

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUAN GABRIEL DOS SANTOS AYRES

**DEVELOPMENT AND SIMULATION OF AN AUTONOMOUS PARKING
SYSTEM USING A LOW-COST LIDAR SENSOR AND ULTRASONIC SENSORS**

PONTA GROSSA

2023

LUAN GABRIEL DOS SANTOS AYRES

**DEVELOPMENT AND SIMULATION OF AN AUTONOMOUS PARKING
SYSTEM USING A LOW-COST LIDAR SENSOR AND ULTRASONIC SENSORS**

**Desenvolvimento e simulação de um sistema de estacionamento autônomo
utilizando um sensor lidar de baixo custo e sensores ultrassônicos**

Dissertação apresentada como requisito para
obtenção do título de Mestre em Engenharia elétrica
no Programa de Pós-Graduação em Engenharia
Elétrica da Universidade Tecnológica Federal do
Paraná (UTFPR).

Orientador: Prof. Max Mauro Dias Santos, Ph.D.

Co-orientador: Prof. Rui Tadashi Yoshino, Ph.D.

PONTA GROSSA

2023



[4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Esta licença permite remixe, adaptação e criação a partir do trabalho, para fins não comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.



LUAN GABRIEL DOS SANTOS AYRES

**DEVELOPMENT AND SIMULATION OF AN AUTONOMOUS PARKING
SYSTEM USING A LOW-COST LIDAR SENSOR AND ULTRASONIC SENSORS**

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Controle E Processamento De Energia.

Data de aprovação: 28 de Agosto de 2023

Dr. Evandro Leonardo Silva Teixeira, Doutorado - Universidade de Brasília (Unb)

Dr. Joao Francisco Justo Filho, Doutorado - Usp-Universidade de São Paulo

Dr. Mauricio Dos Santos Kaster, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Rui Tadashi Yoshino, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 28/11/2023.

ACKNOWLEDGMENTS

I would like to express my gratitude to several individuals and institutions that were crucial for the completion of this work.

I extend my thanks to my mother, Ana, for her unconditional love, support, and encouragement throughout all stages of my life. I am grateful to my girlfriend, Giovanna, for her understanding, affection, support, and love in every moment.

I cannot overlook expressing my gratitude to my academic advisor, Prof. Max Mauro Dias Santos, for guidance, and to Prof. Rui Tadashi Yoshino for co-supervision, teachings, and patience throughout the entire research and writing process. Their advice was invaluable and significantly contributed to the quality of this work.

I also want to thank my colleagues at the GSA laboratory, who were always willing to assist and share knowledge, creating a pleasant and stimulating work environment.

Special thanks to Renault for the funding and support that made this research possible. The UTFPR university also deserves acknowledgment for providing the necessary infrastructure for the development of this project.

I want to express my gratitude to everyone who, in any way, contributed to the completion of this work, whether through moral support, encouragement, knowledge, or resources.

I also extend my gratitude to the National Council for Scientific and Technological Development (CNPq), Araucária Foundation and the Foundation for Support of Education, Research, and Scientific and Technological Development of the Federal Technological University of Paraná (FUNTEF-PR) for financial support in partnership with Renault do Brasil, through the notice CP 16/2020 - PROGRAMA DE BOLSAS FUNDAÇÃO ARAUCÁRIA & RENAULT DO BRASIL.

Thank you very much!

ABSTRACT

The automotive industry is constantly transforming, driven by the increasing demand for modern cars with exclusive features. A major challenge for drivers is the difficulty of parking, and the industry seeks solutions through parking assistance. In this context, this work proposes the development of an autonomous parking system, using a low-cost one-dimensional LiDAR sensor for the detection, measurement and classification of the parking space. For this purpose, a detailed bibliographic research was carried out, analyzing the possible paths and possibilities for the implementation of the system, with special focus on the detection of the parking space. After analysis, the RRT* motion planning algorithm was selected for performing simulations in MATLAB software, due to the ease of implementation and the fast-processing speed of the algorithm. In addition, a small-scale prototype of the autonomous parking system was developed and tested. Achieving good results in identifying, measuring and classifying parallel and perpendicular parking spaces. In future work, the aim is to implement the motion planning algorithm in the small-scale prototype, as well as to implement the complete system in a real-scale test vehicle to evaluate the performance and effectiveness of the system under real conditions.

Keywords: autonomous parking system, LiDAR sensor, parking space detection, motion planning algorithm, RRT*, MATLAB simulation.

RESUMO

A indústria automotiva está em constante transformação, impulsionada pela crescente demanda por carros modernos com recursos exclusivos. Um grande desafio para os motoristas é a dificuldade de estacionar, e a indústria busca soluções por meio de assistência ao estacionamento. Nesse contexto, este trabalho propõe o desenvolvimento de um sistema autônomo de estacionamento, utilizando um sensor LiDAR unidimensional de baixo custo para a detecção, medição e classificação da vaga de estacionamento. Para isso, foi realizada uma pesquisa bibliográfica detalhada, analisando os caminhos possíveis e as possibilidades para a implementação do sistema, com foco especial na detecção da vaga de estacionamento. Após análise, o algoritmo de planejamento de movimento RRT* foi selecionado para realizar simulações no software MATLAB, devido à facilidade de implementação e à velocidade de processamento rápida do algoritmo. Além disso, um protótipo em pequena escala do sistema autônomo de estacionamento foi desenvolvido e testado. Obtendo bons resultados na identificação, medição e classificação de vagas de estacionamento paralelas e perpendiculares. Em trabalhos futuros, pretende-se implementar o algoritmo de planejamento de movimento no protótipo em pequena escala, bem como implementar o sistema completo em um veículo de teste em escala real visando avaliar o desempenho e a eficácia do sistema em condições reais.

Palavras-chave: sistema de estacionamento autônomo, sensor LiDAR, detecção de espaço de estacionamento, algoritmo de planejamento de movimento, RRT*, simulação MATLAB.

LIST OF FIGURES

Figure 1 - Travel speeds of production automobiles	21
Figure 2 - Types of parking spaces	24
Figure 3 - Operation of a lidar sensor.....	25
Figure 4 - Range and field of view of different radar sensors	29
Figure 5 - Car with ultrasonic sensors and their positions	32
Figure 6 - GPS triangulation process	34
Figure 7 - Types of movements measure by inertial sensors	36
Figure 8 - Incremental encoder	37
Figure 9 - ADAS functions and sensors	44
Figure 10 - Driving Scenario Designer with a project running	48
Figure 11 - Code generated from a driving scenario in MATLAB	49
Figure 12 - Example of a DYNA4 Simulation.....	50
Figure 13 - Algorithm RRT*.....	56
Figure 14 - SAE vehicle axis system	58
Figure 15 - ISO vehicle axis system.....	58
Figure 16 - Vehicle in an Earth Fixed Coordinate System	59
Figure 17 - Instantaneous center of rotation.....	59
Figure 18 - Two-wheel representation of a four-wheel system	60
Figure 19 - Kinematic bicycle model with the rear axle reference point.....	61
Figure 20 - Inputs and outputs of the kinematic bicycle model	62
Figure 21 - Steering angle in bicycle model.....	63
Figure 22 - An example of a control architecture	64
Figure 23 - Cruise control schematic	68
Figure 24 - Equation to find the desired acceleration of a cruise control	68
Figure 25 - Typical engine map.....	69
Figure 26 - Path following with pure pursuit controller	70
Figure 27 - Pure pursuit controller with cross track error e	71
Figure 28 - PRISMA flow diagram	74
Figure 29 - Creation of a driving scenario with two cars parked	77
Figure 30 - Base scenario for parking space detection	78
Figure 31 - Error in identifying the real size of the parking space	79
Figure 32 - Generated cost map.....	80
Figure 33 - Top view from the road of the parallel parking simulation.....	81
Figure 34 - Top view with the waypoints generated for perpendicular parking.....	82
Figure 35 - Angle parking in bird's-eye plot.....	83
Figure 36 - Schematic with L298N	85
Figure 37 - Schematic with HC-SR04 ultrasonic sensors	86
Figure 38 - Schematic with the TF Mini Plus LIDAR sensor	87
Figure 39 - Schematic with the Arduino uno and an encoder	88
Figure 40 - Schematic with OLED display	89
Figure 41 - Tkinter example	94
Figure 42 - Parking spot detection.....	95
Figure 43 - Measurement of parking spot size.....	96
Figure 44 - Angle ϕ in the three types of parking.....	97
Figure 45 - Relevant measurements in the calculation of an angled parking space	98
Figure 46 - Parallel parking space plot using real data.....	100

