensors include compasses and magnetometers. The advantage of a compass is that it works almost at any location on Earth's surface as it relies on the magnetic field of the Earth. Another example of active beacon-based navigation-sensor is GPS (Global Positioning System) with the absolute positioning ability, which send signals to the beacon and thus gives accurate information to the vehicular driver. A beacon, on this account, is: signals, sounds and radio frequencies of coordinated release. Such implement goes frequently in cares and planes because it is accurate and fast.

3.4 PATH PLANNING AND OBSTACLE AVOIDANCES FOR ADAS USING A* ALGORITHM

3.4.1 General System Architecture

The system block diagram is shown in Figure 3.3. Mobile phone (Apple or Android operating system) is connected through Bluetooth with another Bluetooth in vehicle that is interfaced with Arduino mega which is the microcontroller of mobile robot. That allows mobile phone to communicate with mobile robot. This system consists of the following hardware and software parts. The hardware used in the system consists of three parts:

a. Automotive Vehicle (Vehicle Body).
b. Wireless communication.
c. Mobile phone (Android operating system).

The software used in the system consists of:

a. The open-source Arduino Software (IDE) to program the Arduino boards.
b. Application on mobile phone (Android operating system) to make GUI which is responsible for the main operation of the PC.

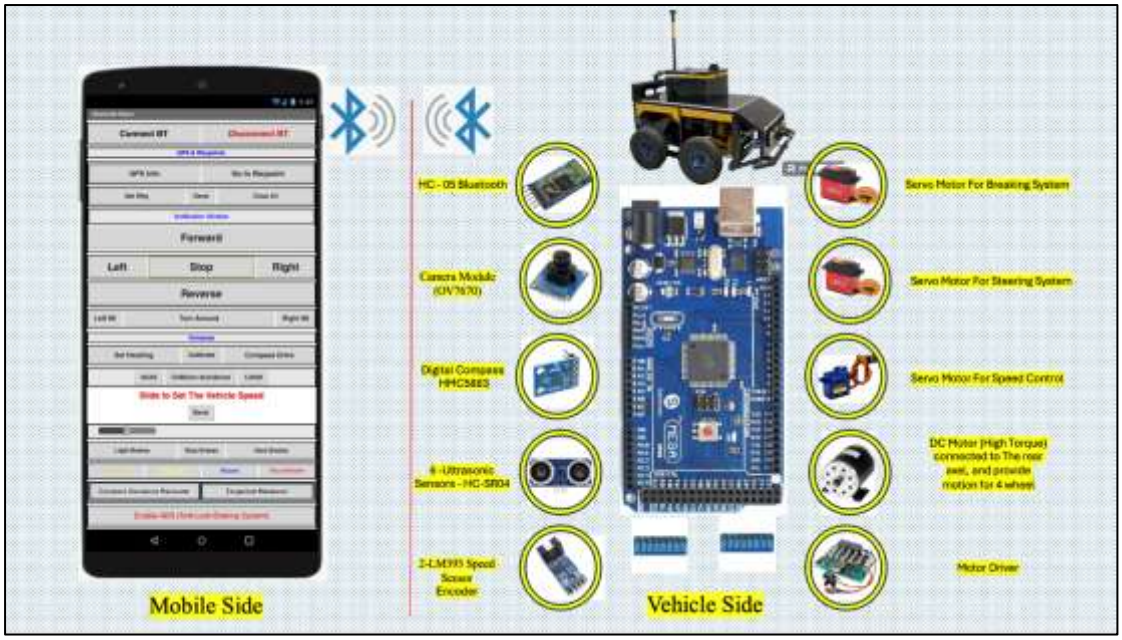

Figure 3.3: General System Architecture.

3.4.2 Hardware System Design for Vehicle Robot

3.4.2.1 Chassis design

Many types of chassis are available. They consist of body of robot with its motors and enough area to add one electronic circuit control panel. The chassis was built to act as the base of the robot which contains 1 DC motors high torque (the motion distributed to 4 wheels) with gear box, differential gearbox to make the vehicle turning smoothly through different angles of orientation, two electromechanical brakes attached to rear wheels disc along with servo motor to implement the braking system, servo motor in the front of the vehicle to provide the steering mechanism, two axels and each axel equipped with two tires. This part is shown in Figure 3.4 for mobile robot chassis. Based on the difference speed between the two wheels, a mobile robot can be moved in any direction on 2D layout.

Figure 3.4: The Designed of Mobile Robot Chassis.

3.4.2.2 Suspension system

A vehicle's suspension system is a crucial assembly of components designed to manage the vehicle's weight distribution, absorb shocks from the road surface, and deliver a smooth and

stable ride for passengers. Its primary functions encompass supporting the vehicle's weight evenly across its wheels, minimizing discomfort by absorbing road shocks and vibrations, and ensuring constant contact between the tires and the road for optimal traction and handling. Through the suspension system which insulates vibrations and disturbances, the resulting comfort in the vehicle cabin is enhanced while the car cabin noise levels are minimized. Besides, it has a role in the handling characteristics of the vehicle, including responsiveness, agility, and stability on evasive actions. A suspension system usually consists of springs, shock absorbers, control arms, anti-roll bars and many types of linkages and joints that get in combination to deliver a smooth and controlled ride. Adaptive suspensions are another technology that is adopted to improve performance. They adjust the damping rates and heights of the vehicle based on driving conditions. Summing up, the suspension system is of paramount importance for the motoring comfort, safety, and performance, soundly and well, for occupants as shown in Figure 3.5.

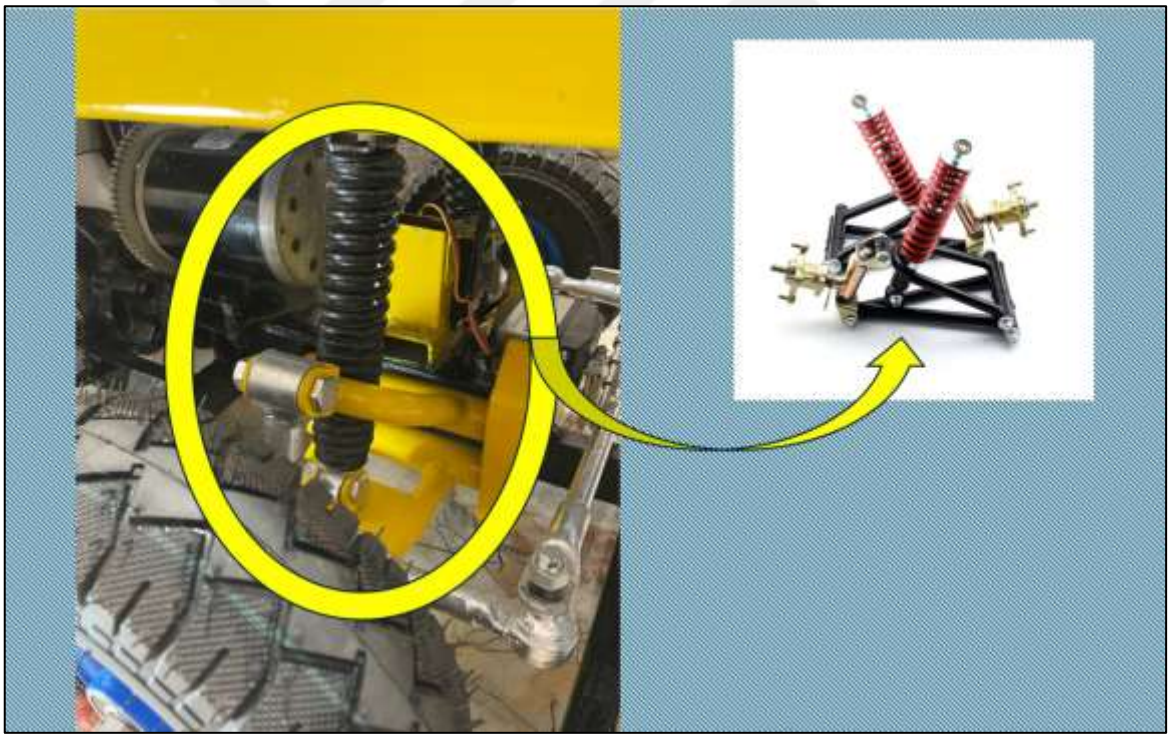

Figure 3.5: Suspension System Model.

3.4.2.3 Mechanism of braking system and ABS

This section demonstrates the process of modeling a simplified Anti-Lock Braking System (ABS) for a vehicle under hard braking conditions. The model focuses on a single wheel but can be replicated to represent a multi-wheel vehicle. Utilizing the signal logging feature in

Simulink, the model logs signals to the MATLAB workspace for analysis. The code for this process can be found in the appendix for reference.

The wheel speed is calculated in a separate model called sldemo_wheelspeed_absbrake, which is referenced using a 'Model' block. Both the main model and the referenced model employ a variable step solver, allowing Simulink to track zero-crossings in the referenced model. The friction coefficient between the tire and road surface, denoted as μ, is determined by an empirical function of slip known as the μ-slip curve. To create these curves, MATLAB variables are passed into the block diagram using a Simulink lookup table.

The model then calculates the frictional force (Ff) acting on the tire circumference by multiplying the friction coefficient (μ) by the weight on the wheel (W). This force is then divided by the vehicle mass to derive the vehicle deceleration, which the model integrates to obtain the vehicle velocity.

For the ABS controller, an ideal anti-lock braking controller based on a 'Slide Mode Controller' is employed. This controller operates on the error between the actual slip and the desired slip. The desired slip is set to the value at which the μ-slip curve reaches its peak, representing the optimal value for minimum braking distance.

The Simulink model of the proposed anti-lock braking systems has been simulated in two parts:

a. Wheel Speed Model based on the Slide Mode Controller (SMC)

This sub system will provide the control panel of the braking systems of our vehicle model, and this subsystem can be represented in MATLAB as show in Figure 3.6, and distributed blocks through this model build based on the specific properties of the proposed vehicle model.

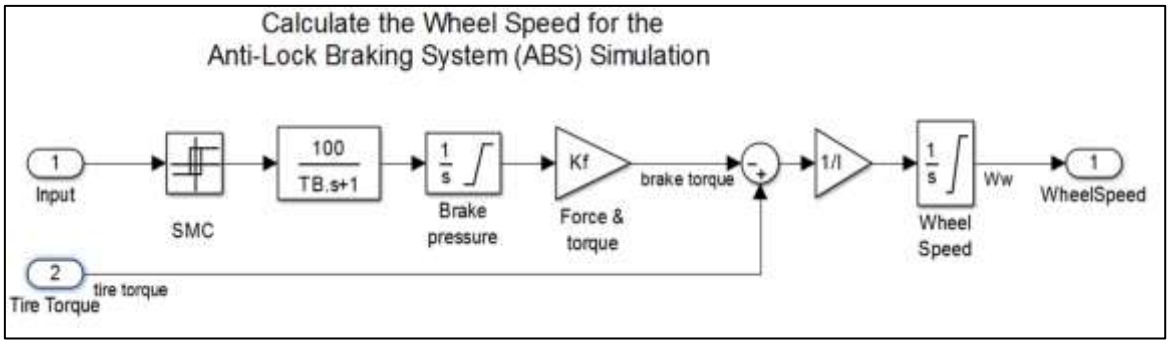

Figure 3.6: Proposed Wheel Speed Model.

Where the in ports (1, 2) well be connected to the desired slip ratio and tire torque respectively.

b. Anti-lock braking system

The complete model of braking system can be considered as shown in Figure 3.7 that equipped to wheels of vehicle.

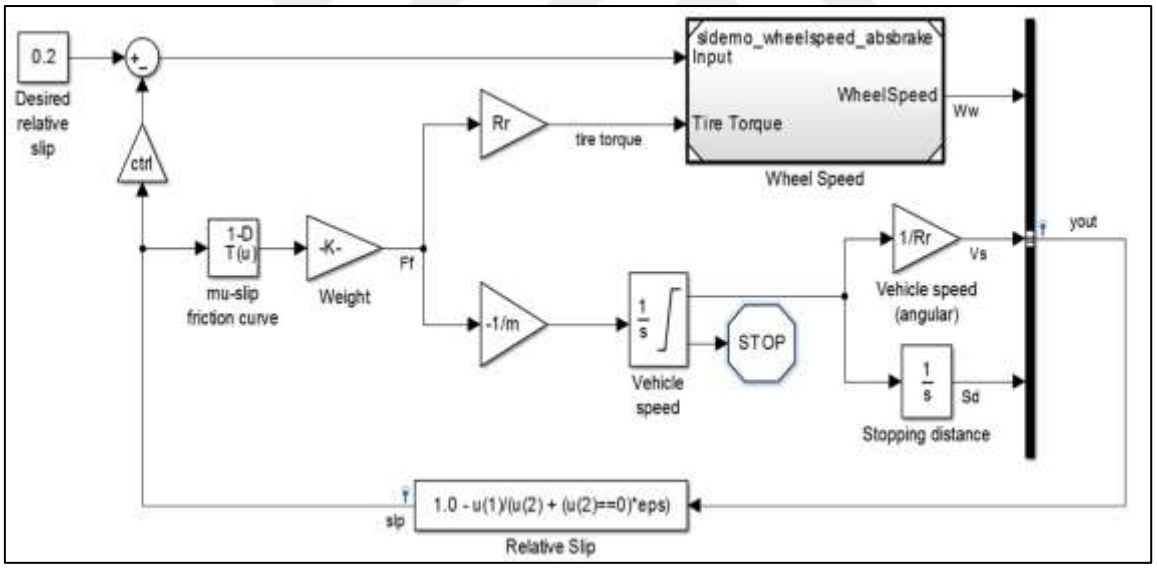

Figure 3.7: Proposed Anti-lock Braking System.

To regulate the brake pressure's rate of change, the model calculates the difference between the actual slip and the desired slip. This signal is then fed into a Slide Mode Controller (SMC), which outputs either +1 or -1 depending on the error's sign, as depicted in Figure 3.7. The on/off rate subsequently undergoes a first-order lag to account for the delay inherent in the electromechanical lines of the brake system. The model then integrates the filtered

rate to yield the actual brake pressure. The resulting signal, multiplied by the piston area and radius with respect to the wheel (Kf), is the brake torque applied to the wheel.

In this model, the frictional force acting on the wheel is multiplied by the wheel radius (Rr) to determine the accelerating torque exerted by the road surface on the wheel. Subsequently, the brake torque is subtracted from this value to obtain the net torque acting on the wheel. Dividing the net torque by the wheel's rotational inertia (I) yields the wheel acceleration, which is integrated to obtain the wheel velocity. To ensure both wheel speed and vehicle speed remain positive, limited integrators are employed in this model.