Figure 47 - Publications per year	102
Figure 48 - Portfolio Keywords	103
Figure 49 - Author Analysis.....	104
Figure 50 - Diagram of the APS simulation process	105
Figure 51 - Sensor data with longitudinal movement of the ego vehicle	105
Figure 52 - Trajectory generated for parking a vehicle in parallel	106
Figure 53 - System block diagram	108
Figure 54 - Circuit with the LM317T voltage regulator	112
Figure 55 - Acquisition of a parallel parking spot performed by the ultrasonic sensor and LiDAR	114
Figure 56 - Test 1 comparing LiDAR and ultrasonic sensors in a parallel parking space	116
Figure 57 - Test 5 comparing LiDAR and ultrasonic sensors in a perpendicular parking space	117
Figure 58 - Test 5 comparing LiDAR and ultrasonic sensors in a perpendicular parking space	119
Figure 59 - Parking assistant program home screen	120
Figure 60 - Initial screen of the parking assistant program.....	121
Figure 61 - Detected parking space after pressing the simulation button	121
Figure 62 - Raw data from test 5 with the perpendicular parking space	125
Figure 63 - Raw data from test 4 with the angle parking space showing two outliers	126
Figure 64 - Raw data from test 4 with the angle parking space	127

LIST OF PHOTOGRAPHS

Photograph 1 - Steam tractor developed by Joseph Cugnot.	20
Photograph 2 - Miura X8, an iconic brazilian super sport car	23
Photograph 3 - TFmini Plus LiDAR Sensor.....	27
Photograph 4 - ParkNet, a deep neural network developed to detect parking spaces using images	32
Photograph 5 - Ultrasonic sensor HC-SR04.....	33
Photograph 6 - Rotary encoder LPD3806.....	38
Photograph 7 - Double H-bridges L-298N	40
Photograph 8 - Arduino UNO	42
Photograph 9 - Types of parking spaces in the testing scenario	90
Photograph 10 - The initial arrangement of components in the small-scale prototype.....	91
Photograph 11 - Final prototype view	92
Photograph 12 - Voltages at the 5V output of the Arduino.....	111
Photograph 13 - Voltage at the output of the circuit with the devices disconnected.....	113
Photograph 14 - Voltage at the output of the circuit with the sensors and modules connected.....	113
Photograph 15 - Representation of the prototype's position at a small scale at the beginning and end of the tests	123

LIST OF TABLES

Table 1 - Technical specifications of the TFmini Plus sensor.....	27
Table 2 - Specification of Benewake lidar sensors	28
Table 3 - Technical specification of the Arduino UNO	43
Table 4 - Classification table of parking space types	99
Table 5 - Comparative tests of LiDAR and ultrasonic sensors in parallel parking	115
Table 6 - Comparative tests of LiDAR and ultrasonic sensors in perpendicular parking	117
Table 7 - Comparative tests of LiDAR and ultrasonic sensors in angle parking	118
Table 8 - Tests of classification and measurement of parallel parking spaces	123
Table 9 - Tests of classification and measurement of perpendicular parking spaces	124
Table 10 - Tests of classification and measurement of angle parking spaces	125

LIST OF FRAMES

Frame 1 - Other ADAS systems	46
Frame 2 - Changes in gains of the PID controller and their potential effects by properly tuning the PID gains	67
Frame 3 - Research Protocol.....	74

ABBREVIATIONS AND ACRONYMS LIST

ABNT	Associação Brasileira de Normas Técnicas
ACC	Adaptive Cruise Control
ACW	All-round Collision Warning
ADAS	Advanced Driver Assistance System
AEB	Advanced Emergency Braking System
AI	Artificial Intelligence
APA	Advanced Parking Assist
APC	Active Park Assist
APS	Autonomous Parking System
BSD	Blind Spot Detection
FOV	Field of View
LDW	Lane Departure Warning
LKA	Lane Keeping Assist
MATLAB	Matrix Laboratory
RCTA	Rear Cross Traffic Alert
RRT*	Rapidly-exploring Random Tree Star
UTFPR	Universidade Tecnológica Federal do Paraná

SUMMARY

1	INTRODUCTION	16
1.1	Objectives	17
1.1.1	General Objective.....	17
1.1.2	Specific Objectives	17
1.2	Justification	18
1.3	Work structure	18
2	THEORETICAL REFERENCE	19
2.1	History of automobiles.....	19
2.2	Types of parking space.....	24
2.2.1	Parallel	24
2.2.2	Perpendicular	24
2.2.3	Angle	24
2.3	Sensors and components.....	25
2.3.1	Lidar	25
<u>2.3.1.1</u>	<u>TFmini Plus</u>	<u>26</u>
2.3.2	Radar.....	29
2.3.3	Cameras.....	31
2.3.4	Ultrasonic Sensors	32
<u>2.3.4.1</u>	<u>HC-SR04</u>	<u>33</u>
2.3.5	GPS (Global Positioning System).....	34
2.3.6	Inertial Sensors.....	36
2.3.7	Electro-mechanical encoder	37
<u>2.3.7.1</u>	<u>Rotary encoder LPD3806</u>	<u>38</u>
2.3.8	H-bridge.....	39
<u>2.3.8.1</u>	<u>Double H-bridges L-298N.....</u>	<u>40</u>
2.3.9	Microcontrollers	41
<u>2.3.9.1</u>	<u>Arduino UNO</u>	<u>42</u>
2.4	Advanced Driver Assistance System	43
2.4.1	Advanced Emergency Braking	44
2.4.2	All-round Collision Warning	44
2.4.3	Adaptive Cruise Control.....	45
2.4.4	Active Park Assist.....	45

2.4.5	Autonomous Parking System	45
2.4.6	Other systems	46
2.5	Simulation Software	47
2.5.1	MATLAB and Driving Scenario	47
<u>2.5.1.1</u>	<u>Introduction to driving scenario designer</u>	<u>48</u>
2.5.2	Vector's DYNA4.....	50
2.6	Motion Planning.....	51
2.6.1	Classical Approaches	52
2.6.2	Graph Search Approaches	53
<u>2.6.2.1</u>	<u>A* algorithm.....</u>	<u>54</u>
<u>2.6.2.2</u>	<u>RRT*</u>	<u>55</u>
2.6.3	Cost Map	56
2.7	Vehicle Dynamics	57
2.7.1	Coordinate Systems	57
<u>2.7.1.1</u>	<u>Vehicle-fixed coordinate systems</u>	<u>57</u>
<u>2.7.1.2</u>	<u>Earth fixed coordinate system</u>	<u>58</u>
2.7.2	Instantaneous center of rotation (ICR).....	59
2.7.3	Kinematic Bicycle Model.....	60
2.8	Control Strategies	63
2.8.1	Transfer Function	64
2.8.2	Logitudinal Control.....	65
<u>2.8.2.1</u>	<u>Proportional–integral–derivative (PID) control</u>	<u>65</u>
<u>2.8.2.1.1</u>	<u><i>Cruise control</i></u>	<u>67</u>
2.8.3	Lateral Control	69
<u>2.8.3.1</u>	<u>Pure Pursuit.....</u>	<u>69</u>
2.9	Chapter's considerations	72
3	METHODOLOGY	73
3.1	Bibliography Analysis and Literature Review.....	73
3.2	Simulation	75
3.2.1	Software selection for simulation	76
3.2.2	Scenario creation.....	76
3.2.3	Identification of the parking space	77
3.2.4	Maneuver of entering and exiting the parking space	79
3.2.5	Developing a method to create a cost map from the driving scenario	80
3.2.6	Path planning	81

3.2.7	Parallel parking.....	81
3.2.8	Perpendicular parking.....	82
3.2.9	Angled parking.....	82
3.3	Small scale prototype	83
3.3.1	Hardware.....	83
<u>3.3.1.1</u>	<u>Arduino Uno</u>	<u>84</u>
<u>3.3.1.2</u>	<u>Motor's driver.....</u>	<u>84</u>
<u>3.3.1.3</u>	<u>Ultrasonic sensors</u>	<u>85</u>
<u>3.3.1.4</u>	<u>LiDAR sensor</u>	<u>86</u>
<u>3.3.1.5</u>	<u>Rotary encoder.....</u>	<u>87</u>
<u>3.3.1.6</u>	<u>OLED display.....</u>	<u>88</u>
<u>3.3.1.7</u>	<u>Arduino code</u>	<u>89</u>
<u>3.3.1.8</u>	<u>Testing scenario</u>	<u>90</u>
<u>3.3.1.9</u>	<u>Components positioning</u>	<u>91</u>
3.3.2	PC software.....	93
<u>3.3.2.1</u>	<u>Interface</u>	<u>93</u>
<u>3.3.2.2</u>	<u>Data reception</u>	<u>94</u>
<u>3.3.2.3</u>	<u>Parking spot detection.....</u>	<u>95</u>
<u>3.3.2.4</u>	<u>Parking spot measurement.....</u>	<u>96</u>
<u>3.3.2.5</u>	<u>Parking spot classification</u>	<u>99</u>
<u>3.3.2.6</u>	<u>Plotting the data.....</u>	<u>99</u>
<u>3.3.2.7</u>	<u>Saving the data.....</u>	<u>100</u>
3.4	Chapter's considerations	101
4	RESULTS.....	102
4.1	Bibliography Analysis and Literature Review.....	102
4.1.1	Publications per year analysis	102
4.1.2	Keywords analysis.....	103
4.1.3	Analysis of Authors.....	103
4.2	Simulation	104
4.2.1	Parking spot identification.....	105
4.2.2	Path planning	106
4.2.3	Parallel parking.....	107
4.2.4	Perpendicular parking.....	107
4.2.5	Angled parking.....	108
4.3	Small scale prototype	108