The next section of our content will look at how the braking system of the car works along with the Antilock Braking System (ABS). The braking system is an important mechanism which brings a car to a halt or making it slow down in a controlled way. It has its parts all working together to create heat through friction, which finally helps the vehicle to come to a stop. The vehicle's brake system has gone through an elaborate process of engineering aimed at providing effective control on the rate of deceleration and at the same time minimizing the risk of skidding on the road surface. Two electromechanical brakes, placed in the discs, are interconnected with the spinning axle of the rear wheels. This brake can be adjusted by adjusting the angle at which these discs engage with servo motors that have high torque features that are part of the controlling system. Besides, LM393 speed sensor types of rotation sensors have been implemented as encoders, so using mathematical formulas each tire speed will be programmable. This can be achieved by comparing the speed differential between two tires, and a slight deviation from equal speed indicates a potential sliding of the vehicle on the road. These triggers the ABS system that further controls the brake lock up using measured time intervals. This way, they vehicle’s trajectory and stop within the shortest length without skidding or swerving. A button has been added to the app that allows users to turn on or off the Anti-Lock Braking System (ABS). Adding this will be a great asset for running tests and reviewing the performance data with and without ABS turned on. By activating and deactivating ABS system researchers can get data regarding the efficiency of the system performance under various road conditions, it can be helpful in preventing the wheel lock-up and improving vehicle stability during emergency braking maneuvers as shown in Figure 3.8.

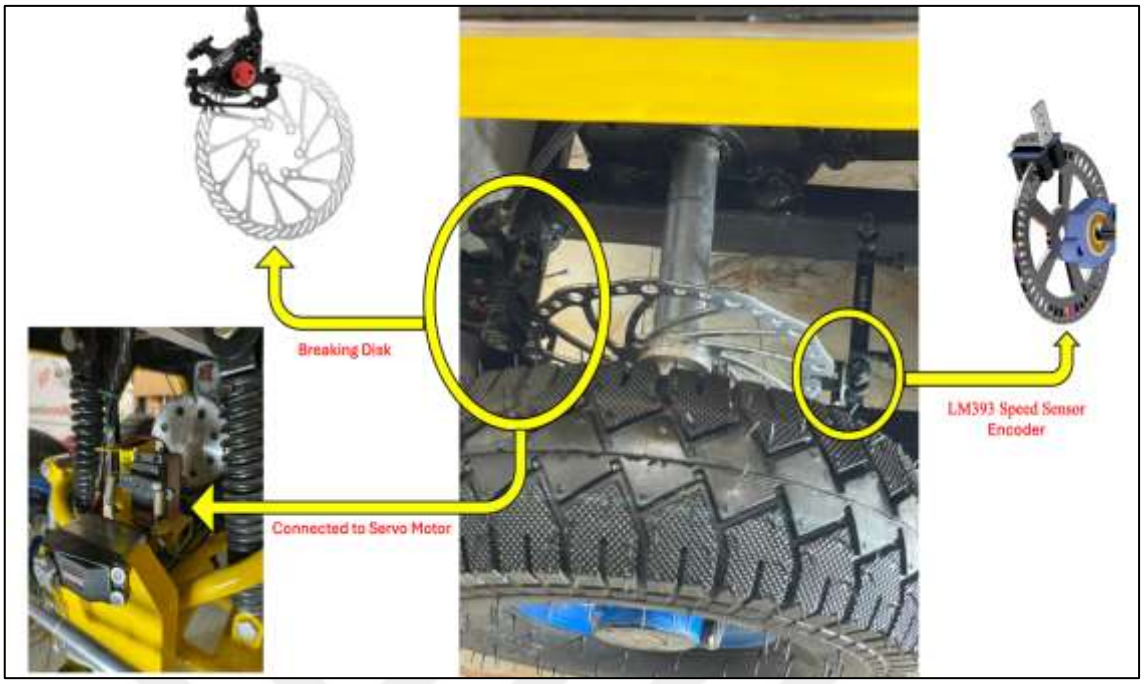

Figure 3.8: Braking and Encoding Systems.

3.4.2.4 Mechanism of steering system

The steering system of a vehicle is an important system that enables the vehicle to move in the required direction. Being composed of interconnected elements, such as steering wheel, steering column, steering gear and steering linkage system, the steering system translates the inputs of the driver into wheel movements. The consequence of the vehicle dimensions is the development of the advanced steering system which managed the angle of deflection of the front wheels. This programmable control of the wheel angle is achieved using a servo motor with a very high torque model (ROVAN-RS-2050D). This function allows the car to shift the track without any effort. The angle of wheel deviation is controllable through mobile application buttons where user guidance is simplified, or the vehicle is programmed to autonomously perform this task. The angles are transmitted to the servo motor after path planning and obstacle detecting to define the path of the vehicle using pre-set algorithms as shown in Figure 3.9.

Figure 3.9: Mechanism of Steering System.

3.4.2.5 Motor and motor driver

Due to the vehicle's weight exceeding 50 kilograms, it necessitates a high-torque motor capable of propelling it in both forward and reverse directions with ease. Therefore, the MY1020 motor type has been selected. This direct current (DC) motor operates at a voltage of up to 36 volts DC and has a power output of up to 1000 watts, with a maximum rotational speed of 3000 revolutions per minute (RPM). The MY1020 motor's high torque output ensures efficient propulsion of the vehicle, enabling smooth movement in various conditions and terrains. Its robust design and performance characteristics make it well-suited for applications where heavy loads and demanding torque requirements are prevalent. The motor's versatility and reliability make it an ideal choice for powering vehicles, offering the necessary power and torque to propel the vehicle effectively in both forward and reverse directions. So far, the issue stemming from the vehicle's high weight has not been resolved. To address this, the motor is connected to a gearbox, which, in turn, links to two axles responsible for transferring motion from the gearbox to the wheels. This setup helps alleviate the load on the motor and ensures smoother movement. Excessive load can lead to high current draw, potentially damaging electronic circuits, particularly the motor driver circuit. It can also cause wire failures and increased energy consumption. Therefore, it was imperative to resolve this issue. Additionally, a larger gear wheel was connected instead of

the one initially attached to the motor to increase speed. Although increasing the wheel's diameter reduces the motor's rotational torque, the gearbox compensates for this, ensuring efficient performance.

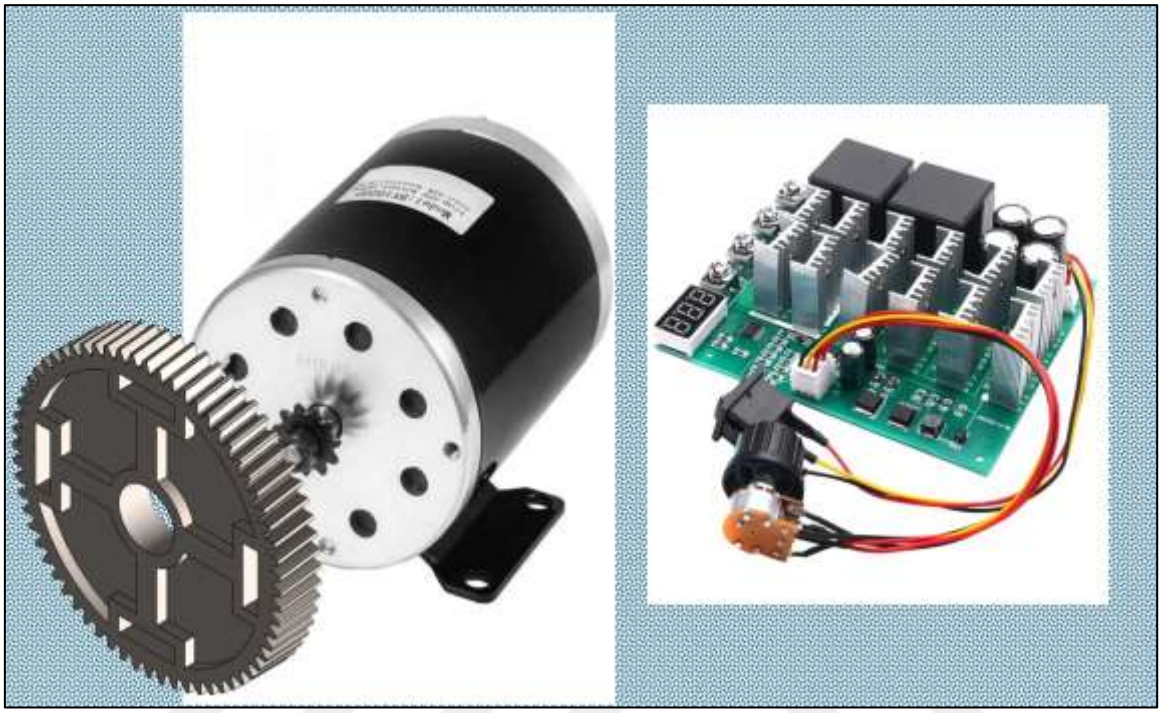

Figure 3.10: MY1020 Motor Type with Motor Driver.

Moreover, a DC motor driver of the [302-1093 10-55V 12V 24V 36V 48V 55V 100A Motor Speed Controller PWM Reverse Control Switch with LED Display] was used. The motor speed controller uses the technology that adjusts the voltage and current input that is given to the motor which helps in the smooth functioning of it. This part of the control circuit serves as an intermediary between the microcontroller or control system and motor, providing the necessary torque and direction as required by the microcontroller or control system. Besides the motor speed controller, the protection features such as overcurrent protection and the feature that stop current if the motor is too hot are also provided to prevent motor and other parts of the circuitry from damage. Its working in the right way is highly important responsible too for better work performance and reliability of the driving system. One of the major issues in integrating motor speed control circuit with Arduino is the lack of compatibility, because it is not prebuilt analog circuit. An electronic speed controller of a motor is usually controlled separately via special command to manage the speed and direction of an engine's movement. While this was the most complex challenge, Arduino

interface was complicated by the protocol mismatch and different requirements for signals. Addressing this difficulty meant building an approach that will allow to translate the control signals which creates by the Arduino into an appropriate format that will be recognizable by the motor speed controller. To address this issue, the motor speed controller was equipped with two types of electrical switches: Varying the frequencies and the duty-cycle was rather simple and gave us interesting responses and the other for selecting the direction of rotation. One switch facilitates clockwise rotation, while the other enables counterclockwise rotation. This mechanical setup allows the vehicle to move both forward and backward. The electrical switches were modified by disconnecting them and connecting the wires to electrical relays controlled by signals from the Arduino. Two relays were dedicated to determining the direction of rotation, while another relay was used to turn the speed controller on and off. Additionally, an emergency button and an external control switch were incorporated to allow manual intervention in emergency situations or instances where control over the vehicle is lost. This integration ensured that the motor speed controller could be seamlessly controlled and monitored within the vehicle's system, enhancing safety and functionality. Another challenge we encountered was controlling the motor's rotational speed and, consequently, the vehicle's speed. The motor speed controller we used was equipped with a variable electrical resistor. This resistor determined the amount of electrical current (through PWM - Pulse Width Modulation) flowing through the motor, hence regulating its speed. However, controlling this variable resistor required precise and effective adjustments to achieve the desired motor speed and, consequently, the vehicle's speed. It's crucial to note that speed control needs to be done programmatically via Arduino, either through a mobile application or directly by the vehicle's system. To address this challenge, the variable resistor was mechanically linked to a small servo motor. This small servo motor rotates the resistor based on the angles it turns, requiring calibration of the servo motor's rotation angles with that of the variable resistor as depicted in Figure 3.10.

3.4.2.6 Sensor’s integration

The ADAS (Advanced Driver Assistance System) is one of the most complex and modern systems that has become the focus of researchers and automotive manufacturers alike. To implement an environment where a vehicle exhibits ADAS features, numerous sensors are required. Among these sensors, one of the most crucial is the LiDAR sensor. The chosen model, RPLIDAR (C1M1-R2), stands out from other rotary LiDAR sensors due to several

distinguishing features. This type is characterized by The RP LiDAR C1M1-R2 Portable ToF Laser Scanner Kit offers a comprehensive 360-degree 2D laser scanning solution, featuring high-speed rotation and a sampling frequency of 5000 times per second. It considers not solely positioning, but also the reflectivity data and the 2.5 dimensionality information. With the outstanding measuring distance of up to 12 meters and an almost invisible blind range of just 0.05 m, it is very successful in immediate obstacle avoidance. The miniaturized and adaptable design as well as being complied with the Class 1 Laser Safety Standards guarantee safety and ease of application integration. In home robots to commercial vehicles, the RPLiDAR C1M1 delivers the reliable performance and adaptability suited for an extensive range of uses. The lidar is designed to be placed on the top of the vehicle, above the main electrical control box. It scans the whole area around the vehicle and avoids any visibility obstacles. Nevertheless, the lidar still has a challenge to detect obstacles within the line of sight which may miss low-lying obstacles that can be risky for the car. In order to solve this problem, the car is fitted with six ultrasonic sensory placed respectively around all its sides. A sensor is put on the front center and another on the rear center while four sensors are to be distributed one each side at the front and rear. Therefore, it gives a chance to detect obstacles in the area easily and to handle them ahead of time. Moreover, it has been used an OV7670 Arduino compatible camera whose characteristics have been covered earlier in Chapter Two. It is mounted on the vehicle's front. We will discuss its functionality in machine learning components when we use it in data collection.

The location of the car is found using a Ublox NEO-6M GPS Module, which is based on getting the position from satellites. This module is an integral part of robot's functioning which yields the precision in locating the robot. A shield on the Arduino Mega board is used for an interface between the Ublox NEO-6M GPS Module and the Arduino Mega. The shield reduces the complexity of the connection process and allows D-Line/UART switching. Enabling "D-Line" mode on the GPS module works well by connecting the module to digital pins 2 and 3. This is done through seamless integration into the design to avoid conflicts with other components. In order to achieve accurate positioning and orientation, the system contains HMC-5886l digital compass. Besides the head-up display, the compass also has advanced algorithms for computing the destination viewing angle and features 2-axis magneto-resistive sensors. Its electronics and analog and digital support circuits create a system that is accurate and reliable. In between the robot and the base station unit a

communication is performed using the Bluetooth wireless module. This module provides data and command link between vehicle telemetry and ground station control, thus facilitating situation awareness during operations as depicted in Figure 3.11.

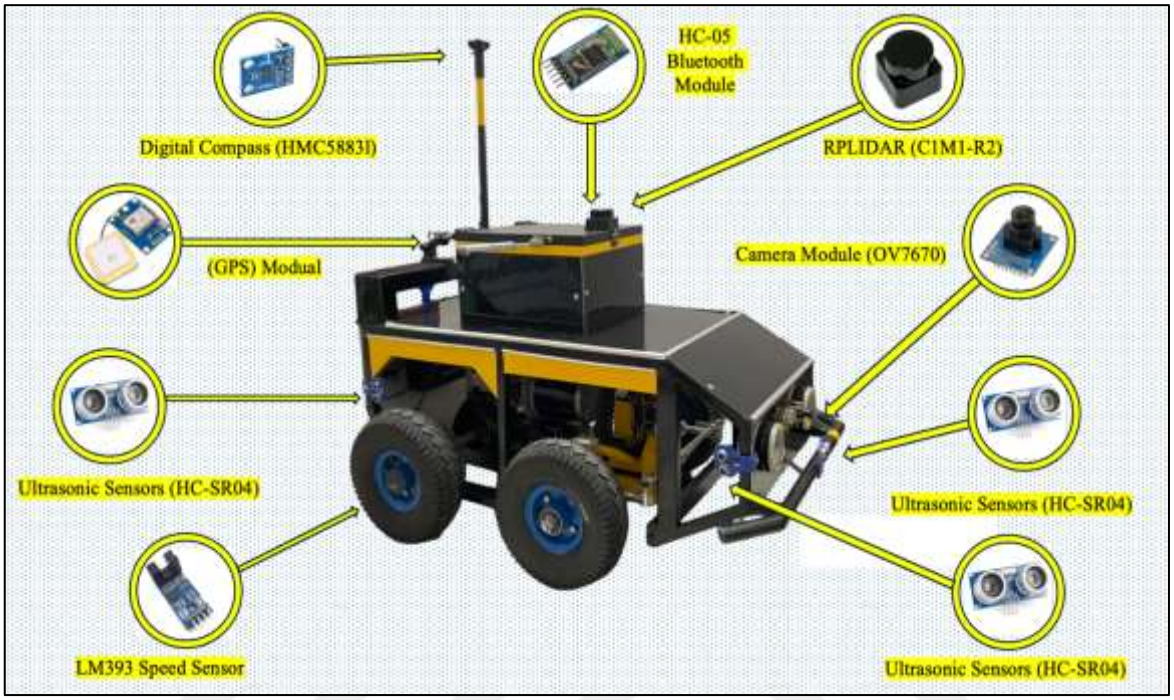

Figure 3.11: Vehicle Model with Sensors.

3.4.2.7 Power supply system and battery charger

In a vehicle integrated with an ADAS system, the power supply system and battery charger are among the key part. With this system in place, all electronic components can be operated in steady manner and provide the required power for the vehicle's systems. The main power consists of the battery as the main source of electric power for the car. We used two of 12V 12AH battery banks and connected them in series for our case. Therefore, the total voltage will be 24 Vdc. while the battery charger serves as the source for the stored energy in the battery. It is made up of a charge cable that is linked with an outer power source like a wall outlet or a charging station. The charging unit regulate the charging current and voltage thus the battery is charged to the maximum level without damage.

3.4.2.8 Vehicle control

The Arduino Mega 2560 was the CPU for the robot selected. This microcontroller board is based on ATmega2560 chip and a lot of features to be built in such a way to make it a great

device. The ATMega2560 board has 54 digital input/output pins, of which 15 are PWM outputs. The board also comes with 16 analog inputs, which provides extensive connection options Apart from this it also has necessary components like a power jack, reset button, a USB connection, 4 UARTs (hardware serial ports), and a 16 MHz crystal oscillator supplied, hence giving a comprehensive prototyping environment for microcontroller-based projects. On the other hand, Arduino Mega 2560 was manufactured to functionally communicate with several third-party shields which are well supported by the family of Arduino Uno and Arduino Duemilanove. It has a physical character that can wear out a lot of different types of shields thus increasing its versatility and expanding its applications. The Arduino Mega 2560, which can be powered by a 5V DC power supply and conforms to the typical microcontroller board specifications, is used in this project. Yet notably, the greatest current output that can be drawn from the Arduino pins is only 40mA, which is why care must be taken while connecting to other external devices or circuitry that might require more current for proper operation as shown in Figure 3.12.

Figure 3.12: Vehicle Main Control Circuit.

3.4.2.9 Communication modules

To facilitate communication between the robot and the base station unit (mobile phone), a Bluetooth wireless module is employed. This module serves as a conduit for the exchange

of data and commands between the two entities. The setup for establishing this wireless communication link does not entail any intricate configurations; instead, it allows for straightforward one-on-one communication between the robot and the base station unit.

3.4.2.10 Horn and indicators

A warning alarm (Horn) has been added to the vehicle by connecting it to an electrical relay for programmable control through signals from the Arduino. This alarm can be activated manually through a dedicated button within the mobile application or automatically when the vehicle reverses. In the event of no obstruction behind the vehicle, the alarm operates intermittently. However, if an obstacle is detected, it operates at longer intervals until the vehicle comes to a complete stop or changes its path away from the obstacle. Additionally, two illumination spotlights have been installed, each operating at 12 volts DC. Each spotlight features two brightness settings, which are also connected to two electrical relays. These spotlights can be turned on or off using buttons within the application as well. Finally, two turn signal lights have been added, positioned inside the front bumper of the vehicle. Each of these lights is connected to a dedicated electrical relay for individual control. These lights are activated when a vehicle is detected behind, serving as a warning to indicate the intention to change lanes. They operate as an alert when the rear blind-spot sensors detect vehicles, helping to prevent accidents.

3.4.3 Vehicle Control Using Mobile Platform

Since the vehicle unit's main controller is the mobile phone station an application is made by Android Environment Program and the responsibility of this App is divided into three tasks. The first task is to get the map points of selected areas, the second being to calculate the path for the mobile robot using A* algorithm and converting it to a series of Latitude / longitude points. The third task for the App is to transmit the path to the mobile robot. Furthermore, both brake forces and total control of the mobile robot can be done manually using buttons on the Mobile Phone Station and the scheme of wiring diagram of our vehicle model is depicted in Figure 3.13.

Procedure Steps:

a. The mobile station is connected to the vehicle.
b. Compass HMC5883l is used as the base to set Heading for the vehicle.

c. Compass HMC5883l is used to set calibration for the vehicle.
d. GPS Module is used to obtain the GPS location of the Vehicle.
e. Multiple target's locations are selected, and locations saved on phone.
f. After crate a path, apply A* Algorithm to reach to targets.
g. Apply collision avoidance algorithm along with path.
h. ABS for braking system is applied.
i. Send path to the vehicle via Bluetooth.

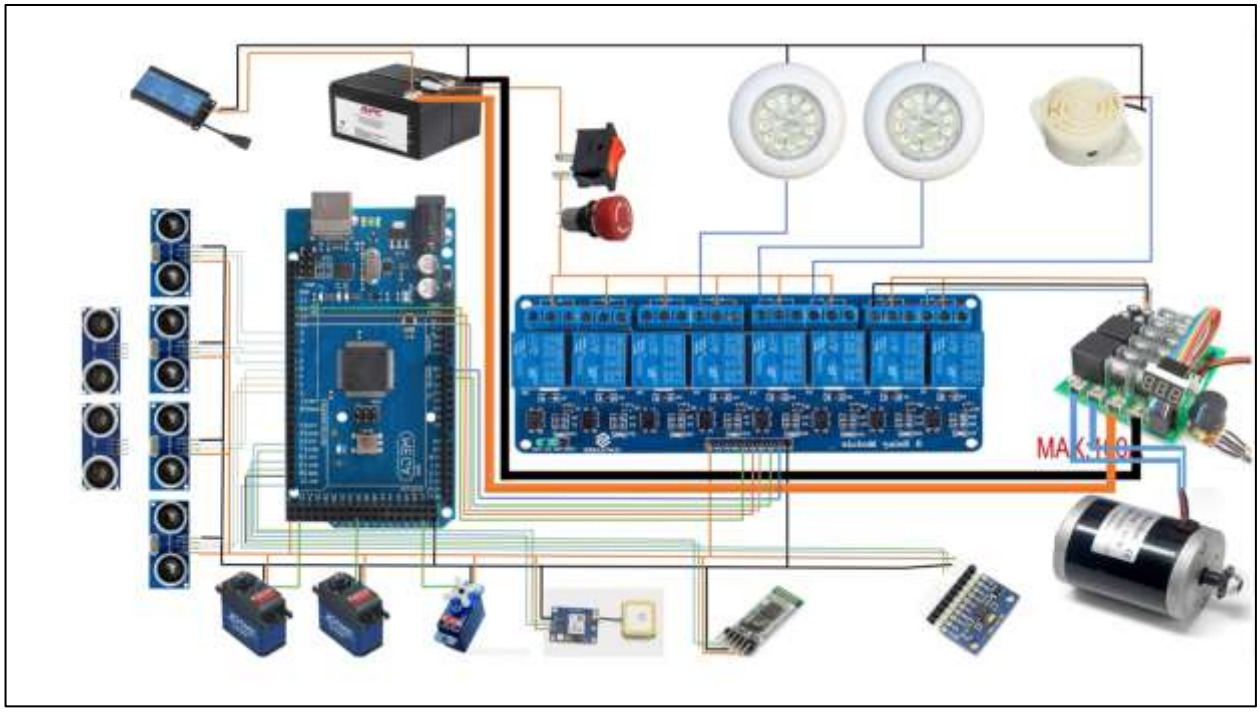

Figure 3.13: Hardware Wiring Connection.

3.4.3.1 Mobile phone application for vehicle control

The Mobile phone Station uses an Android application that was developed with the use of an open-source web application originally owned by Google and maintained by Massachusetts Institute of Technology (MIT) also known as MIT App Inventor [26]. Building an android app is simpler when you use MIT App Inventor. With this technology you can easily develop any Android app with little or no experience as a programmer. This is a very effective way of developing an Android application. You don't need to install any heavy software in order to achieve this because it can be done over the web browser with your Google account. Listed below are the three aspects of building an application online:

a. Application inventor designer.
b. Application inventor block editor.
c. An emulator or android phone.

Figure 3.14 shows these three aspects of building the mobile phone station application.

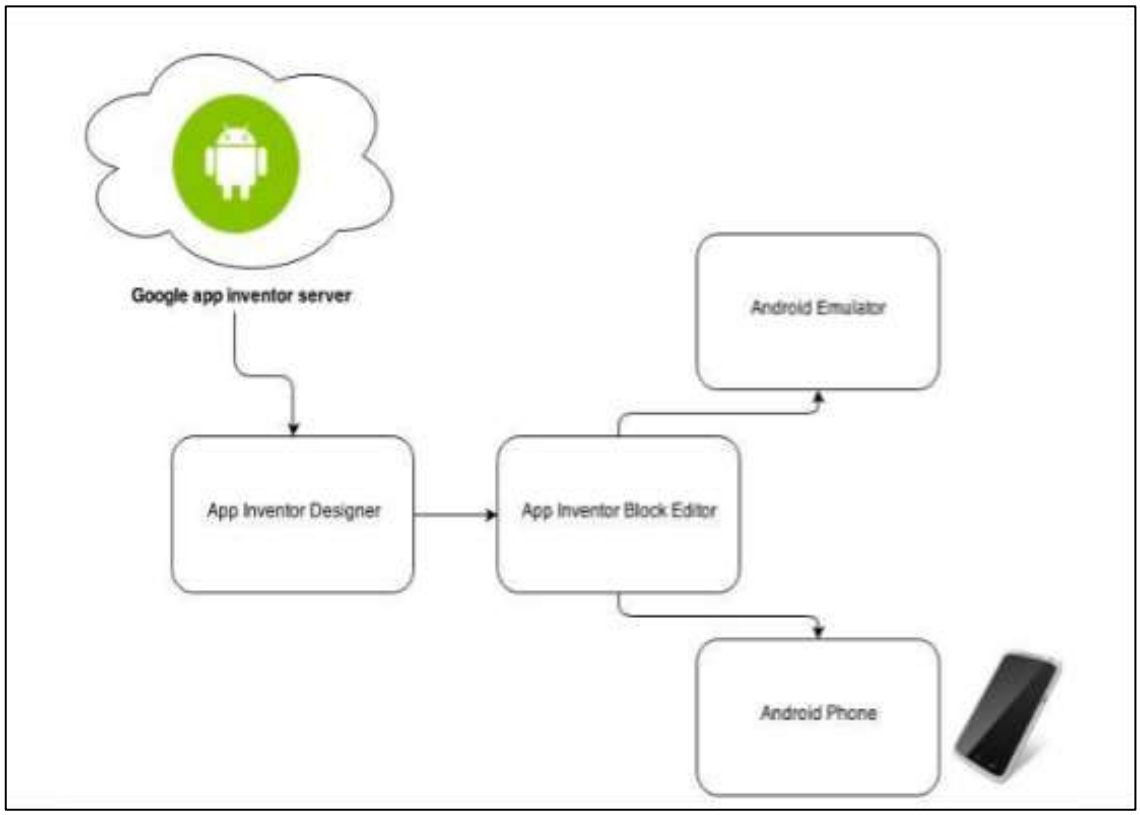

Figure 3.14: Aspects of Building Mobile Phone Station **[27]**.

Firstly, the use of the application inventor is the first process of developing an Android application for controlling the vehicle. All the variable ingredients are present on the web page of MIT App Inventor 2 which is used to design the proposed application. The ingredients required for this development are screen, button, textbox, image and many more. As illustrated in Figure 3.15, we can see that the process of developing this application is simply a drag-and-drop process. In the application are one button used for Bluetooth module connection and another button used in connecting the speech recognizer. The description of these buttons in the block editor Table 3.1.