4.3.1	Hardware	108
<u>4.3.1.1</u>	<u>Electric motors.....</u>	<u>109</u>
<u>4.3.1.2</u>	<u>Ultrasonic sensors.....</u>	<u>109</u>
<u>4.3.1.3</u>	<u>TFmini Plus LiDAR sensor</u>	<u>109</u>
<u>4.3.1.4</u>	<u>LPD3806 rotary encoder</u>	<u>110</u>
<u>4.3.1.5</u>	<u>Measurement Error.....</u>	<u>110</u>
<u>4.3.1.6</u>	<u>Sensor's comparison.....</u>	<u>114</u>
<i>4.3.1.6.1</i>	<i>Parallel parking spot.....</i>	<i>115</i>
<i>4.3.1.6.2</i>	<i>Perpendicular parking spot.....</i>	<i>116</i>
<i>4.3.1.6.3</i>	<i>Angle parking spot.....</i>	<i>118</i>
<i>4.3.1.6.4</i>	<i>Tests interpretation.....</i>	<i>119</i>
4.3.2	Software	120
<u>4.3.2.1</u>	<u>Interface</u>	<u>120</u>
<u>4.3.2.2</u>	<u>Data reception</u>	<u>122</u>
<u>4.3.2.3</u>	<u>Parking spot detection.....</u>	<u>122</u>
<u>4.3.2.4</u>	<u>Parking spot measurement and classification</u>	<u>122</u>
<i>4.3.2.4.1</i>	<i>Parallel parking spot.....</i>	<i>123</i>
<i>4.3.2.4.2</i>	<i>Perpendicular parking spot.....</i>	<i>124</i>
<i>4.3.2.4.3</i>	<i>Angle parking spot.....</i>	<i>125</i>
4.4	Chapter's considerations	127
5	CONCLUSION	129
	REFERENCES.....	131

1 INTRODUCTION

The history of cars can be traced back to the late 19th century, when German inventor Karl Benz built the first gasoline-powered car in 1885. This was followed by other early pioneers such as Gottlieb Daimler and Henry Ford. In the early 20th century, cars became more widely available and affordable to the general public, with the Model T by Ford becoming one of the most popular and iconic cars of the time (ULRICH, 2011). Over the next few years, increasingly luxurious and exclusive cars were produced, with the second world war being the driving force behind this trend. This trend would only return from the 1970s, which was a period of prosperity, especially for the Brazilian automotive industry, with Brazilian car brands achieving record sales (ALMEIDA, 2016; DIETSCHKE; KUHLGATZ, 2014).

Currently, the industry's focus is once again on personalization and catering to customer desires, as well as a strong trend towards vehicle safety. In this context, the ADAS (Advanced driver-assistance system) has emerged, which are systems developed with a focus on the driver and their safety and comfort.

Among the many skills necessary for the safe driving of a car, parking is a reason for stress and anxiety and is considered one of the biggest challenges for the driver. According to G1 (2016), more than 70% of failures in the exam to obtain the national driver's license (CNH) are caused by the driver's failure to park at the designated location. IBM (2011) shows that more than half of the drivers in 16 of the 20 cities surveyed admitted to getting so frustrated when looking for vacancies that they end up giving up.

This is a problem that has become worse over time, as shown by the data, according to Rodrigue (2020), the number of cars registered in the world exceeded one billion in 2017, which is double the number of cars in the year 2000 and four times the number of cars in the world in 1990.

Indicators such as these demonstrate the importance of developing technologies that help the driver to park in different conditions, small spaces, the presence of intense vehicle traffic, poorly signposted spaces, among others. In this way, one can understand the importance of efficiently using parking spaces, performing maneuvers quickly and safely.

As a result, the development of ADAS has significantly increased, and various new jobs focused on assisting the driver have been developed. In this way, systems

such as the APS (autonomous parking system) are in the spotlight. This is a system that aims to park a car completely autonomously in the three main types of parking: parallel, perpendicular, and angle. Developing an APS requires the use of multiple sensors because they are how data from the car's surroundings can be obtained, such as lidar sensors, radar, ultrasonic, cameras, GPS, and inertial sensors, as well as the use of motion planning methods and longitudinal and lateral control.

This work aims to contribute to the literature with the development of an APS system capable of performing parking in the three different types of spaces, with the main focus being on the simulation carried out in MATLAB software.

1.1 Objectives

This section aims to present the general objective and some specific objectives of the work, which are detailed in sections 1.1.1 and 1.1.2. The general objective seeks to guide the research, and it will be fulfilled at the end of the work through the execution of the proposed specific objectives.

1.1.1 General Objective

- Develop an autonomous parking system using LiDAR sensor.

1.1.2 Specific Objectives

- Identify the types of systems used in the development of autonomous and automatic parking systems through a literature review;
- Develop an algorithm for parking space detection;
- Cost map generation system;
- Adapt a motion planning algorithm for the parking process;
- Design and develop a small-scale prototype of the APS;
- Evaluate the performance of the APS through tests in a controlled environment, analyzing aspects such as accuracy, reliability, and system efficiency;
- Identify challenges and opportunities to improve the performance and functionality of the APS, proposing enhancements and recommendations for future work.

1.2 Justification

The APS is an automatic parking system that uses robotics and artificial intelligence technologies to perform parking maneuvers safely and efficiently, without the need of human intervention. It can be a solution to parking problems in large cities, where parking space is rare and the demand for parking spots is high. Additionally, the system can help reduce the time spent on parking maneuvers and minimize the possibility of accidents caused by human errors (ZIEBINSKI *et al.*, 2017).

The objective of developing an Autonomous Parking System is to design, develop and test an efficient and safe automatic parking system, capable of recognizing and adapting to different types of vehicles, parking spot sizes, and parking conditions. The development of the APS involves the use of technologies such as data processing, control algorithms, and sensors to capture information about the environment and the vehicle. The system should be designed to operate autonomously and ensure the safety of passengers and other parking users. Upon completing the development of the Autonomous Parking System, the following contributions are expected:

- Academic Contribution: Present a systematic study of the main themes present in the topic;
- Economic Contribution: Provide an innovative and effective solution to parking problems;
- Social Contribution: Improve people's quality of life by making the parking process easier, faster, and safer.

1.3 Work structure

Chapter 2 deals with the theoretical background, introducing the history of automobiles, types of parking spaces, commonly used sensors and components in automotive systems. Then, a brief introduction to the software used in simulations and motion planning algorithms is given, followed by an explanation of vehicle dynamics and longitudinal and lateral control systems. Lastly, the concept of ADAS and some of the most important systems in the present are introduced. In Chapter 3, the methodologies used are demonstrated and described. Chapter 4 presents the tests and obtained results. Chapter 5 presents the conclusions and future work.

2 THEORETICAL REFERENCE

This section is the theoretical framework that underpins the methodological choices in this work, which will be divided into eight sections. Section 2.1 will provide a contextualization of the history of the automobile, followed by the types of parking in section 2.2. Section 2.3 will detail the functioning and types of sensors and components relevant to the work. Section 2.4 will introduce the concept of Advanced driver-assistance system (ADAS) and the different levels of vehicle automation. Simulation, Motion Planning, and Vehicle Dynamics topics are being addressed in sections 2.5, 2.6, and 2.7, respectively. In the section 2.8 an introduction to control strategies is given. Finally, section 2.9 will present the final considerations on the chapter.

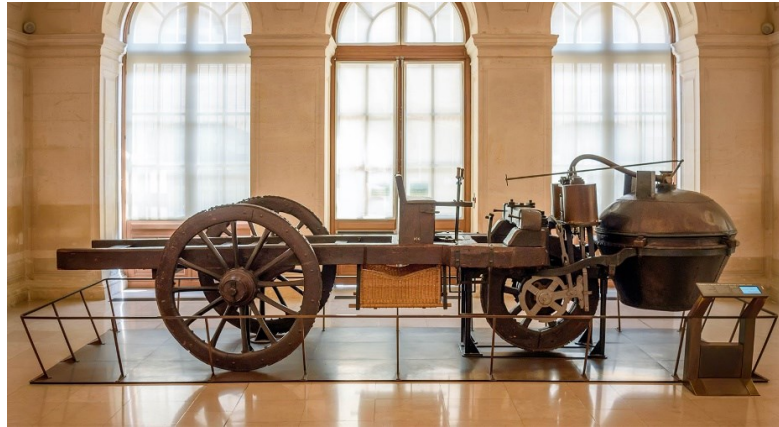
2.1 History of automobiles

Since ancient times, mobility has always been an important characteristic in human evolution. Throughout history, there have been several efforts to transport people over long distances in increasingly shorter periods of time (DIETSCHE; KUHLGATZ, 2014).

The first wheeled vehicles, such as carriages, were created and improved with the addition of steering, suspension, and springs (DIETSCHE; KUHLGATZ, 2014). In the 13th and 15th centuries, writings by Roger Bacon and Leonardo da Vinci presented ideas about self-propelled vehicles. With the evolution of modern industrial society, particularly in Western Europe and the United States, there was a growth in the development of motor vehicles (FLINK, 1990).