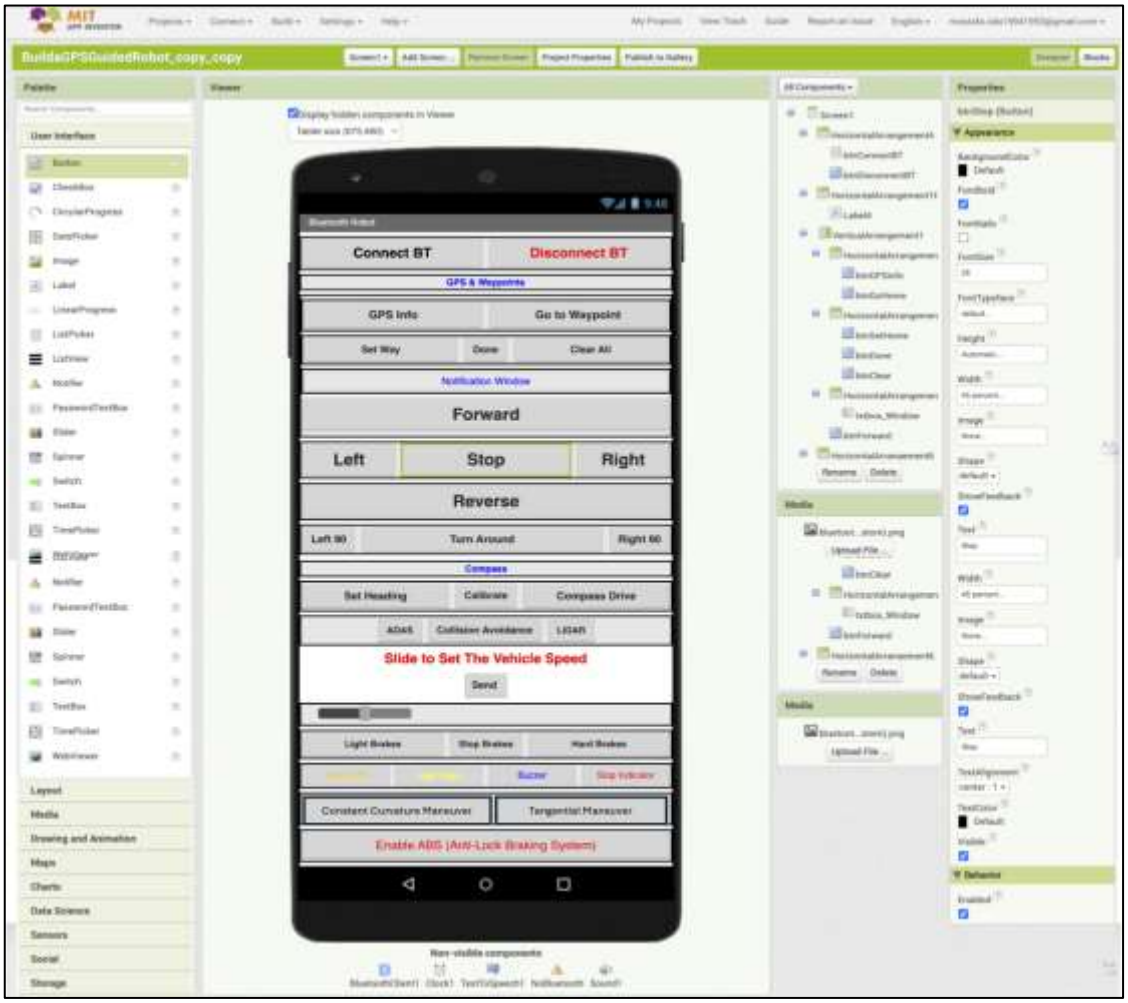

Figure 3.15: Screen Design of Application.

Each button and then edited individually according to the functions they are to be programmed for. All codes that are related to this development can be found in the appendix of this work.

Table 3.1: Buttons Description.

Description	Setting	Name of Button
Forward Car	Send text "1" to the robot	Forward

Table 3.1: Buttons Description “Table Continued”.

Backward Car	Send text "2" to the robot	Reverse
Steering Left	Send text "3" to the robot	Left
Steering Right	Send text "4" to the robot	Right
Stop Car	Send text "5" to the robot	Stop
Set Waypoints	Send text "6" to the robot	Set Home
Go to Waypoint	Send text "7" to the robot	Go Home
Turn Car Around	Send text "8" to the robot	Turn Around
Drive Car Using Compass	Send text "9" to the robot	Comps-Drive
Set Car Heading	Send text "10" to the robot	Heading
Read GPS Location as Lat and Long	Send text "11" to the robot	GPS info
Turn Car to 90 Right	Send text "12" to the robot	Turn Right
Turn Car to 90 Left	Send text "13" to the robot	Turn Left
Do Calibration for Compass	Send text "14" to the robot	Calibrate
Turn ON-OFF Sensors	Send text "15" to the robot	Ping
Clear Waypoints	Send text "16" to the robot	Clear
Save Waypoints	Send text "17" to the robot	Done
Enable ABS	Send text "18" to the robot	Enable ABS
Enable Light-Brakes	Send text "20" to the robot	Light-Brakes
Enable Hard-Brakes	Send text "21" to the robot	Hard-Brakes

Table 3.1: Buttons Description “Table Continued”.

Speed Control for Car	Send text "1155-1250" to the robot	Slider
Activate Advance Driver Assistance System	Send text "22" to the robot	ADAS
Activate LIDAR	Send text "23" to the robot	LIDAR
Activate Low Light	Send text "24" to the robot	Low Light
Activate High Light	Send text "25" to the robot	High Light
Activate Buzzer	Send text "26" to the robot	Buzzer
Constant Curvature Maneuver	Send text "27" to the robot	Activate First Manoeuvre
Tangential Maneuver	Send text "28" to the robot	Activate Second Manoeuvre

3.4.3.2 Vehicle side software

The vehicle is designed to carry our three tasks. The first is to receive path from the mobile application, the second is to detect and avoid obstacles and the third is to activate the ABS (anti-lock braking system) during the braking points. The robot will receive the path to follow from the mobile phone in forms of number points and each one of these points will be considered a target point that will be numbered T1, T2…Ti…., Tgoal. For each iteration I, the robot finds the distance and the angle bearing between the current position that is received from GPS and the target point (Ti) according to ‘haversine’ formula equation (3.1) and bearing formula (3.4) [13].

$$\alpha = sin^2\left(\frac{\Delta lat}{2}\right) + cos(lat_1) * cos(lat_2) * sin^2(\frac{\Delta lng}{2}) \tag{3.1}$$

$$\emptyset = 2 * tan^{-1}(\frac{\sqrt{a}}{\sqrt{1-a}}) \tag{3.2}$$

$$d = R_e.\emptyset \tag{3.3}$$

$$\emptyset = tan^{-1}\begin{pmatrix} sin(\Delta lng) * cos(lat_2) * cos(lat_1) * sin(lat_2) - \\ sin(lat_1) * cos(lat_2) * cos(\Delta lng) \end{pmatrix} \quad (3.4)$$

According to the formula above, *a* is the square of half the chord length between the points, lat is latitude, lng is longitude, d is distance, ∅ is the angular distance in radians, *Re* is earth's radius, θ is bearing angle.

After the mobile robot reaches the target point Ti then the next target point of path Titl will be taken, and the robot returns the same operation and so on until the end of path. To make vehicle detects and avoids obstacles, four Ultrasonic sensors are used as explained before in hardware design section. The mobile robot is programmed to detect obstacles at 3 m or less away. When mobile robot tries to move forward, if any obstacle prevents its movement it tries to turn its direction depending on the front left and right sensors, if any obstacle is on the right it tries to turn left and if any obstacle in left it tries to turn right, this turning on is a continuous process before frequently moving forward to the goal.

3.5 DESIGN OF OBJECT DETECTION AND TRAFFIC SIGN RECOGNITION SYSTEMS USING MACHIN LEARNING

3.5.1 Object Detection using LIDAR

In this section of the thesis, machine learning techniques were utilized for terrain obstacle avoidance during the vehicle's journey from the starting point to the pre-defined destination, which was stored and identified using GPS coordinates. Additionally, these techniques were employed during maneuvers executed for conducting various tests on the vehicle. Here, we will delve into the explanation of the fundamental components utilized in machine learning, including data acquisition, data preprocessing, training the classifier, and running the machine learning classifier on the Arduino platform.

3.5.1.1 Data acquisition

In this stage, all dataset was manually collected by driving and controlling the vehicle through a pre-designed and implemented mobile application. The vehicle was connected to the mobile phone via Bluetooth, where buttons within the application allowed the user to manually control various directions (forward, backward, and steering angle) and other

properties, as previously explained in this chapter. The collected data represents distances between the vehicle and obstacles or between the vehicle and road edges. RPLIDAR and Ultrasonic sensors were used for this purpose, with the LIDAR sensor placed at the top of the vehicle and the ultrasonic sensor at the bottom. Six of these sensors were used to enhance their ability to sense distances to obstacles that might be invisible to the LIDAR sensor, as the latter is two-dimensional (2D). The vehicle was driven on 11 different tracks. Each file corresponds to a session of data collection, involving 10 to 30 minutes of driving on the racetrack. Most files contain a blend of data captured from both clockwise and counterclockwise driving maneuvers, and the data, including distances and actions taken by the mobile application such as directions (forward, right, left, forward right, and forward left), were printed via the Arduino serial port using MSCODE with Python extension. This data was stored in an external memory connected to the Arduino, and data from each track was collected in a separate CSV file.

3.5.1.2 Data pre-processing

Data preprocessing plays a vital role in the machine learning pipeline by being a set of necessary jobs which main purpose is to refine a dataset in order to improve the results of the training and the evaluation processes. Library utilization was our strategic approach for library processes that had to be streamlined in our machine learning project. Of these libraries, the main one of the bunch (NumPy) has precisely been the most critical, providing the basis of meaningful advanced scientific computations and complex mathematical operations which are important for data handling. While Panda witnessed the eclipse of its usefulness, it scored a monumental entry into data analysis and manipulation field, crafting a group of features that functioned seamlessly to perform numerous structured rearrangement tasks, and thus speeding up our workflow. On the essential part of feature selection, we leveraged the f_classif and SelectKBest functions of scikit_learn to identify the responds data with the aid of a statistical criterion. Our dataset partitioning strategy stretched out into limelight with the incorporation of train_test_split function where the dataset was equally divided into two unique parts, namely training and testing datasets, for a greater model accuracy. On top of that, Label Encoder by scikit-learn enabled representation of categorical labels as numbers. In this way, we reached maximum compatibility of the models and their effectiveness. On the side of classification, Random Forest Classifier tested itself to be our model of choice, with unequalled ability of finding

subtleties and thus excelling in assigning a sought-after class to a pattern fitted in. To measure the performance of our models, we mostly relied on sklearn accuracy_score and classification_report functions, which give you highly detailed understandings of how your model is performing and a variety of metrics such as accuracy that will help you make informed decisions. Besides model development, the micromlgen library has also become a crucial piece, linking trained models to scikit-learn models for C code generation, making the task of getting machine learning models working on microcontrollers a lot easier. Within the scope of our journey through machine learning, collectively, the abov-mentioned libraries played the crucial role across various stages, from initial data preprocessing to model evaluation and deployment. The next step was data loading from the txt file, and we used pandas module to transform CSV files into a structured Data Frame format using the panel's features. This dataset had different subjects and each subject had exactly 241 columns. The fourth column contained both LIDAR and Ultrasonic measurement and the fifth column was the categorical label generated from a single letter. They became critical as they formed a string of directions like Forward, Right, Left, Forward right, Forward left. These commands were further used as inputs for supervised learning and evaluation tasks. Moreover, we delved into data cleaning, underscoring the cardinal principle in data science: "junk in, junk out." We started being extremely careful cleaning of the datasets by, for example, changing the end column to "label" for more readability and removing the samples of "L", "R", "H" and "J" to simplify further classifying tasks. These meticulous ways tried to raise model sturdiness and efficacy by homing in the directions which facilitated the proper movement of the racetrack. Besides that, the matching data should be selected with particular attention paid to extracting key features which are important for a highly performing model. Using the “SelectKBest” from scikit-learn library, we kickoff the feature selection process, intentionally reducing the number of features and keeping only the ones that are relevant and to the model generalization and performance optimization. Consequently, this process which is highly data dependent and is designed to make model application as effective as possible while minimizing the strain of it computationally, has become our most important pillar in model development.

3.5.1.3 Training the classifier

Training the classifier is one of the most important processes of our machine learning journey which is the result of various preprocessing data methods we have undertaken. Through the

library of rich Python modules, the process becomes an effortless, smooth and uncomplicated task. Whether or not the classifier we have selected is effective or not depends on the quality as well as authenticity of the dataset being used which demonstrates the importance of a thorough data preparation process. After the data preprocessing stage, we dive right in by training our classifier, a process prompted by the number of libraries available in Python. The accuracy test is a quality-check of our model's predictive power, with the accuracy being the key metric measure to achieve. Having done the training, we evaluate the performance of the classifier using the test set, always aiming at the attainment of acceptable accuracy to strengthen the model's effectiveness. the accuracy was as high as 75% that we achieved through the classifiers we used. Although it might not be the way things are expected to be and likely can be improved, but this rate of accuracy is good enough to let the car drive itself along the racetrack. In spite of some occasional collisions involving the wall, the robot also exhibited some spot of impeccable maneuvering often navigating for long distances without any issues. For the training process we used the Random Forest Classifier, which is an ensemble model with a trusted versatility and robustness. The model was trained with the max depth equal to 3 and the random state of 42 using the curated dataset that includes only the most relevant features chosen because of a scrupulous statistical analysis. Random Forest Classifier was the best tool among others in several experiments. The classifier was trained with the set of test values and the accuracies computed as well as the comprehensive classification report generated to describe the accuracy level for all classes. This detailed assessment indeed, helped to reveal crucial aspects of classifier’s performance, thus providing an opportunity for the further refinement of the model and optimization of its parameters. While the implementations of machine learning algorithms might be complex at the core, the training process was simple and quick, due to the comprehensive libraries in Python. The blending of data preprocessing, model training, and evaluation highlighted the advantages of our method as well as its universal applicability, generating, in the end, a conclusive classifier that could face real-life racetracks with complete autonomy. Alongside model training, we also utilized micromlgen to translate our applied Random Forest Classifier model into C code, which made it simpler to deploy it onto microcontrollers with limited resources. The significance of this point in time was that machine learning could be now blended smoothly with embedded systems thus ushering a new era of smart robots and automation devices.

3.5.1.4 Running the machine learning classifier on the arduino

To run a machine learning classifier on an Arduino board, the code needs to be change from the version written in Python to a version written in a C-language that supported by the Arduino. The compiling procedure follows the use of the micromlgen library that changes the machine learning model from Python to C, a programming language for Arduino execution. Through assembly, we're able to transform the learned model such that its machine was readily compatible with the few processing and material resource abilities of the Arduino device. After making the reference tutorial, the code consequently saved as .h file. It is essential to balance the datasets in terms of the dimensions of the training and classification data set for the model performance. Finally, after the code is uploaded it into the Arduino, the classifier shows that machine learning is possible on Arduino board. It clearly shows that the Arduino board can perform machine learning.

3.5.2 Traffic Sign Recognition

3.5.2.1 Traffic sign recognition overview

Recognition of traffic signs play a significant role in the work of ADAS and self-driving cars as they allow to interpret traffic rules and regulations. This procedure conducts the detection, classification, and interpretation of various types of traffic signs, including speed limits signs, stop signs, yield signs, and a lot more. In this section we were able to use python with CNNs for traffic signs recognition based on a deep learning approach. It will be the model which will be learning about different kinds of traffic sign features and patterns while being trained on a large dataset of traffic sign images. The project will incorporate the TensorFlow and Keras libraries to create and train the CNN model. Therefore, we will be training the dataset in the framework of the German Traffic Sign Recognition Benchmark (GTSRB), because there is a lot of different images of traffic signs for various classes. The project will take its course following the steps of data preprocessing, model architecture designing, model training and evaluation. The end evaluation will be conducted based on the metrics of accuracy, precision, recall and F1-score. Project aim is to build the strong and precise system of traffic signs which can be a part of autonomous driving, this is to ensure that the robots' driving scheme is safe and efficient on the road.