In the 17th, 18th, and 19th centuries, European inventors attempted to create self-propelled machines. In 1748, in Paris, the inventor Jacques de Vaucanson demonstrated a carriage driven by a large clockwork motor. At the beginning of the 19th century, Isaac de Rivas developed a hydrogen-powered engine in Paris with manually operated valves and ignition, but there were problems with the motor's synchronization (PURDY; FOSTER, 2023). Various experiments with steam were carried out in the 18th and 19th centuries, including the steam tractor developed by Joseph Cugnot in 1770 with the support of France to pull cannons (FLINK, 1990). The photograph 1 shows Cugnot's tractor.

Photograph 1 - Steam tractor developed by Joseph Cugnot.



Source: Purdy and Foster (2023)

From 1801 to 1803, Richard Trevithick was one of the first to develop high-pressure steam-powered vehicles, and his vehicle was able to reach speeds of about 19 km/h. Oliver Evans, also experimented with a steam-powered dredge in 1805, which reached 6 km/h. However, both Trevithick and Evans encountered difficulties in obtaining funding due to competition with steam trains, which were more reliable and accessible due to the conditions of the roads and the size of the steam engines at the time (FLINK, 1990).

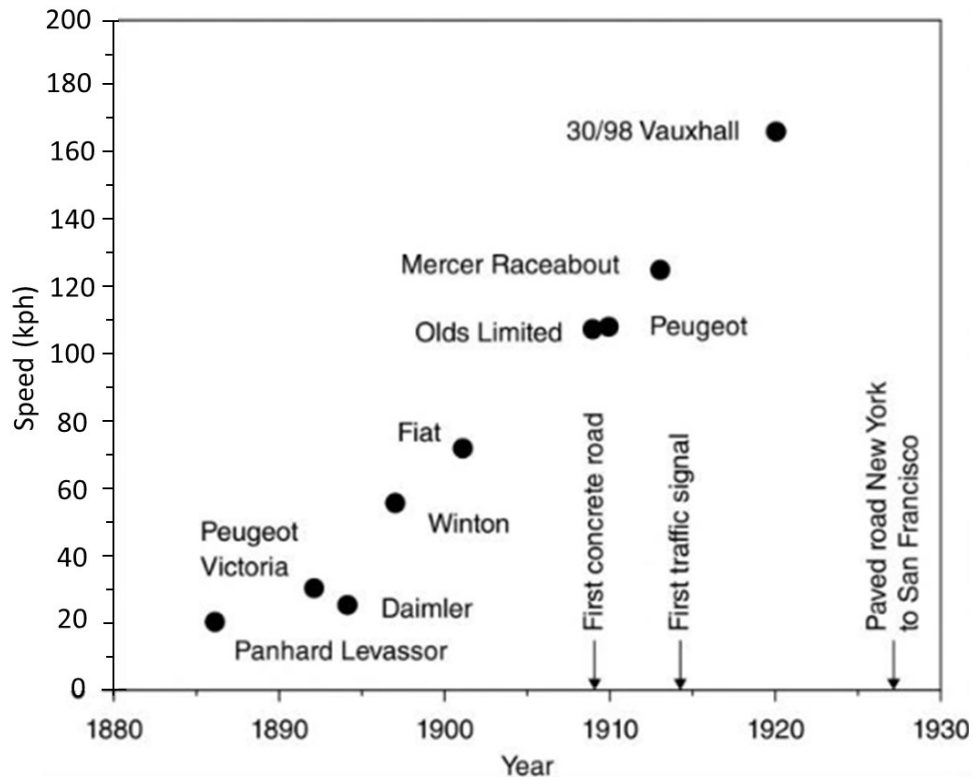
According to ULRICH (2011), in 1886, Carl Benz filed a patent application for his motor-powered tricycle. This action marked the beginning of the accelerated growth of the automotive industry that used internal combustion engines (DIETSCHKE; KUHLGATZ, 2014). The Otto engine patent had already been registered in 1876, while in 1892, Rudolf Diesel filed his patent application for a more efficient version of the engine in Berlin (ULRICH, 2011).

At first, few entrepreneurs saw the potential of automobiles as a viable business option. However, the Frenchmen Panhard and Levassor contributed to the development of the automobile, with Panhard being one of the pioneers in the construction of elements such as the tilted steering wheel, the steering column, the clutch pedal, and the tube-shaped radiator. However, several obstacles arose, including the negative public perception of the pollution and noise caused by the engines. In addition, there was a complete lack of infrastructure for vehicles, such as paved roads, spare parts, and gas stations (DIETSCHKE; KUHLGATZ, 2014).

At the turn of the 19th to the 20th century, electric cars were considered the most promising, given that in 1901 an electric car set the speed record, reaching 100 km/h. In 1900, in the United States, 75 manufacturers produced 4192 motor vehicles,

of which 1688 were steam-powered, 1575 were electric, and only 929 were gasoline-powered (FLINK, 1990; ULRICH, 2011). Figure 1 shows the automobiles with the record for highest top speed over the years.

Figure 1 - Travel speeds of production automobiles



Source: Gillespie (2021)

It was in the 1920s that the gasoline engine gained strength as the preferred solution, thanks to its better efficiency and autonomy compared to electric vehicles, as well as the availability of petroleum-derived fuel at affordable prices at the time (ULRICH, 2011). In the following years, the automobile industry experienced the emergence of many companies, including Peugeot, Citroën, Renault, Fiat, Ford, and many others. The market began to realize the importance of the automobile for society (DIETSCHKE; KUHLGATZ, 2014).

Initially, each vehicle was unique, built entirely by manual laborers (DIETSCHKE; KUHLGATZ, 2014). However, with the arrival of the Model T in 1908 and the implementation of Henry Ford's assembly line in 1913, everything changed (ULRICH, 2011). The Model T revolutionized the American automotive industry, being a car with fewer luxuries and produced in large quantities, making automobiles accessible to the American public (DIETSCHKE; KUHLGATZ, 2014).

Henry Ford vision was to permit that any worker could buy your own automobile and enjoy free time, establishing the bases to the modern consumer society (ULRICH, 2011). Through the automakers Citroën and Opel, Henry Ford's idea was taken to Europe, but will only gain the market acceptance in the early 20's (DIETSCHE; KUHLGATZ, 2014).

At the same time, the adoption of cars in Brazil faced challenges due to poor infrastructure and technical problems related to the operation of vehicles, such as mechanical and electrical issues. In 1919, Ford Motor Company became the first company to establish a factory in the country, located in downtown São Paulo. At the time, the company was facing financial difficulties and saw the Brazilian market as an opportunity to produce and sell its popular Ford T model (ALMEIDA, 2016).

Subsequently, several manufacturers set up their factories in Brazil, such as General Motors in 1925, International Harvester in 1926, and FIAT in 1928. These companies were almost exclusively American due to the impact of World War I and the weakening of the European industry, which allowed Americans to consolidate their position in the Brazilian market (ALMEIDA, 2016).

Manufacturers soon discovered that to stand out in a competitive market, they needed to meet consumers' demands. Thus, victories in races were used as a form of advertising, with drivers proudly displaying their cars' brands. This resulted in an increase in the production of unique and luxurious vehicles, especially during the interwar period, which was marked by some of the most exclusive cars of all time, such as the Mercedes-Benz 500K, the Rolls-Royce Phantom III, and the Bugatti Royale (DIETSCHE; KUHLGATZ, 2014).

The production of more affordable and popular cars was only resumed after the end of World War II. At this time, there was a need for small and cheap vehicles, and that was when the Volkswagen Beetle was designed by Ferdinand Porsche (DIETSCHE; KUHLGATZ, 2014). Its official name was Käfer, with a four-cylinder air-cooled engine and a rear-mounted "boxer" configuration. In 1972, production surpassed 15 million units, becoming the world's most produced vehicle, surpassing the previous record holder, the Ford T (ULRICH, 2011).

The automotive industry responded to the demands of the time with the creation of other models, such as the Citroën 2CV, the Goliath GP 700 and the Fiat 500 C. New standards were developed, considering the incorporation of advanced

accessories and technology, while seeking to maintain a good cost-benefit ratio (DIETSCHKE; KUHLGATZ, 2014).

During the 1960s, 1970s, and 1980s, the national car industry prospered with brands such as Puma, VEMAC, Gurgel, and Gobbi having prominence. Puma, with its sports car popular among young people, saw its sales grow between 1964 and 1979. However, most manufacturers suffered a decline after this period. In 1987, one of the last moments of a fully national automotive industry resurgence, the Miura X8 emerged, an innovative and advanced car with its iconic "gullwing" doors (ALMEIDA, 2016). It can be seen in photograph 2.

Photograph 2 - Miura X8, an iconic brazilian super sport car



Source: Noal (2015)

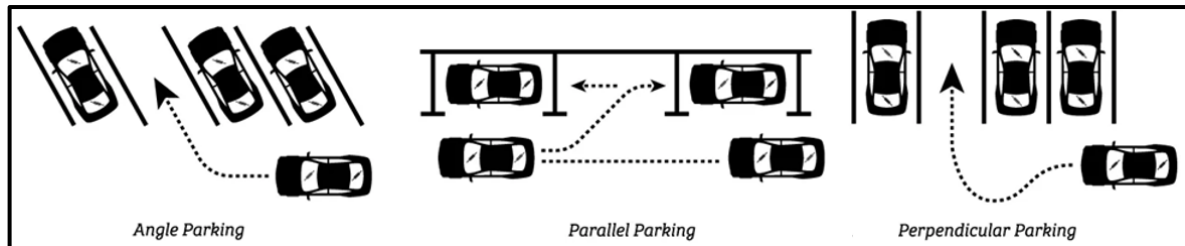
During the 90s, the national automotive sector faced a series of challenges, including competition with foreign companies and lack of government support. As a result, many national companies could not survive and the market was dominated by foreign companies (ALMEIDA, 2016).

Today, the priority is to ensure the safety of vehicles, especially in the face of increased speed and traffic. To meet this need, advanced systems have been developed, such as airbags, ABS, TCS, intelligent sensors, among others (DIETSCHKE; KUHLGATZ, 2014). More recent vehicle models feature some types of safety and driver assistance systems, known as ADAS. These systems can help prevent accidents and protect vehicle occupants. Some examples are AEB (Advanced Emergency Braking System), ACW (Around View Monitor), ACC (Adaptive Cruise Control), and APS (Advanced Parking System), which will be further detailed in section 2.4.

2.2 Types of parking space

There are several different types of parking spaces that a vehicle may encounter in a parking lot or on the street. Some common types of parking spaces include parallel, perpendicular and angle parking, these are shown in figure 2.

Figure 2 - Types of parking spaces



Source: Leremy (2017)

2.2.1 Parallel

Parallel parking spots are spaces in which a vehicle is parked alongside the curb or edge of the road. This type of parking is typically used on streets and requires the driver to maneuver the vehicle into the space by backing into it. To park in a parallel parking space, the driver typically stops the vehicle alongside the space and signals to indicate that they are planning to park. The driver then puts the vehicle in reverse and carefully backs into the space, using the rearview and side mirrors to guide them (MORENCY, CATHERINE; TRÉPANIÉ, 2008) .

2.2.2 Perpendicular

Perpendicular parking spots are spaces in which a vehicle is parked at a 90-degree angle to the curb or edge of the road. This type of parking is typically used in parking lots and is the most common type of parking space. To park in a perpendicular parking space, the driver typically drives the vehicle into the space and stops when the rear bumper is aligned with the painted lines marking the space. The vehicle is then placed in park and the driver exits the vehicle (MORENCY, CATHERINE; TRÉPANIÉ, 2008).

2.2.3 Angle

Angle parking spaces are a delimitation on a street at an angle of 45-degrees with the curb, with the right-hand side of the vehicle nearest the curb and the right-front

wheel approximately 30 cm from the curb. In figure 4 the type of parking space mentioned above can be observed (MORENCY, CATHERINE; TRÉPANIER, 2008).

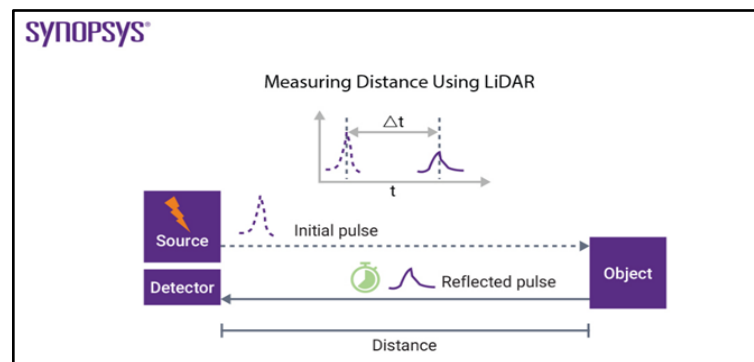
2.3 Sensors and components

Sensors are essential components of an autonomous parking system, as they provide the vehicle with information about its surroundings and enable it to navigate and operate safely and efficiently. There are several types of sensors that are commonly used, each with its own specific characteristics and capabilities. Some examples of sensors used in autonomous parking systems are: Lidar (Light Detection and Ranging), Radar (Radio Detection and Ranging), Cameras, Ultrasonic and GPS (Global Positioning System). The use of sensors is crucial for enabling the vehicle to perceive and understand its surroundings, and to make decisions and take actions accordingly (PALLAS-ARENY, RAMON; WEBSTER, 2012). By combining the data from multiple sensors, the vehicle can create a comprehensive and accurate model of its environment and navigate safely and efficiently.

2.3.1 Lidar

Light Detection and Ranging, also known as LiDAR, is a technology that uses lasers to measure distances. It works by sending out a beam of light and measuring how long it takes for the light to bounce back after it hits an object, as can be seen in figure 3, which was obtained from Synopsys, Inc. This information is then used to calculate the distance to the objects and to create a map of the environment. LiDAR is commonly used in autonomous vehicles to help them sense their surroundings and navigate. Many companies are working on developing smaller, more affordable LiDAR systems for use in autonomous vehicles and advanced driver assistance systems.

Figure 3 - Operation of a lidar sensor



Source: Synopsys (2023)

LiDAR (Light Detection and Ranging) sensors are a key component of autonomous parking systems, as they provide the vehicle with a 3D map of its surroundings and enable it to navigate and park safely and efficiently. Advantages of lidar sensors in the context of autonomous parking systems include:

- 1) Wide range of operation: LiDAR sensors can operate in a wide range of lighting conditions, including complete darkness, making them well-suited for use in parking garages and other enclosed spaces;
- 2) High accuracy and resolution: They can detect objects with high accuracy and resolution, allowing the vehicle to identify and avoid obstacles and navigate with precision;
- 3) Resistant to interference: LiDAR sensors are resistant to interference from external sources, such as sunlight or radio waves, and can provide reliable data even in challenging environments;
- 4) Reliability in different weather conditions: High resistance to interference from rain and extreme weather conditions (Filgueira *et al.*, 2017; Goodin *et al.*, 2019).
- 5) High data rate: Can provide a high data rate, allowing the vehicle to process and analyze a large amount of information in real-time.

However, there are also some disadvantages of lidar sensors:

- 1) Cost: Can be really expensive in comparison with other types of sensors, which may be a limiting factor;
- 2) Size and weight: Are typically larger and heavier than other types of sensors, which may be a concern for some vehicle designs;
- 3) Hard to operate: In comparison with other types of sensors, it requires a larger set of skills.

Overall, lidar sensors could be a component of autonomous parking systems, as they enable the vehicle to perceive and understand its surroundings. However, the high costs and the difficulty to operate the system can be really challenging to make practical and competitive solutions (LI; AL., 2022).

2.3.1.1 TFmini Plus

The TFmini Plus sensor is a short-range, single-point LiDAR sensor that utilizes the Time-of-Flight (ToF) principle to measure the distance to an object. It is based on the TFmini sensor but incorporates several improvements in various aspects, including

measurement frequency, blind zone, accuracy, and stability. It features an IP65 protection rating, which makes it resistant to water and dust (BENEWAKE, [s. d.]). The TFmini Plus sensor finds applications in robotics, drones, intelligent vehicles, security, and industrial control, among others. Photograph 3 presents the sensor.

Photograph 3 - TFmini Plus LiDAR Sensor



Source: Mouser (2023)

The table 1 presents the key specifications of the TF Mini Plus sensor.

Table 1 - Technical specifications of the TFmini Plus sensor





Characteristic	Value
Operating Range	0.1 m ~ 12 m
Accuracy	$\pm 5 \text{ cm}@ (0.1-6 \text{ m}) \pm 1\%@(6 \text{ m}-12 \text{ m})$
Distance Resolution	5 mm
Frame Rate	1-1000 Hz (adjustable)
Light Source	LED 850 nm
Field of View (FOV)	3.6°
Supply Voltage	5 V \pm 0.5 V
Average Current	$\leq 110 \text{ mA}$
Peak Current	500 mA
Power Consumption	550 mW
Communication Level	LVTTTL (3.3 V)
Protection Level	IP65

Source: Benewake (2023)

These features make the TFmini Plus lidar sensor suitable for various applications and scenarios, such as robotics, drones, autonomous vehicles, security, industrial measurement, and level control.

It has the features necessary to the task of measuring a parking space. It also has a superior precision when compared to an ultrasonic sensor, the commonly used sensor in industry for this task. The descriptions of the modules could be seen in table 2, and it is important to note that the enclosure of the sensor offers protection IP67 to the environment, a necessary feature since the sensor in this application is designed to be put outside the vehicle.