3.5.2.2 Traffic signs dataset

This project was done using the German Traffic Sign Recognition Benchmark [28] which is used for training an algorithm recognition model and to test its performance. This dataset was presented for the first time in the 2011 International Joint Conference on Neural Networks (IJCNN) [29]. Our dataset consists of almost 50,000 images of traffic signs divided into 43 classes as shown in Figure 3.16. The traffic signs classes encompass various common signs on roads. The NLR traffic sign detection benchmark contains 900 images of traffic signs, while the GTSDB contains 7000 images of traffic sign. This dataset is strongly characterized by the comprehensive nature of the dataset, together with its massive size and multiplicity of classes, which turns it into the best one for research into development of traffic sign recognition systems.

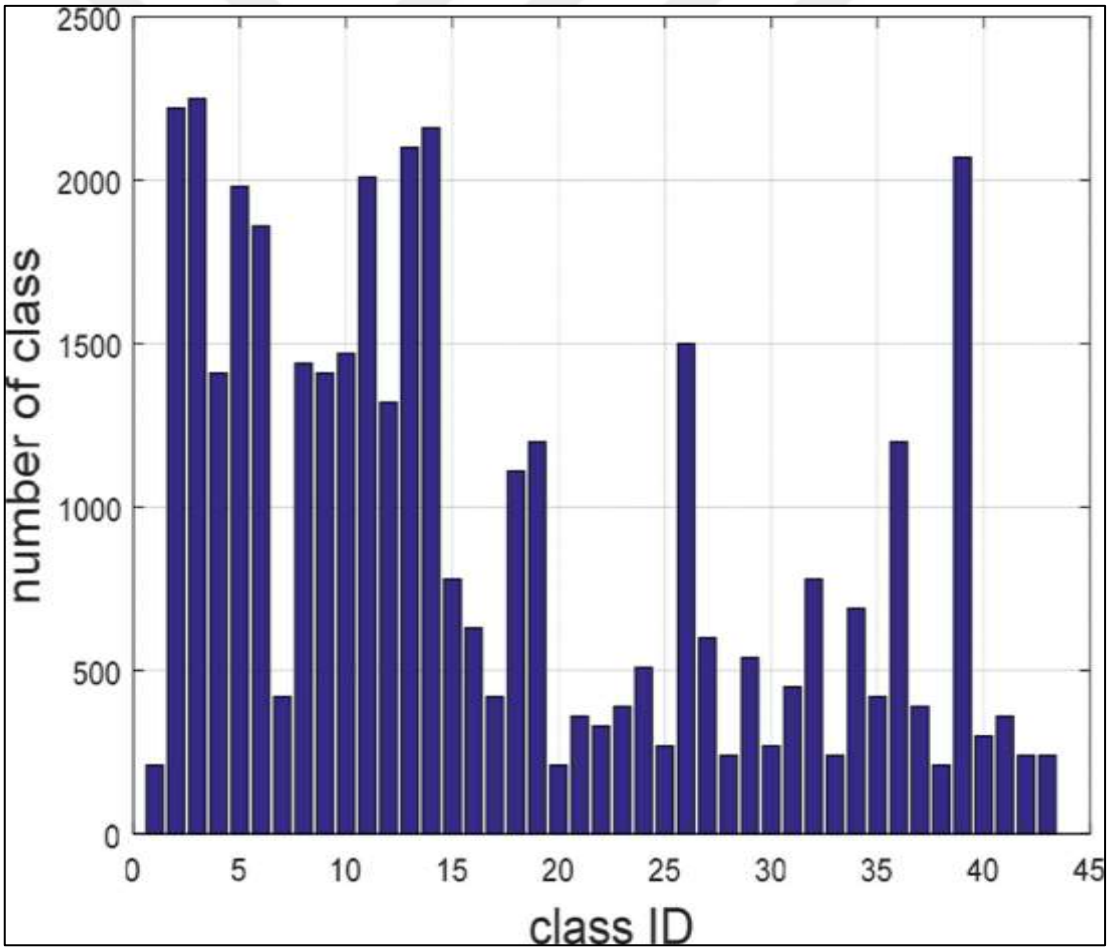

Figure 3.16: Number in Each Class of GTSRB Dataset [28].

The GTSRB dataset is divided into three main parts: training, testing, and validation. We have a training dataset of around 39,209 images, while the testing dataset includes about

12,630 images. The validation dataset is usually used for fine-tuning of the model during training and often is a tiny subset of the training dataset. The size of the image varies from 15x15 to 250x250 pixels and has RGB format of a colored picture. It can be said that the dataset is well-balanced, it means that each class has nearly the same number of examples, as it provides the model a fair chance of not being biased towards any particular class. The data set too has a CSV file that details more about the Image record such as the class label, file path and other metadata. Such data is used to pre-process and load the images into the model before model is fed with the images during training and testing. Overall, the GTSRB dataset is a clear choice for the training and evaluation of traffic sign recognition models because of its wide range of coverages and well-annotated collection of traffic sign images, making it an ideal choice for training and evaluating traffic sign recognition algorithms. Figure 3.17 **Random Representations of the 43 Traffic Signs [28].**

Figure 3.17: Random Representations of the 43 Traffic Signs [28].

In addition, I just apply Google Colab if I have a notebook to work with. The dataset was put in my directory which was in Google Drive. I could easily use the command 'drive.mount' to get access to it.

3.5.2.3 Prerequisites

During our project setups, we focus on the libraries that meet our individual requirements. E.g. such basic libraries as Numpy, Pandas, OS, and Matplotlib are vital for major data manipulation, file handling, data visualization purposes. Lastly, we make use of cv2 which is a robust library that has proven its mettle as a potential solution to various computer vision challenges. This library developed by OpenCV group offers a wide range of algorithms for image and video processing, feature detection, object recognition, etc. Moreover, we apply the power of sklearn library. model_selection.train_test_split, which is coming from scikit-learn library and makes the splitting of our data into training and testing subsets quite seamless. This approach is the key to designing reliable and flexible models for our machine learning. Lastly, we are employing some specific functionalities of TensorFlow Library, particularly tf. import keras.models, tf.keras.layers to build and configure the neural network model. These features provide a full-fledged environment for mode building, training, and testing deep learning models, enabling me to devise sophisticated structures leveraged to my own goals.

3.5.2.4 Pre-processing

However, before machine learning models are in the training phase, there is an image preprocessing step that is been done where the images are processed, and a number of operations are done so that the data can be ready for the utilization. At first, we store the photos that are available in the dataset. The data set consists of 43 classes that are marked with 43 different folders containing of images. This cyclic process for us revolves around going through these folders and the images inside them. Every time we open a picture, we are accessing its pixel data array to reduce the size of the image to 32x32 and possibly convert it from the RGB color space to grayscale. A grayscale version of the pictures can be helpful in this respect generating the model more precisely as it removes extra biases from the network therefore allowing the network to process them better. With this conversion the number of computations is decreased because the number of channels are also decreased. Thus, it assures a higher-level accuracy at the completion point. These transformations help by simplifying the data as it retains the essential features and discards the others. Finally, after completing the processing of both images, we store them as NumPy arrays which works better with numerical information processing in Python programming language. The final

step is to bring back the original values by dividing each by 255. This scaling procedure, widely known as normalization, on the other hand, is performed by mapping the pixel values between 0.0 and 1.0, which improves the algorithms' convergence during the training phase. We should know how to prepare images for learning and modelling from experience. In all, we have up to 39.209 images scattered across the 43 classes making this dataset suitable to train and test our machine learning models.

3.5.2.5 Design of convolutional neural network model

Our classification task will be done using a LeNet-5 Convolutional Neural Networks that will be implemented. Launched in 1998 by Yann LeCun and colleagues [30], the LeNet-5 holds as one of the first convolutional neural networks. Model structure is composed of various important parts. The one first of the Input Preprocessing is the job of the Rescaling Layer, which normalizes the input pixel values in the range that provide a solid and sound model stability during a training period. Not only, input images are resized to a 32x32 pixel format and grayscale representation to be converted to a single-channel image. The model uses three Convolutional Layers for feature extraction. The first layer has 16 filters with (5, 5) kernel size, and Rectified Linear Unit activation function. Then, the filters of the next layers include 16 and 120 with the previous ones' same size and function. Along with the sequence of the convolutional layers there are two AveragePooling2D layers that compress the spatial dimensions and preserve the informative spatial information. The regularization is seen through the Dropout Layer after the third Conv2D layer, with a rate of 0.2, which is to make sure that overfitting does not happen by dropping out a fraction of units at random. After that, a Flattening Layer will reshape the output of the convolutional layers into a one-dimensional list, which will be ready for input into two fully connected layers. The first Dense layer, with 120 units, performs the combination of features using the ReLU activation function and further abstraction for abstraction, and the second Dense layer, with the output layer of 43 units as the number of classes of the classification task, follows it. It applies SoftMax activation function to output the class probabilities. Since the architecture comprised of convolutional, pooling, dropout, and dense layers can extract hierarchical features from the input images and make inference across different classes, the model is comprehensive. For training, the model is compiled with the Adam optimizer, categorical cross-entropy loss function, and accuracy metric for evaluation.Table 3.4 describe the CNN architecture.

Table 3.2: CNN Architecture Model.

Layer Type	Description
Input Pre-processing	Rescaling layer normalizes input pixel values and resizes images to 32x32 pixels with grayscale.
Convolutional	Three Conv2D layers extract features with 6, 16, and 120 filters, respectively, using a (5, 5) kernel size and ReLU activation function.
Pooling	Two AveragePooling2D layers down sample feature maps by averaging values in a 2x2 window.
Regularization	Dropout layer with a dropout rate of 0.2 randomly drops units during training to prevent overfitting.
Flattening	Reshapes the output from convolutional layers into a one-dimensional array.
Dense	Two Dense layers with 120 units and ReLU activation, and 43 units (output) with SoftMax activation.

4. PRACTICAL RESULTS AND DISCUSSION

4.1 INTRODUCTION

Chapter 4 focuses on implementing and evaluating the proposed system. It starts by listing the system requirements and then discusses the maneuvers of the vehicle model, showing their accuracy under different conditions. Practical implementations include turning with a constant radius and tangential entry/exit approaches. The chapter also covers practical path tracking, demonstrating how the vehicle navigates to a target point with minimal error. It discusses path tracking with obstacles and the simulation of an anti-lock braking system (ABS). Furthermore, the chapter addresses the practical implementation of object detection and traffic sign recognition. It evaluates the performance of machine learning classifiers and visualizes the results. Overall, this chapter provides a detailed look at how the proposed system operates in real-world scenarios.

4.2 YSTEM REQUIREMENTS

To setup the proposed system, few requirements must be met in the PC base station. The following software and hardware are to be installed:

a. MacBook or Windows 7 and above.
b. VS Code program Software.
c. Google Colap.
d. MIT App Inventor with mobile phone.
e. The open-source Arduino Software (IDE).
f. A USB port (to connect the Arduino Mega).
g. The complete mobile robot.

4.3 SCENARIO VEHICLE MODEL

Many of vehicle manoeuvres are developed, that showing the accuracy of the different models under low and high circumstances. Specifically, body of vehicle, tire, vehicle dynamics, tire force saturation, and tire side force lag are shown to be important effects to include in models for emergency manoeuvres.

4.3.1 Practical Implementation of Turning with Constant Radius

In this part of practical implementation of manoeuvre, the mobile phone station will send the programmed manoeuvre scenario to the vehicle model by Bluetooth, and the vehicle in regards will process the manoeuvre program based on the Arduino controller and other hardware components.

First, the vehicle model must be connected with the mobile phone station to make the vehicle established a wireless communication with station, the Figure 4.1 shows the connection procedures by Bluetooth and choosing the pair Bluetooth (HC-05) that belong to the vehicle. After the connection process completed, the calibration process is very important to the vehicle to make it aware about the environment and in which X, Y positions it is. Calibration process based on the connected digital compass (HMC-5883l); Figure 4.2 explains the calibration positions of vehicle.

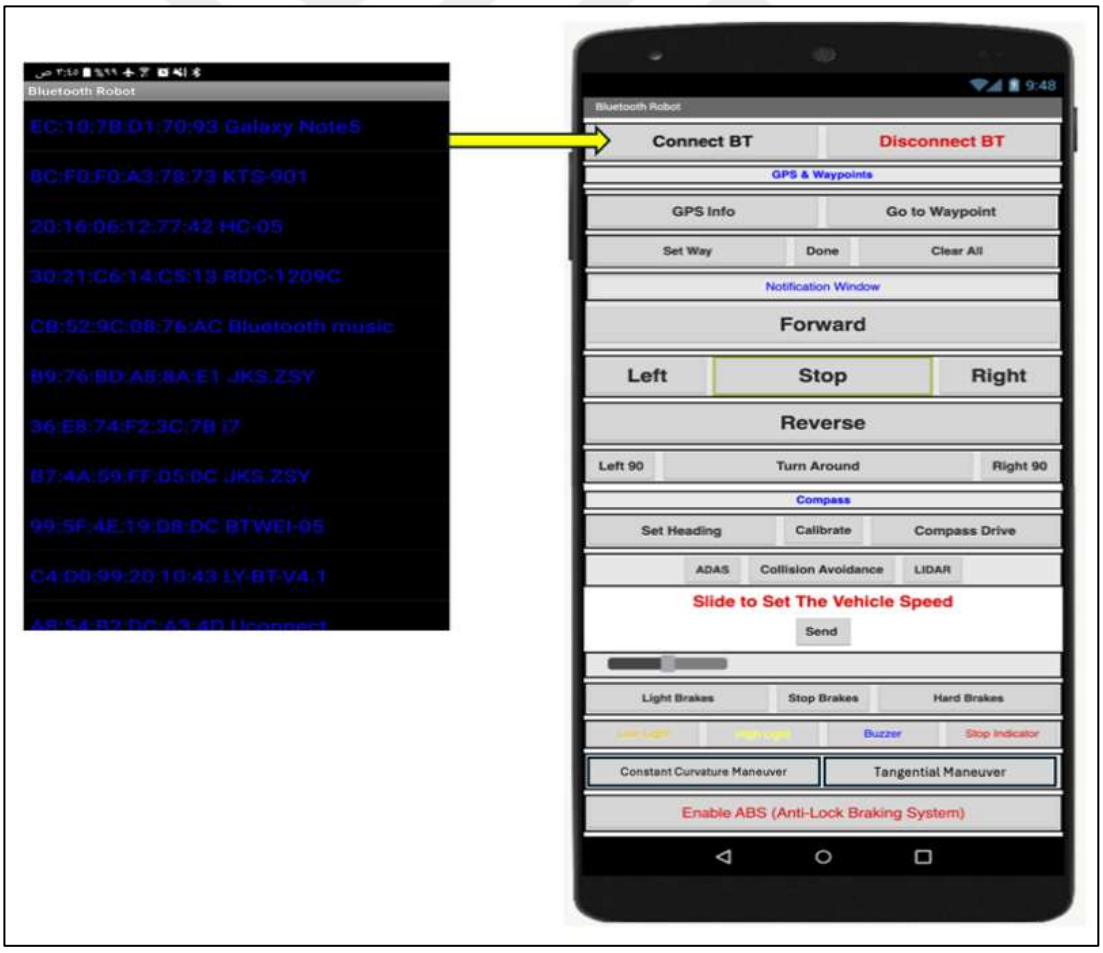

Figure 4.1: Connection Procedure of Mobile Phone.