Table 2 - Specification of Benewake lidar sensors

Parameter	TF02	TFmini	TFmini Plus	TF03
Picture				
Status	Mass Production	Mass Production	Mass Production	Mass Production
Ranging Indoor	0.4 - 22 m	0.3 - 12 m	0.1 - 12 m	0.1 - 180 m
Ranging Outdoor	0.4 - 10 m	0.3 - 5 m	0.1 - 7 m	0.1 - 70 m
FOV	3°	2.3°	3.6°	0.25°
Precision	CM	CM	CM	CM
Communication Interface	UART/CAN	UART/I2C	UART/I2C	TTL/CAN (RS232/485)
Light Sensitivity	<100k Lux, Sunlight	<70k Lux, Sunlight	<70k Lux, Sunlight	<100k Lux, Sunlight
Weight	52 g	5 g	11 g	77 g
Input Voltage	4.5 V - 6 V	5 V	5 V	5 V
Photobiological Safety	PASS	PASS	PASS	PASS
Protection	IP65	/	IP65	IP67
Dimension	62*39*26 mm	42*15*16 mm	35*28*19 mm	44*42*29 mm

Source: Liu (2019)

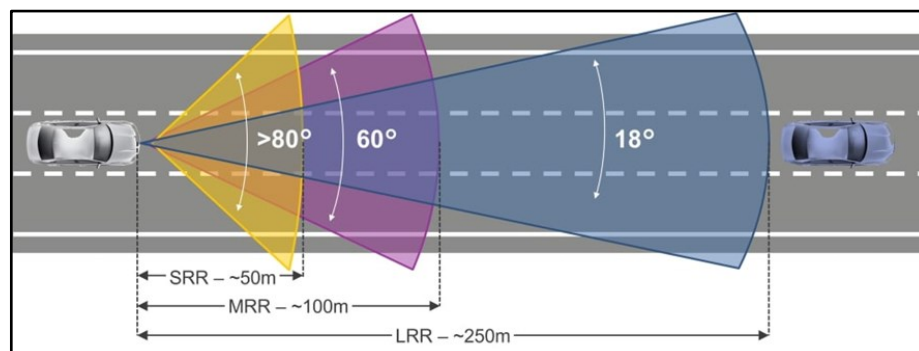
Overall, the TF Mini Plus is a versatile and reliable LiDAR sensor that offers high performance and ease of use. Its small size, low power consumption, and accurate measurements make it ideal for a wide range of applications, from robotics and drones to smart cities and industrial automation.

2.3.2 Radar

Radar (Radio Detection and Ranging) is a technology that uses radio waves to detect and measure the distance, speed, and other characteristics of objects. In an autonomous parking context, radar is used to detect the presence and position of objects around the vehicle, such as other vehicles, pedestrians, and objects in the parking environment. Radar works by emitting a radio frequency signal and then measuring the time it takes for the signal to bounce back after it hits an object. The system uses this information to determine the distance, speed, and other characteristics of the object (PALLAS-ARENY, RAMON; WEBSTER, 2012).

It can be used to detect the presence and position of other vehicles, pedestrians, and other objects in the parking environment. This information can be used by the autonomous vehicle's control system to navigate safely through the parking environment and find a suitable parking spot. The radar can also be used to detect any obstacles or hazards that might be in the vehicle's path, such as pedestrians or other vehicles. Therefore, the system can take appropriate action to avoid them. There are three types of radar sensors used in vehicles: short-range radar (SRR), medium-range radar (MRR) and long-range radar (LRR) (PALLAS-ARENY, RAMON; WEBSTER, 2012). The differences in range can be seen in the figure 4. For parking the best option is the SRR, since the range in parking scenarios is short, not trespassing a few meters.

Figure 4 - Range and field of view of different radar sensors



Source: Vazquez (2022)

Advantages of using radar sensors in an autonomous parking system:

- 1) High accuracy: Can provide highly accurate measurements of distance, speed, and other parameters, which is important for navigating and parking safely in confined spaces;
- 2) Long range: Can detect objects at long distances, making them suitable for detecting obstacles in the surrounding environment;
- 3) All-weather performance: Are not affected by weather conditions such as rain or fog, making them suitable for use in outdoor parking environments;
- 4) High resolution: Can provide high-resolution images of the objects they detect, making them useful for detecting and avoiding obstacles.

Some disadvantages found in the use of radar sensors:

- 1) Cost: Can be expensive, especially those with high accuracy and long range;
- 2) Interference: Can be affected by other sources of radio frequency interference, such as cell phone towers or other radar systems, which can affect their accuracy;
- 3) Limited imaging capabilities: While radar sensors can provide high-resolution images, they are generally not as detailed as those produced by other types of sensors, such as cameras;
- 4) Limited penetration: Radar signals do not penetrate certain materials, such as metal or concrete, which can limit their effectiveness in certain environments;
- 5) Target classification: due to the reflectivity, shape and size and other factors, the radar sensors need complex algorithms to correctly distinguish between different types of objects.

It is understood that radar sensors have multiple advantages in their use in an autonomous parking system, such as high resolution in object detection and high reliability in adverse weather conditions. However, it is a costly type of sensor, and several modules would be needed for its use in an autonomous parking system, greatly increasing the cost of the project. If cheaper acquisition options are found that perform well at short distances, this sensor may become crucial in the data acquisition process.

2.3.3 Cameras

Camera sensors can be used in autonomous parking systems to help vehicles navigate and park in confined spaces. They are able to capture images of the surrounding environment and provide the vehicle with visual information about its surroundings (PALLAS-ARENY, RAMON; WEBSTER, 2012). There are several advantages to using camera sensors in autonomous parking systems:

- 1) High resolution: Camera sensors can provide high-resolution images of the surrounding environment, allowing the vehicle to detect and avoid obstacles with greater accuracy;
- 2) Wide field of view: Can have a wide field of view, allowing the detection of a large area around it and detect potential obstacles in its path;
- 3) Good performance in low light: Some camera sensors are able to perform well in low light conditions, making them suitable for use in poorly lit parking garages or at night.

However, there are also some limitations to using camera in parking applications:

- 1) Limited range: Camera sensors generally have a shorter range than radar sensors, making them less suitable for detecting obstacles at long distances;
- 2) Sensitivity to weather: Can be affected by weather conditions such as rain or fog, which can reduce their accuracy and reliability;
- 3) Dependence on external lighting: May require external lighting to function properly, which may not be available in all parking environments.

In summary, camera sensors can be a useful tool for autonomous parking systems, many bibliographical works use cameras and neural networks to identify suitable parking spaces to the vehicle, in photograph 4 it is possible to see an example. However, the advantages of using cameras may not be sufficient on their own and may need to be used in combination with other types of sensors to provide a complete picture of the surrounding environment (PALLAS-ARENY, RAMON; WEBSTER, 2012).

Photograph 4 - ParkNet, a deep neural network developed to detect parking spaces using images

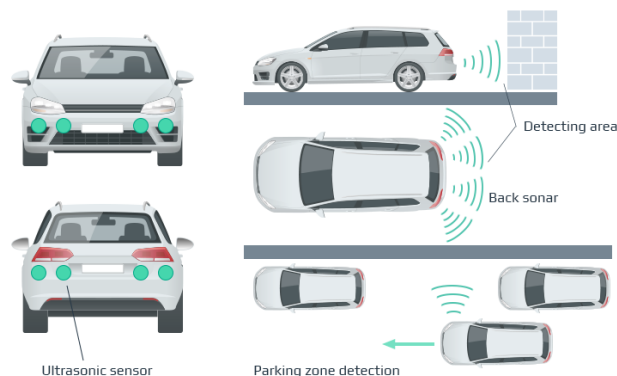


Source: Cvijetic (2019)

2.3.4 Ultrasonic Sensors

Ultrasonic sensors are devices that use high-frequency sound waves to measure distance or detect objects. In the context of an autonomous parking system, ultrasonic sensors can be used to detect the presence of other vehicles or obstacles in the parking environment. One of the main advantages is that they can operate in a variety of lighting conditions, including low light or darkness. This makes them a useful tool for navigating through parking garages, which may not always be well lit. Ultrasonic sensors are also relatively inexpensive and easy to install. They can be mounted on the front, rear, and sides of the vehicle to provide 360-degree coverage and help the vehicle navigate safely through a parking garage for example (PALLAS-ARENY, RAMON; WEBSTER, 2012). The figure 5 shows an example of a car with 8 ultrasonic sensors mounted in the front and back of the vehicle and one in the lateral side.

Figure 5 - Car with ultrasonic sensors and their positions



Source: Intellias Mobility (2018)

However, there are also some limitations to using these sensors in an autonomous parking system. One of the main disadvantages is that they may not be

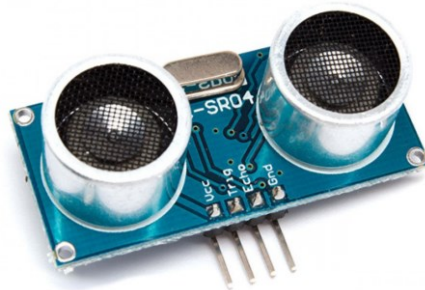
as accurate as other types of sensors, such as laser sensors, in certain situations. They may also have difficulty detecting objects that are too close or too far away. Another limitation is that they may not be able to detect certain types of objects, such as those that are transparent or highly reflective. This can make it difficult for the autonomous parking system to navigate safely (CARULLO *et al.*, 2001).

Overall, ultrasonic sensors are a useful tool, as they can provide valuable information about the environment and help the vehicle navigate safely through the parking garage. Even so, they should be used in conjunction with other types of sensors to provide a complete picture of the parking environment and ensure the safety of the autonomous vehicle (PALLAS-ARENY, RAMON; WEBSTER, 2012).