Figure 4.2: Calibration Vehicle Procedure.

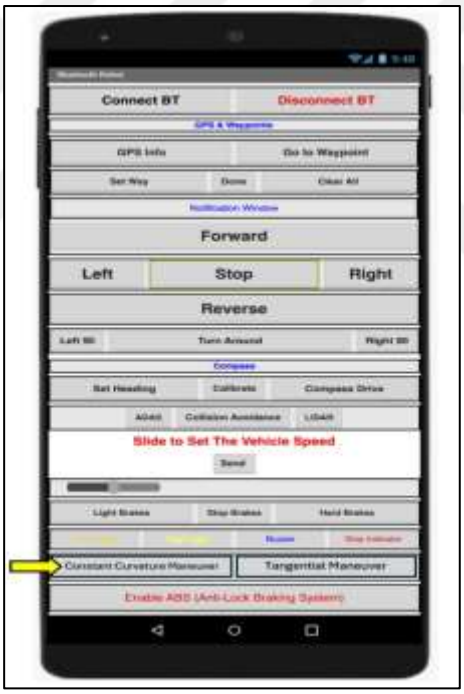

Figure 4.3: Sending Procedure of First Manoeuvre.

The calibration result show in Figure 4.2, and it's 67 for X axis and -252 for Y axis and these coordinates are changeable from location to another, after completing from calibration process for the vehicle, the manoeuvre is ready to send to the vehicle as shown in Figure 4.3, and the vehicle will take the manoeuvre according to the directions that send by station.

After receiving the manoeuvre by vehicle, the vehicle will process the manoeuvre and start the manoeuvre scenario, in other word the vehicle will begin moving slowly and comparing the GPS locations and digital compass heading based on given manoeuvre, the Figure 4.4 showing the implementation of manoeuvre. The diameter implementation curvature was 20 m with 4 km/h speed velocity of vehicle and the motion time of vehicle was around 10s, the turning angle of vehicle that provided by the servo motor is about 6 rad. The Table 4.1 explains all factors of practical implementation of turning with constant radius manoeuvre.

Figure 4.4: Implementation of Constant Curvature.

Table 4.1: Factors of Practical Implementation of Turning with Constant Radius Manoeuvre.

NO.	**Implementation Factors**	**Practical Value**
1	Velocity Speed of Vehicle	4 Km/h
2	Diameter curvature of Manoeuvre	20 m
3	Heading Angle of Vehicle	6 rad
4	Calibration Positions of X and Y Axis	X=60, Y=248
5	Manoeuvre Time	10 s

The error or deviation is generated from the process of calculating the direction of proposed manoeuvre which depends on the GPS, digital compass, servo motor, DC motor and other dynamic parts of vehicle. In other words, the vehicle must run the planning manoeuvre that is sent and reaches a final of manoeuvre with minimum error.

4.3.2 Practical Implementation of Tangential Entry and Exit

The practical implementation of manoeuvre will be done as the same procedure that explained before in previous manoeuvre, instead of sending the constant radius manoeuvre, tangential entry and exit manoeuvre well be send to the vehicle through android application of mobile phone station via Bluetooth wireless communication. This is clearly illustrated in Figure 4.5.

Figure 4.5: Sending Procedure of Second Manoeuvre.

After receiving the manoeuvre by vehicle mobile robot, it will start the calibration and set heading exactly as the previous and start the manoeuvre accordingly. But with different values of set heading and calibration due to use different location not as the same place of last manoeuvre. Then the vehicle will process the manoeuvre and start the manoeuvre scenario as showing clearly in Figure 4.6. The implementation curvature diameter was 50 m

with 4 km/h speed velocity of vehicle and the motion time of vehicle was around 16s. The turning angle of vehicle that provided by the servo motor is variable in this manoeuvre due to multiple directions of vehicle based on the proposed manoeuvre. Table 4.2 explains all factors refer to this manoeuvre for practical implementation of this manoeuvre.

Figure 4.6: Implementation of Tangential Entry and Exit.

Table 4.2: Factors of Practical Implementation of Tangential Entry and Exit Manoeuvre.

NO.	**Implementation Factors**	**Practical Value**
1	Velocity Speed of Vehicle	4 Km/h
2	Distance of Manoeuvre	50 m
3	Heading Angle of Vehicle	3.5 rad
4	Calibration Positions of X and Y Axis	X=66, Y=200
5	Manoeuvre Time	16 s

After the simulation implementation of the two manoeuvres have been completed, there are observation that should be taken into consideration which is the vehicles heading angle or orientation must not be the same with the steering angle of servo motor in the implementation and simulation of the vehicle model. The steering angles means the ability of servo motor shaft to rotate forgetting the desired vehicle orientation.

4.4 PRACTICAL PATH TRACKING

The vehicle navigation is said to be accurate when it determines the path line to the target point with minimum error and minimum oscillations, i.e. if a vehicle draws a trail on its motion, the shape of this trail on the floor would be closest to the shape of the original line. The error or deviation is generated from the process of calculating the direction to target point which depends on the GPS and digital compass. In other words, the robot must run straight forward to each point in the path that is sent and reaches a final target point with minimum error.

The first case has been conducted to test the vehicle's ability to navigate a given points of a path and reach the desired destination successfully without obstacles in the environment. The scenario of this test is that the points of the path that was found by the A* algorithm for a given waypoints are sent to the mobile robot, testing the navigated algorithm of vehicle, receiving the actual path of the vehicle through GPS, and comparing it with the selected path.

This test is implemented on the robot in one place but with different paths, the first path is a straight line between two points and these points are located in Missan university stadium as shown in Figure 4.7 that's getting from google maps., the first location are received from the vehicle GPS via Bluetooth and save it on the mobile phone station, the second location is selected by same way and save also in mobile phone station. The selection of points here done by user of mobile phone station by controlling manually in vehicle to desired points and remarked them as a waypoint to make the vehicle tracking optimal path between these points.

Now, after the waypoints completed the A* algorithm is implemented, and it selects an optimal path between start and goal location. Then the vehicle will behave according to the results of A* algorithm, the length of the path was 20 meters and consists of two points as shown in Figure 4.7. Figure 4.8 shows the selected path and the actual path that is received of the vehicle.

Figure 4.7: Target and Start Waypoints.

Figure 4.8: The Selected Path from A* Algorithm with Real Path from Vehicle.

and results in Table 4.3 in which it can be noticed that the vehicle determines its path line to the target point with minimum error miss distance withon of 1 meter.

Table 4.3: Practical Results Recorded of Vehicle Path.

Path number Target	Start locations	Final target	Length of path(meter)	Max. deviation from path(meter)	Target miss distance (meter)	Total time (min)
1	Lat:31.278126 Lng:47.374010	Lat:33.278415 Lng:44.373978	35	0.6	2.0	2.1
2	Lat:31.278152 Lng:47.374234	Lat: 33.278442 Lng:44.374074	33	0.6	2.0	2
3	Lat:31.229614 Lng:47.389876	Lat: 33.277955 Lng:44.374105	25	0.5	1.2	1.8
4	Lat:31.278152 Lng:47.374234	Lat: 33.229686 Lng:44.390143	24	0.5	1.1	1.7
5	Lat:31.278126 Lng:47.374010	Lat: 33.278377 Lng:44.374246	20	0.7	0.7	1.5
6	Lat:31.277681 Lng:47.377192	Lat: 33.278308 Lng:44.373924	15	0.4	0.6	1.2
7	Lat:31.229765 Lng:47.319886	Lat: 33.229703 Lng:44.389897	8	0.4	0.4	0.8

4.5 PRACTICAL PATH TRACKING INCLUDING OBSTACLES

The other case has been performed to test the obstacle avoidance algorithm. In this case the vehicle moves forward to the target points avoiding all the obstacles in its path. It has been proven that the sensors of the mobile robot can detect the obstacles which far away from the vehicle by 1 m and this value of distance can be changed by programming up to 4.5m.

The vehicle detects an obstacle and avoids it by using obstacles avoiding algorithm explained in chapter three. Figure 4.9 shows how the vehicle behaves when there is only one obstacle in its path, Figure 4.10 shows the case when there are two obstacles and Figure 4.11 shows when there are complicated obstacles and how the vehicle deals with them. The scenario of the first case is repeated but with local obstacles in the path of vehicle, Figure 4.12 shows the actual path of mobile robot when there are obstacles in its way, the location of obstacles is drawn by hand in the map for clarification.

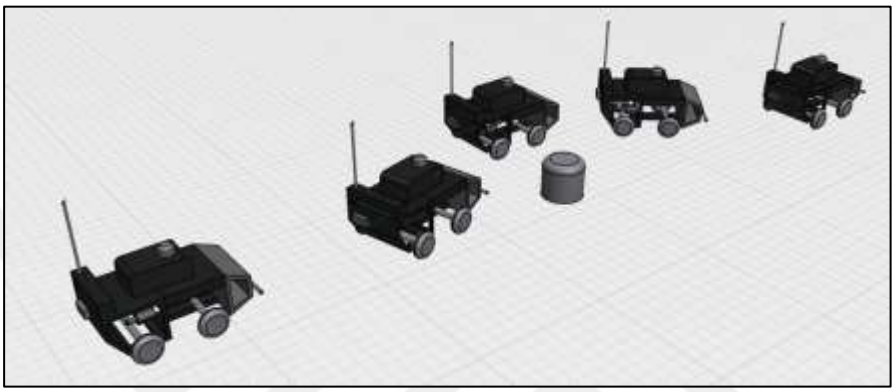

Figure 4.9: Vehicle with One Obstacle.

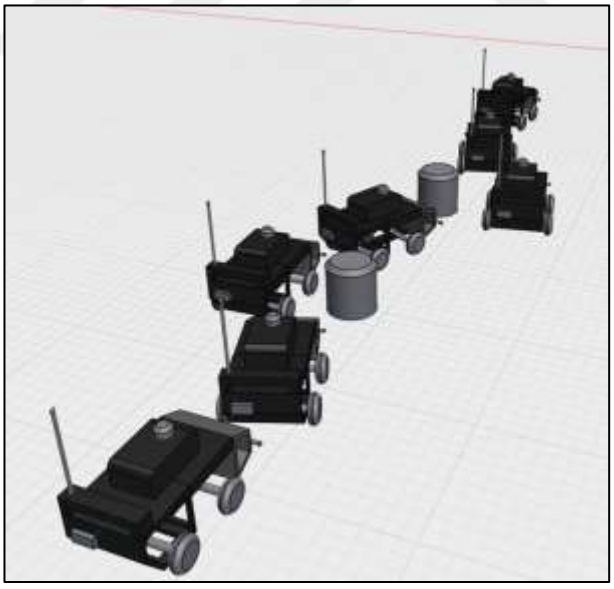

Figure 4.10: Vehicle with Two Obstacles.

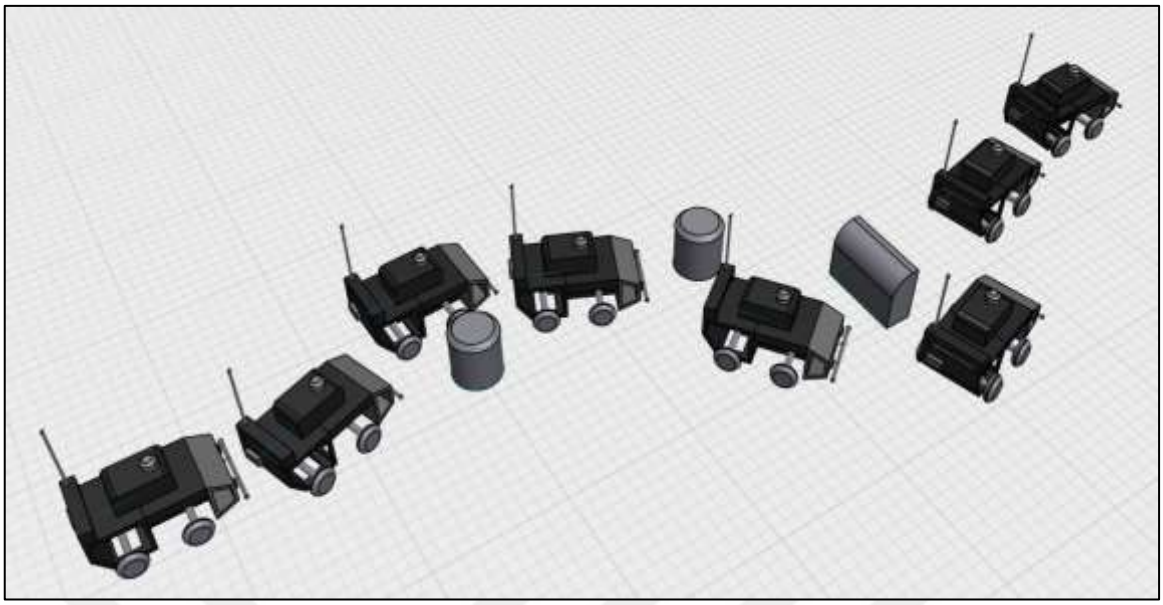

Figure 4.11: Vehicle with Three Obstacles.