2.3.4.1 HC-SR04

The HC-SR04 is an inexpensive and widely used ultrasonic sensor module that measures distance by sending out ultrasonic waves and receiving their echoes. The sensor is easy to use and can be interfaced with microcontrollers like Arduino, Raspberry Pi, and other embedded systems. The photograph 5 show the sensor.

Photograph 5 - Ultrasonic sensor HC-SR04



Source: Piborg (2023)

It works on the principle of sonar, which is similar to the echolocation system used by bats and dolphins to navigate and locate prey. It has a transducer that sends out a high-frequency sound pulse, typically at 40 kHz, which travels through the air until it hits an object. The pulse then bounces back from the object and is detected by the receiver on the sensor module. The time taken for the pulse to travel to the object and back is proportional to the distance between the sensor and the object. The distance can be calculated by measuring the time delay between the transmitted and received pulse.

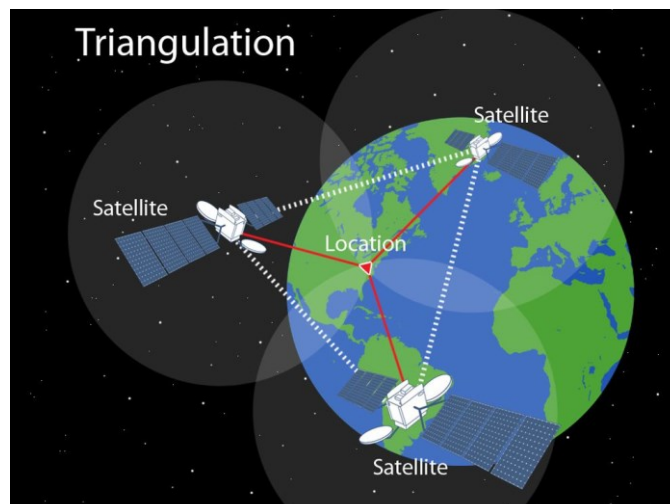
The HC-SR04 sensor has the following technical specifications:

- Operating Voltage: 5V DC
- Operating Current: 15mA
- Operating Frequency: 40kHz
- Measurement Range: 2cm to 400cm
- Measurement Accuracy: ± 0.3 cm
- Trigger Pulse Width: 10 μ s
- Echo Pulse Output: TTL level signal output proportional to the distance
- Dimensions: 45mm x 20mm x 15mm

2.3.5 GPS (Global Positioning System)

GPS is a satellite-based navigation system that uses a network of satellites orbiting the Earth to determine the location, speed, and direction of an object on the Earth's surface. It is widely used for navigation and tracking purposes, including in car navigation systems, smartphones, and other devices. The GPS sensor works by receiving signals from a network of satellites orbiting the Earth. Each satellite transmits a radio signal that includes the satellite's current position and the time the signal was transmitted. The sensor receives these signals and uses them to calculate the distance between the satellite and the device. By measuring the distance between the device and at least three satellites, the GPS sensor can determine the device's position on the Earth's surface using a process called trilateration or triangulation (PALLAS-ARENAY, RAMON; WEBSTER, 2012). This process is shown in figure 6.

Figure 6 - GPS triangulation process



Source: Gunther (2022)

In addition to determining the location, the GPS sensor can also be used to determine the speed and direction of the device. By continuously tracking the location of the device over time, the GPS sensor can determine the speed at which the device is moving and the direction it is headed. This information can be used in applications such as tracking the movement of a vehicle or providing turn-by-turn navigation instructions. In the context of an autonomous parking system, the GPS sensor is used to determine the location of the vehicle as it moves through the parking lot. This information is used to guide the vehicle to an available parking space and to navigate to the desired parking location within the space (PALLAS-ARENY, RAMON; WEBSTER, 2012). Advantages of using a GPS sensor:

- 1) High accuracy: GPS sensors can provide very accurate location information in ideal conditions, which is important for precise navigation and maneuvering in a parking lot;
- 2) Wide coverage: Can be received almost anywhere, making it possible to use these sensors in a variety of environments, including outdoor parking lots;
- 3) Cost-effective: GPS sensors are relatively inexpensive compared to other types of sensors, such as LIDAR or radar;
- 4) Easy to integrate: It's a type of sensor that are readily available and easy to integrate into existing systems.

Some disadvantages:

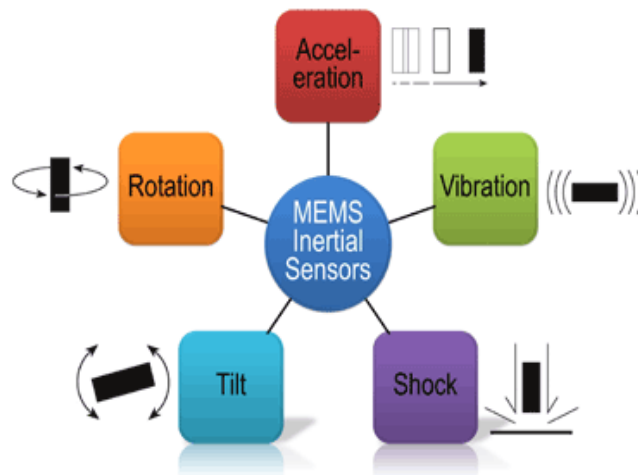
- 1) Dependence on satellite signals: GPS sensors rely on satellite signals to function, which can be disrupted by certain environmental conditions such as heavy cloud cover or tall buildings;
- 2) Limited accuracy in certain conditions: Accuracy can be affected by factors such as atmospheric conditions and the presence of buildings or other structures, which can affect the quality of the satellite signals received by the sensor;
- 3) Potential for hacking: GPS signals can be vulnerable to hacking, which could potentially compromise the accuracy and reliability of the sensor;
- 4) Limited range: They have a limited range, which may not be sufficient for certain applications, such as indoor parking garages or multi-level parking lots.

GPS sensors play a fundamental role in autonomous vehicles as they not only provide the vehicle's positioning but can also assist in calculating the speed and direction of movement. They are a powerful tool when used in conjunction with other sensors to obtain accurate location through sensor fusion (PALLAS-ARENY, RAMON; WEBSTER, 2012).

2.3.6 Inertial Sensors

Inertial sensors could measure acceleration, orientation, and angular velocity. They are typically used in applications where precise measurement of these quantities is required, such as in inertial navigation systems, aircraft and missile guidance systems, and virtual reality headsets. There are several types of inertial sensors, including accelerometers, gyroscopes, and magnetometers. Accelerometers measure acceleration, which is the rate of change of velocity over time. Gyroscopes measure angular velocity, which is the rate of change of orientation over time. Magnetometers measure the strength and direction of the magnetic field. In the figure 7 it is possible to observe the types of movements that can be measured.

Figure 7 - Types of movements measure by inertial sensors



Source: Electronic Products (2011)

Inertial sensors work by using physical principles such as Newton's laws of motion and the conservation of angular momentum. For example, an accelerometer may use a mass suspended on a spring that is displaced when the accelerometer is subjected to acceleration. The displacement of the mass is then measured and used to calculate the acceleration. In ADAS, inertial sensors are used in conjunction with

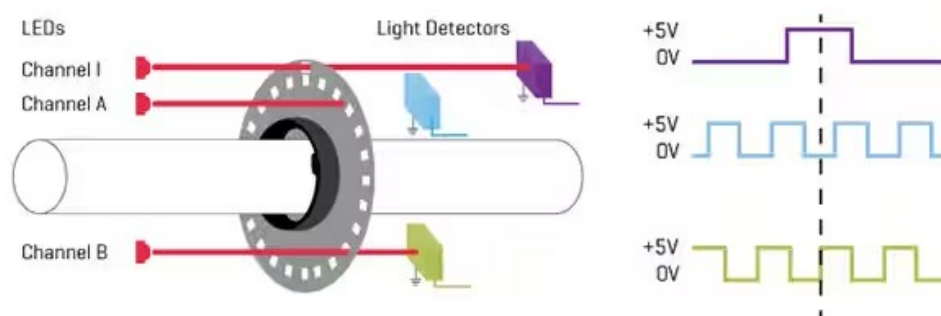
other sensors such as cameras, lasers, and radar to provide a complete picture of the surrounding environment. This information is used by the autonomous parking system to navigate the vehicle safely and accurately through the parking lot and into a parking space (PALLAS-ARENY, RAMON; WEBSTER, 2012). To summarize, the use of inertial sensors in an autonomous parking system allows the vehicle to move smoothly and precisely, ensuring a safe and efficient parking experience.

2.3.7 Electro-mechanical encoder

Electro-mechanical encoders are devices used to convert rotational or linear motion into electrical signals, allowing measurement and control of position, velocity, and rotational direction of an axis or mechanical system. There are different types of electro-mechanical encoders, including incremental, absolute, and magnetic encoders, each with its own specific features and applications.

Incremental encoders generate electrical pulses in response to mechanical motion, providing information about the velocity and direction of axis rotation. They are often used in applications requiring real-time position and velocity control, such as CNC machines, industrial robotics, printers, among others (DIGI-KEY ELECTRONICS, 2020). In figure 8 is possible to see an example.