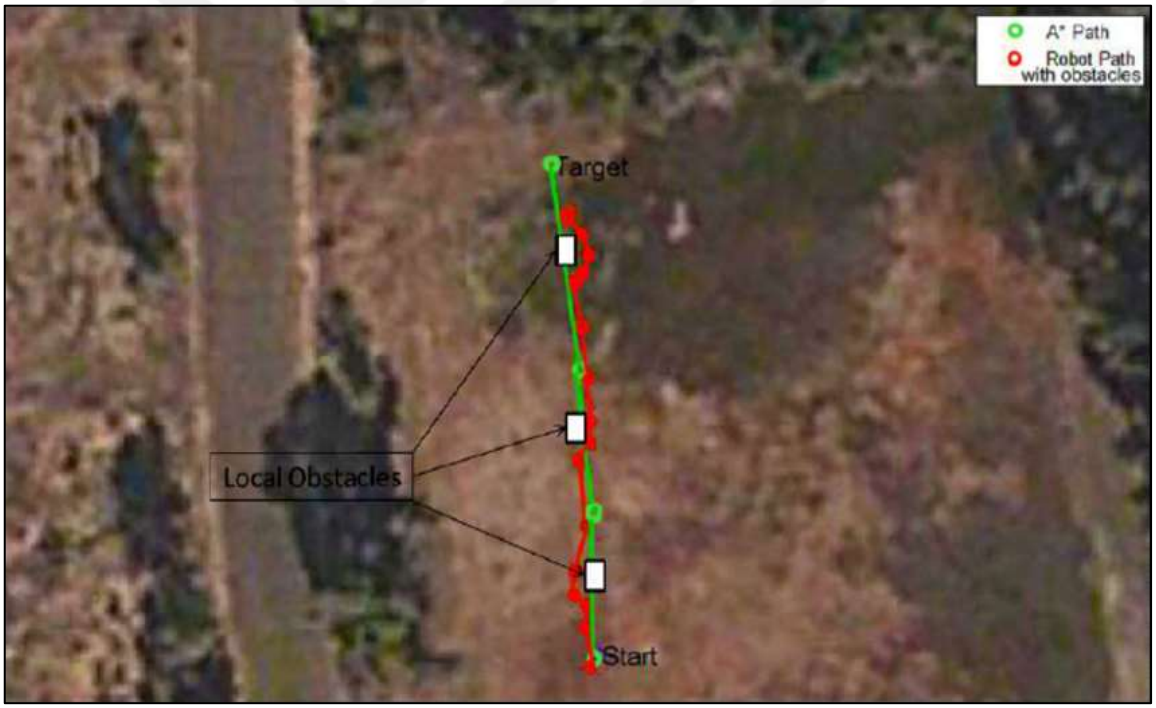

Figure 4.12: Vehicle Path with Local Obstacles.

4.6 SIMULATION AND PRACTICAL IMPLEMENTATION OF ABS

The simulation of anti-lock braking system shows the dynamic and kinematics behaviour of a vehicle mobile robot under multi braking conditions. The model represents a single wheel, which may be replicated several times to create a model for a multi-wheel vehicle.

To control the rate of change of brake pressure, the model subtracts actual slip from the desired slip and feeds this signal into a bang-bang control (+1 or -1), depending on the sign of the error, see Figure 4.13. This on/off rate passes through a first-order lag that represents the delay associated with the hydraulic lines of the brake system. The model then integrates the filtered rate to yield the actual brake pressure. The resulting signal, multiplied by the piston area and radius with respect to the wheel (Kf), is the brake torque applied to the wheel.

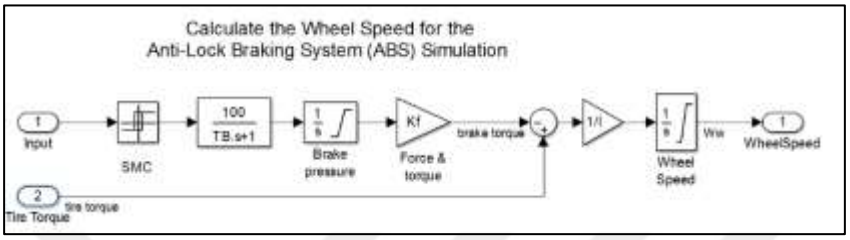

Figure 4.13: Wheel Speed Subsystem.

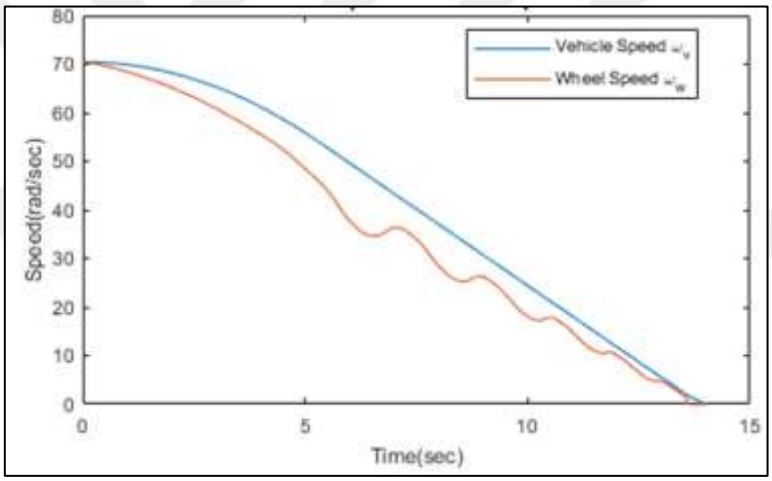

Figure 4.14: Wheel Angular Velocity.

Figure 4.14 shows visualize the ABS simulation results (for default parameters) of wheel angular velocity and corresponding vehicle angular velocity. This plot shows also that the wheel speed stays below vehicle speed without locking up, with vehicle speed going to zero in less than 15 seconds.

Figure 4.15 shows the normalized slip ratio that made the wheel speed stays below vehicle speed without locking up.

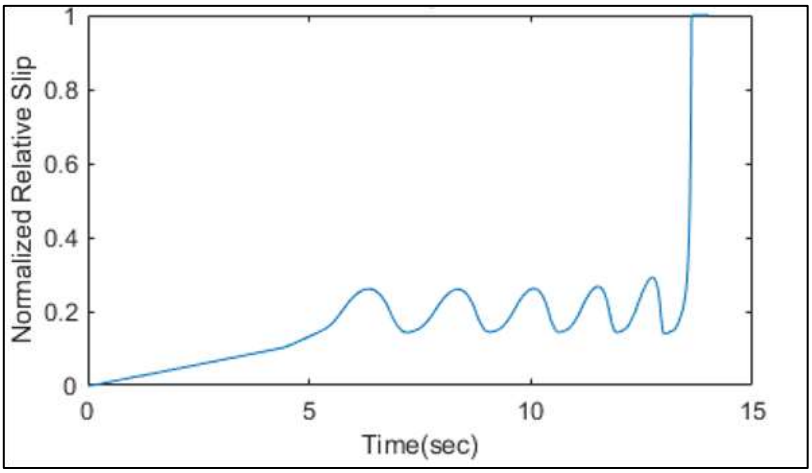

Figure 4.15: Normalized Slip Ratio.

4.7 PRACTICAL IMPLEMENTATION OF OBJECT DETECTION

4.7.1 Performance Evaluation

Performance assessment with machine learning refers to testing the efficacy of a trained model on previously unseen information. It's the same as putting the model on an exam to check how well it can predict something or classify. Various means are used to measure its effectiveness. For instance, precision will indicate the percentage of correct predictions the model makes out of all the predictions it makes. Precision determines how many of the predicted instances that were positive were indeed positive and recall tells us how many of the actual positive instances the model can identify. F1-score balances the precision and recall. Through assessing a model's performance, one can tell its strengths and weaknesses as well as improve it if need be. This helps us decide what a model will look for and how it will perform in different environments. For our case, combined more than ten datasets and placed them in the same folder, and we will use three different classifiers; Multilayer Perceptron Classifier (MLP), Decision Tree Classifier, and Random Forest Classifier to check which one is the best among them. Each classifier was performance-assessed on our dataset.

We will start with MLP Classifier, and the results shown below in the Table 4.4:

Table 4.4: Performance Evaluation Using MLP Classifier.

Metric	Precision	Recall	F1-Score
F	0.63	1.00	0.77
G	0.00	0.00	0.00
I	0.00	0.00	0.00
Accuracy	-	-	0.63
Support			
F	-	-	2318
G	-	-	767
I	-	-	581

While when we used Decision Tree Classifier the result shown below in the Table 4.5:

Table 4.5: Performance Evaluation Using Decision Tree Classifier.

Metric	Precision	Recall	F1-Score
F	0.71	0.71	0.71
G	0.48	0.48	0.48
I	0.49	0.49	0.49
Accuracy	-	-	0.63
Support			
F	-	-	2318
G	-	-	767
I	-	-	581

But when we used the Random Forest Classifier with our dataset, we got better results as shown below in the Table 4.6:

Table 4.6: Performance Evaluation Using Random Forest Classifier.

Metric	Precision	Recall	F1-Score
F	0.75	0.90	0.81
G	0.69	0.50	0.58
I	0.78	0.43	0.55
Accuracy	-	-	0.74
Support			
F	-	-	2318
G	-	-	767
I	-	-	581

4.7.2 Visualization Results

In this section of the thesis, we will build simulation models to execute all the data collected for each scenario separately and observe how the vehicle will behave in avoiding collisions. Three different paths were created: circular, square, and infinity shapes as shown inFigure 4.16, Figure 4.17, Figure 4.18, Figure 4.19, and the vehicle was driven within these paths. Data collection was facilitated by the usage of lidar and ultrasonic sensors. Three files were created to be used as inputs in the simulation. In order to use random forest classifier and its output was excellent because the vehicle drives on these channels without collision. The images, below, represent the outcome of a simulation as well as the shape of every track.

Figure 4.16: Clockwise Circular Path.

Figure 4.17: Anti Clockwise Circular Path.

Figure 4.18: Square Path.

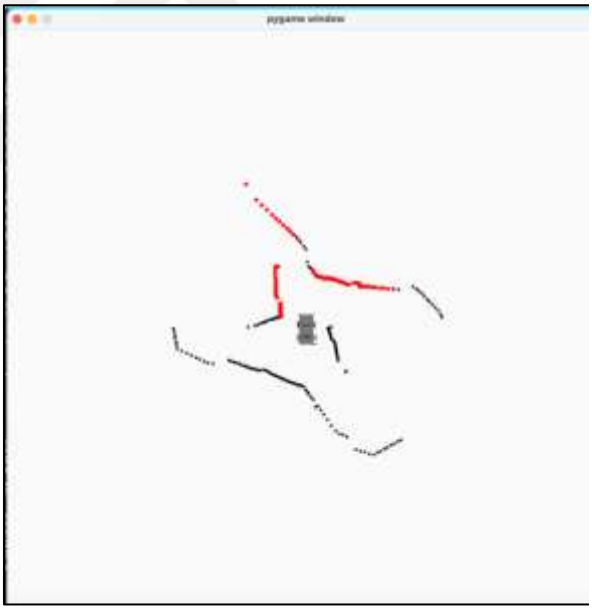

Figure 4.19: Infinity Path.

4.7.3 Practical Experimental Results

In this section we will test our vehicle model in new environment path that didn't see it before to evaluate our vehicle performance and see how the vehicle can behave without any collisions. In the beginning we send the vehicle the starting and target point through as longitude and latitude though our mobile platform and put some different of obstacle in its

path and send it the order to start its journey. The vehicle was able to deal with all fixed and moving obstacles and walls with impressive success. Also, when it was surprised by an obstacle such that there was no distance to overcome this obstacle, the vehicle would stop, move backwards for a few feet, then change its course, heading toward the target point. However, when the vehicle was placed on an unorganized track, the edge of the vehicle collided with the track several times, so it still needs more training on many tracks, as the more data trained on it increases, the better its performance and thus increases the safety of the vehicle.

4.8 PRACTICAL IMPLEMENTATION OF TRAFFIC SIGNS RECOGNITION

4.8.1 Train and Validate the Model

The training for a machine learning model is a fundamental part of the model development and consists of an iterative process of fine-tuning the parameters of the model and choosing a loss function to minimize. Usually, this phase of the training runs over many rounds, known as epochs. The model within each epoch is shown the training set of batches. In each batch, the model is computing the loss which shows the difference between the output rates produced by it and the target values actually present. For this the ANN relies on an optimization algorithm, for instance, stochastic gradient descent, to change these parameters so it has less incurred loss. The repeated cycle of model development and training enables the model to extract the hidden patterns in data and to refine its forecast capabilities. Along with appropriate training data validation data is also important during training stages. Validation data is a whole different dataset that the model does not train on but is still going to evaluate its performance at the end of each epoch. The model is assessed by making use of unseen data during validation which gives a measure of how well the model performs on new instances. Tracking model's performance on both training and validation data can help us to identify overfitting that takes place when the model fails to predict new data correctly despite the good results on the training data. The advantage of this information for leading the search for the optimal hyperparameters and verifying the model's capacity to generalize to samples that have not been seen before. below table presents the training and validation metrics details for epochs ranging from 1 and up to 50 during which the training was in process as shown in the Table 4.7.

Table 4.7: Evaluate Training Results for 50 Epoch.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	1.9154	0.4598	0.7200	0.7932
2	0.6807	0.7951	0.4133	0.8827
3	0.4622	0.8607	0.2995	0.9104
4	0.3555	0.8924	0.2199	0.9449
5	0.2844	0.9143	0.1839	0.9485
6	0.2390	0.9267	0.1505	0.9614
7	0.2055	0.9367	0.1292	0.9638
8	0.1871	0.9435	0.1285	0.9668
9	0.1590	0.9500	0.0991	0.9748
10	0.1469	0.9565	0.0991	0.9750
11	0.1340	0.9579	0.1188	0.9697
12	0.1228	0.9620	0.0923	0.9756
13	0.1121	0.9658	0.0873	0.9787
14	0.1053	0.9668	0.0782	0.9787
15	0.0932	0.9707	0.0870	0.9777
16	0.0940	0.9702	0.0890	0.9769
17	0.0861	0.9726	0.0741	0.9824
18	0.0810	0.9747	0.0828	0.9802

Table 4.7: Evaluate Training Results for 50 Epoch “Table Continued”

19	0.0798	0.9740	0.0845	0.9790
20	0.0733	0.9764	0.0680	0.9851
21	0.0717	0.9772	0.0628	0.9851
22	0.0657	0.9791	0.0712	0.9838
23	0.0620	0.9802	0.0608	0.9861
24	0.0620	0.9797	0.0600	0.9875
25	0.0584	0.9817	0.0587	0.9861
26	0.0557	0.9817	0.0570	0.9872
27	0.0547	0.9827	0.0576	0.9878
28	0.0503	0.9835	0.0651	0.9848
29	0.0518	0.9835	0.0570	0.9875
30	0.0472	0.9849	0.0656	0.9862
31	0.0476	0.9837	0.0577	0.9878
32	0.0436	0.9861	0.0764	0.9832
33	0.0527	0.9837	0.0657	0.9857
34	0.0414	0.9870	0.0513	0.9902
35	0.0407	0.9865	0.0571	0.9878
36	0.0451	0.9858	0.0545	0.9898
37	0.0368	0.9876	0.0569	0.9883

Table 4.7: Evaluate Training Results for 50 Epoch "Table Continued"

38	0.0401	0.9871	0.0596	0.9884
39	0.0394	0.9870	0.0669	0.9847
40	0.0339	0.9892	0.0587	0.9878
41	0.0376	0.9883	0.0499	0.9906
42	0.0326	0.9894	0.0540	0.9897
43	0.0369	0.9885	0.0571	0.9888
44	0.0354	0.9883	0.0590	0.9865
45	0.0362	0.9885	0.0581	0.9885
46	0.0342	0.9891	0.0503	0.9895
47	0.0327	0.9898	0.0640	0.9890
48	0.0336	0.9893	0.0575	0.9894
49	0.0329	0.9900	0.0510	0.9892
50	0.0302	0.9903	0.0598	0.9892

4.8.2 Evaluation of Training Results

Whether a model is good or bad is determined by evaluating numerous factors, and it is among learning curves that their role is of prime importance in the process. Our model is evaluated in terms of the accuracy of training, testing, as well as the loss curve on validation.

a. Train Accuracy and Validation Accuracy Convergence: For an ideal case the divergence of both curves should be decreased with increasing the training: that is when the model learns well from the data. In our instance, converging curves turn out to be 50 epochs after reaching 98.9% of accuracy. This very high accuracy signifies the model has a good

performance on the training and validation set the simulation results shown in Figure 4.20.

b. Validation Loss Curve Behaviour: An undulation of the Validation Loss curve is usually caused by the fact that the model performs not quite correctly on the unobserved validation data. To make matters ideal, we want this curve to go down steadily which insinuate that the model is generalizing positively to new data. On the other hand, the curve not being smooth indicates the performance of this model may differ from one type of data to another on the validation data subset. Moreover, if the curve is constantly increasing along epochs, it is a sign of overfitting, in which the model is attending to training data as opposed to generalizing the patterns. In our case, the Validation Loss curve exhibits some fluctuations, and after about 25 epochs, it surpasses the Train Loss, indicating a slight overfitting. However, since the difference between Validation and Train Loss is not substantial, and the curve doesn't consistently rise over epochs, this overfitting is tolerable, and the simulation results shown in Figure 4.21.

In conclusion, although there are minor fluctuations in the validation metrics and indications of slight overfitting, the model's high accuracy and the relatively stable behaviour of the learning curves suggest that it performs well. Therefore, we can consider the model to be good, and it's acceptable to stop training at this point.

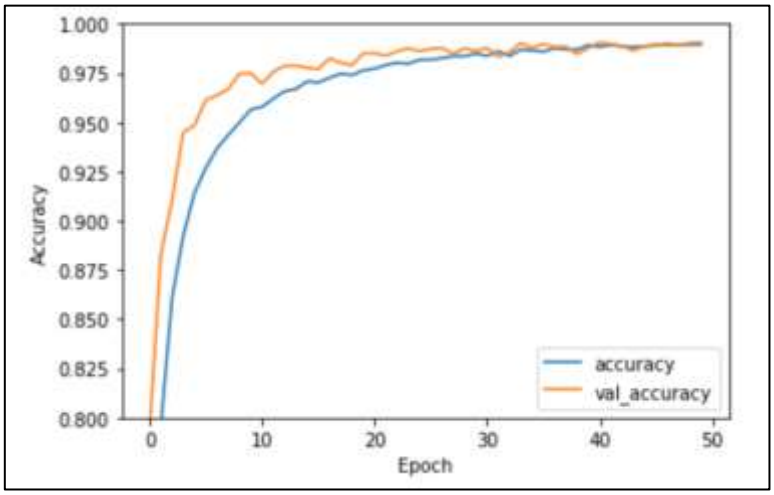

Figure 4.20: Training and Validation Accuracy by Using Training Epochs.

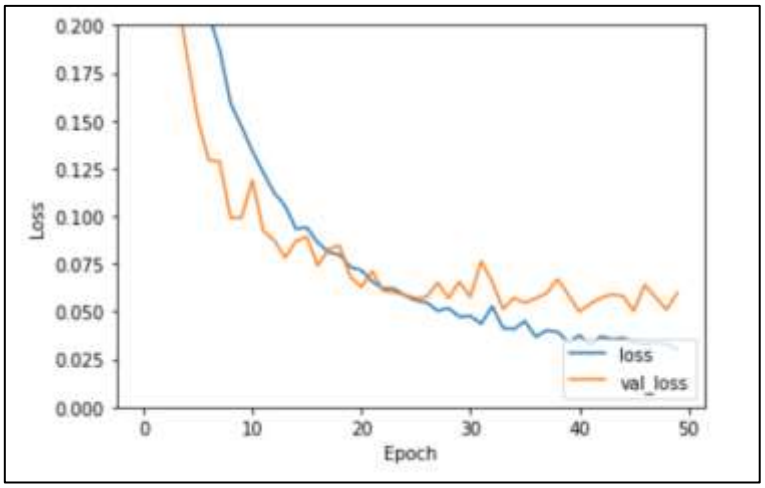

Figure 4.21: Training and Validation Loss by Using Training Epochs.

In the process of identifying misclassified images, we undertake a detailed examination of sample images by comparing their predicted classes with their ground truth labels. When the model's prediction aligns with the ground truth, indicating a correct classification, we denote the image label in green. Conversely, if the predicted class differs from the ground truth, implying a misclassification, we designate the label in red. For instance, upon analysing image number 5955, we uncover an instance where the model misclassified the "Speed limit (50km/h)" sign as "Speed limit (80km/h)". This misclassification underscores the challenges associated with accurately recognizing the text on the sign, contributing to the discrepancy between the predicted and actual classes as shown in Figure 4.22.

Figure 4.22: Some Samples of Traffic Sign and Find the Wrong Classified Pictures.

4.8.3 Test the Model in an Application

In the final step of our workflow, we safeguard the trained model by storing it in a designated folder on Google Drive through the model. Save method. This repository holds not only the model's graph definitions but also its weights, serving as indispensable assets for subsequent predictions in our traffic sign recognition application. It's evident that end-users, spanning from customers to those unversed in Data Science, may not delve into the technical intricacies of our Notebook, such as graphs, models, or accuracy metrics. Their primary interest lies in a user-friendly application where they can seamlessly upload an image and promptly receive an accurate prediction. To streamline this process, we've implemented

features enabling users to upload their images and obtain predictions for their respective traffic signs. Before initiating the prediction process, however, it's essential to pre-process the user's image by converting it to grayscale, resizing it to a standard 32x32 pixel dimension, and saving it in np. float32 format. Moreover, we've prioritized enhancing user experience by providing visual representations of both the uploaded image and its pre-processed 32x32 grayscale version. Additionally, we furnish users with an expanded list enumerating the 43 classes available in the model, fostering transparency, and understanding. Following the image upload, users receive the prediction alongside a confidence rating, thereby empowering them with actionable insights concerning the identified traffic sign the Figure 4.23 shows testing on four different traffic signs by predicting the name of sign and its confidence.

Figure 4.23: Screenshot for Sampling of Traffic Sign Recognition.

5. CONCLUSION

The design and implementation of an autonomous vehicle enhanced with Advanced Driver Assistance Systems (ADAS) represents a significant advancement in the field of autonomous vehicle technology. This vehicle can be classified as an educational platform suitable for researchers and specialists in the field of autonomous vehicles. Its structure can be easily modified to meet the needs of researchers, and it can be reprogrammed with ease. Throughout this thesis, we have presented a detailed explanation of the work execution, outlining the steps involved in constructing the vehicle's structure and providing in-depth descriptions of the suspension system, steering system, brake system, and anti-lock braking system (ABS), among others. One of the key features of this autonomous vehicle is its ability to be controlled via a mobile phone application. Through the development of a control program using MIT App Inventor, users can wirelessly drive the vehicle via Bluetooth, offering a convenient and user-friendly interface. Moreover, we have discussed the primary tasks that this vehicle can perform, including path planning and navigation using a specially designed algorithm capable of selecting the shortest route to the destination. Additionally, the vehicle is equipped with sensors to detect both stationary and moving obstacles during its journey, utilizing ultrasonic sensors distributed around the vehicle and LIDAR technology. Furthermore, a machine learning model has been developed to sense obstacles, with data collected from various sensors and used to train the model for autonomous driving. The implementation of this work has been carried out through simulation and augmented reality, demonstrating the vehicle's ability to navigate obstacles effectively during its journey. Lastly, the vehicle has been trained to recognize various traffic signs using machine learning, specifically trained on a dataset of German traffic signs comprising over 50,000 samples representing 43 classes. The model's performance has been evaluated through visualization and testing, including real-time recognition using the vehicle's onboard camera. In conclusion, the design and implementation of this autonomous vehicle with ADAS represent a significant step forward in autonomous vehicle technology, showcasing its capability to navigate, detect obstacles, and recognize traffic signs efficiently. The results obtained from this work highlight the potential for further advancements in autonomous vehicle research and development, paving the way for safer and more efficient transportation systems in the future.

REFERENCES

[1] L. Ruhrort, J. Steiner, A. Graff, D. Hinkeldein, and C. Hoffmann, "Carsharing with electric vehicles in the context of users' mobility needs–results from user-centred research from the BeMobility field trial (Berlin)," International Journal of Automotive Technology and Management, vol. 14, no. 3-4, pp. 286-305, 2014.

[2] J. Yang, "Review of injury biomechanics in car-pedestrian collisions," International Journal of Vehicle Safety, vol. 1, no. 1-3, pp. 100-117, 2005.

[3] SAE On-Road Automated Vehicle Standards Committee, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," SAE Standard J3016, pp. 1, 2014.

[4] S. Automotive, "The 6 levels of vehicle autonomy explained," [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html. [Accessed: Sep. 20, 2022].

[5] NHTS, "Automated vehicles for safety," May 2018. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety. [Accessed: Jul. 3, 2024].

[6] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in 2006 2nd International Conference on Information & Communication Technologies, vol. 1, pp. 781-786, IEEE, April 2006.

[7] P. Naderi, A. R. Naderipour, M. Mirsalim, and M. A. Fard, "Intelligent braking system using fuzzy logic and sliding mode controller," Control and Intelligent Systems, vol. 38, no. 4, pp. 236, 2010.

[8] K. Židek and E. Rigasová, "Path planning algorithm based on search algorithm, edge detector and GPS data/satellite image for outdoor mobile systems," in 2012 IEEE 10th

International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 349-354, IEEE, January 2012.

[9] S. Khan, K. Ahmad, M. Murad, and I. Khan, "Waypoint navigation system implementation via a mobile robot using global positioning system (GPS) and global system for mobile communications (GSM) modems," International Journal of Computational Engineering Research (IJCER), vol. 3, no. 7, pp. 1-7, 2013.

[10] M. Zhou and S. He, "Research of autonomous navigation strategy for an outdoor mobile robot," International Journal of Control and Automation, vol. 7, no. 12, pp. 353-362, 2014.

[11] J. A. Oroko and G. N. Nyakoe, "Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review," in Proceedings of the Sustainable Research and Innovation Conference, June 2022, pp. 314-318.

[12] W. C. Lin, C. L. Lin, P. M. Hsu, and M. T. Wu, "Realization of anti-lock braking strategy for electric scooters," IEEE Transactions on Industrial Electronics, vol. 61, no. 6, pp. 2826-2833, 2013.

[13] M. Z. Al-Faiz and G. E. Mahameda, "GPS-based navigated autonomous robot," International Journal of Emerging Trends in Engineering Research, vol. 3, no. 4, 2015.

[14] Z. Wang, "Trajectory planning for four wheel steering autonomous vehicle," 2018.

[15] Z. Luo, "LiDAR based perception system: Pioneer technology for safety driving," Ph.D. dissertation, 2017.

[16] V. Karlquist, "Design and implementation of an automotive experimental platform for ADAS," 2017.

[17] S. Fernando, "A Structured Testing Framework for ADAS Software Development," Master's thesis, University of Waterloo, 2023.

[18] M. Xie, "Road Elements Identification and LiDAR Integration for Advanced Driver Assistance Systems," Ph.D. dissertation, Politecnico di Torino, 2023.

[19] T. Y. Lim, A. Ansari, B. Major, D. Fontijne, M. Hamilton, R. Gowaikar, and S. Subramanian, "Radar and camera early fusion for vehicle detection in advanced driver assistance systems," in Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems, vol. 2, no. 7, December 2019.

[20] H. B. and J. Larsson, "End-to-end prediction of the fully unoccluded state of the vehicle surroundings with convolutional recurrent neural network trained in an unsupervised manner," Sweden, 2018.

[21] M. S. N. G. Hanna, "Vehicle Distance Detection Using Monocular Vision and Machine Learning," Master's thesis, University of Windsor, Canada, 2019.

[22] J. Omidokum, "Design and Implementation of an Autonomous Driving System with a Deep Learning Approach on a Scaled Vehicle Platform," Ph.D. dissertation, 2022.

[23] I. Siniosoglou, P. Sarigiannidis, Y. Spyridis, A. Khadka, G. Efstathopoulos, and T. Lagkas, "Synthetic Traffic Signs Dataset for Traffic Sign Detection & Recognition in Distributed Smart Systems," in 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 302-308, IEEE, July 2021.

[24] P. Naderi, A. R. Naderipour, M. Mirsalim, and M. A. Fard, "Intelligent braking system using fuzzy logic and sliding mode controller," Control and Intelligent Systems, vol. 38, no. 4, pp. 236, 2010.

[25] J. Han, A. Sciarretta, L. L. Ojeda, G. De Nunzio, and L. Thibault, "Safe-and eco-driving control for connected and automated electric vehicles using analytical state-constrained optimal solution," IEEE Transactions on Intelligent Vehicles, vol. 3, no. 2, pp. 163-172, 2018.

[26] MIT App Inventor. [Online]. Available: https://appinventor.mit.edu.

[27] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, et al., "Path planning of automated guided vehicles based on improved A-Star algorithm," in 2015 IEEE International Conference on Information and Automation, pp. 2071-2076, IEEE, August 2015.

[28] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: a multi-class classification competition," in The 2011 International Joint Conference on Neural Networks, pp. 1453-1460, IEEE, July 2011.

[29] A. A. Minai, "2011 International Joint Conference on Neural Networks (IJCNN 2011)," IEEE Computational Intelligence Magazine, vol. 7, no. 1, pp. 13-15.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[31] V. V. Gómez, "Lidar-based scene understanding for autonomous driving using deep learning," Ph.D. dissertation, Universitat Politècnica de Catalunya (UPC), 2020.