Figure 8 - Incremental encoder



Source: Digi-Key Electronics (2020)

Absolute encoders provide precise information about the angular or linear position without the need for an initial reference position. These encoders are widely used in applications requiring high precision and reliability in measurement, such as motion control systems in aircraft, satellites, telescopes, and other high-precision equipment (DIGI-KEY ELECTRONICS, 2020).

Magnetic encoders, on the other hand, use magnetic sensor technology to detect the axis position and are commonly used in harsh industrial environments where dust, humidity, or vibrations may affect the reliability of other types of encoders. (PALLAS-ARENY, RAMON; WEBSTER, 2012).

There are various types of encoders used in different domains and applications. One common type is the digital encoder, which converts analog signals into digital representations, enabling their processing and manipulation within digital systems. Another widely used type is the audio/video encoder, which converts analog audio or video signals into compressed digital formats suitable for transmission or storage (PALLAS-ARENY, RAMON; WEBSTER, 2012).

2.3.7.1 Rotary encoder LPD3806

The LPD3806 rotary encoder is a high-precision sensor that can be used to measure the position and speed of rotating shafts in a wide range of applications. The LPD3806 encoder is widely used in robotics, automation, and machine control systems due to its high accuracy, resolution, and durability. The encoder works by using an optical sensor to detect the rotation of a disc with evenly spaced slots. As the disc rotates, the slots pass through the optical sensor, which detects the changes in the amount of light passing through the slots. The encoder then converts the changes in light into electrical signals, which are used to measure the position and speed of the rotating shaft (OGUNTOSIN; AKINDELE, 2019). The photograph 6 show the actuator.

Photograph 6 - Rotary encoder LPD3806



Source: Doublehero (2023)

The LPD3806 rotary encoder has the following technical specifications:

- Operating Voltage: 5V DC;
- Output Signal: A, B, and Z phase signals;
- Maximum Speed: 6000 RPM;
- Maximum Resolution: 600 PPR (pulses per revolution);
- Operating Temperature: -10°C to 70°C;
- Protection Class: IP65 (dust-proof and water-resistant);
- Dimensions: 38mm x 38mm x 28mm.

2.3.8 H-bridge

An H-bridge is an essential electronic circuit configuration commonly employed in power electronics and motor control applications. It consists of four switching elements arranged in the shape of an "H," from which it derives its name. The purpose of the H-bridge circuit is to facilitate bidirectional control over current flow through a load, such as a motor or an actuator, by enabling or disabling the switching elements in a specific pattern (IMAN-EINI *et al.*, 2009).

The primary function of an H-bridge is to provide a means of controlling the direction and magnitude of current flowing through the load. By appropriately activating the switching elements in the H-bridge, the voltage polarity and magnitude applied to the load can be manipulated, resulting in forward or reverse motor rotation, for instance (IMAN-EINI *et al.*, 2009). This bidirectional control capability makes H-bridges particularly valuable in applications that necessitate precise speed and direction control, including robotics, electric vehicles, and industrial automation.

Typically, the four switching elements in an H-bridge are solid-state devices, such as transistors or MOSFETs, capable of handling high currents and voltages. Two of these switches, known as high-side switches, are connected to the positive supply voltage, while the other two, called low-side switches, are connected to the ground or negative supply voltage. By selectively turning on and off the appropriate combination of switches, the H-bridge can effectively reverse the voltage polarity across the load, allowing for bidirectional current flow (IMAN-EINI *et al.*, 2009).

To ensure proper operation and prevent short circuits, H-bridges often incorporate additional control circuitry. This circuitry includes gate drivers, which provide the necessary signals to drive the switching elements and protect against shoot-through, a condition in which both high-side and low-side switches are momentarily activated simultaneously. Furthermore, H-bridges may feature braking

and freewheeling diodes to suppress voltage spikes and safeguard the switches from reverse current flow during switching transitions (IMAN-EINI *et al.*, 2009).

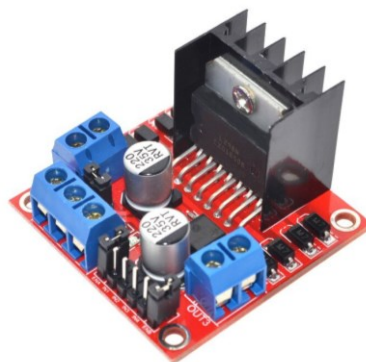
The control of an H-bridge is typically achieved through a microcontroller or a dedicated motor control circuit. By supplying appropriate signals to the gate drivers, the microcontroller can control the timing and sequence of switching events, enabling precise speed and direction control of the load. Moreover, advanced designs may incorporate additional features like pulse width modulation (PWM) for smooth speed control or current sensing for monitoring and protection purposes (IMAN-EINI *et al.*, 2009).

In summary, the H-bridge is a critical circuit configuration widely utilized for bidirectional current control in power electronics and motor control applications. By selectively activating the switching elements, the H-bridge facilitates precise control over voltage polarity and magnitude across a load (IMAN-EINI *et al.*, 2009). Its versatility and effectiveness make it an indispensable component in numerous systems requiring reversible motor control and precise manipulation of power flows.

2.3.8.1 Double H-bridges L-298N

The L298N is a popular motor driver IC that can be used to control the speed and direction of DC motors and stepper motors. The L298N IC is widely used in robotics and automation projects due to its ease of use and reliability. It works by using two H-bridges to control the flow of current through the motor. Each H-bridge consists of four MOSFETs, which can be switched on and off to change the direction of current flow through the motor. By changing the state of the MOSFETs, the L298N can control the speed and direction of the motor. A picture of it is shown in photograph 7.

Photograph 7 - Double H-bridges L-298N



Source: Instituto Digital (2023)

The L298N motor driver IC has the following technical specifications:

- Operating Voltage: 5V to 35V DC;
- Maximum Output Current: 2A per channel (with heat sink);
- Peak Output Current: 3A per channel;
- Logic Voltage: 5V DC;
- Number of H-bridges: 2;
- Maximum Power Dissipation: 25W;
- Dimensions: 20mm x 58mm x 15mm.

2.3.9 Microcontrollers

Microcontrollers are integrated electronic devices that combine a microprocessor, memory, and peripherals on a single chip. They are widely used in various applications due to their processing capabilities, ease of programming, and low power consumption (MAZIDI; MCKINLAY; CAUSEY, 2008). These devices provide an efficient solution for real-time control and system monitoring, enabling process automation and interaction with the external environment.

The architecture of microcontrollers varies, with Harvard and Von Neumann architectures being common. The Harvard architecture separates the program memory from data memory, allowing simultaneous access to instructions and data. On the other hand, the Von Neumann architecture uses a single memory to store both instructions and data (AYALA, 1996). Additionally, microcontrollers feature integrated peripherals such as input/output ports, analog-to-digital converters, timers, and communication interfaces, enabling interaction with the external environment.

Programming is typically carried out using high-level programming languages such as C or C++, which are compiled into machine code compatible with the target microcontroller. Integrated Development Environments (IDEs), such as Arduino IDE and MPLAB X, provide tools and libraries to facilitate the programming process (PREDKO, 2008). Microcontroller programming involves defining input/output pins, configuring peripherals, and implementing control algorithms.

Microcontrollers have been rapidly evolving, driven by advancements in semiconductor technology. Recent advances include increased processing power, enhanced peripheral integration, reduced power consumption, and support for short-range wireless communications such as Bluetooth and Wi-Fi. Additionally, the

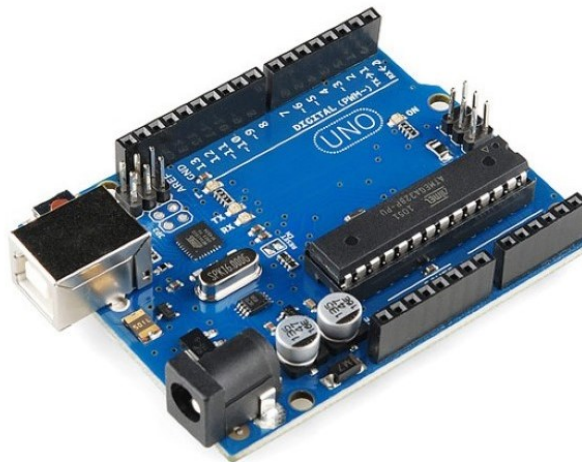
development of open-source platforms like Arduino and Raspberry Pi has democratized access to microcontrollers, enabling more people to explore their applications and create innovative projects (BANZI, 2008).

2.3.9.1 Arduino UNO

The Arduino UNO is a widely adopted microcontroller board that is based on the ATmega328P microcontroller, belonging to the Atmel AVR microcontroller family (ARDUINO, 2023). This board is recognized as one of the most popular and well-documented in the Arduino platform, which encompasses a comprehensive set of hardware and software tools for the development of interactive electronic projects.

Developed to offer ease of use to both beginners and professionals, the Arduino UNO enables rapid prototyping and integration with various sensors, actuators, and communication modules. For programming, it is possible to use a language based on C/C++ and an Integrated Development Environment (IDE) compatible with different operating systems such as Windows, Linux, and Mac OS X. Additionally, the board can be programmed through a web interface or via command line. Photograph 8 presents the Arduino UNO module.

Photograph 8 - Arduino UNO



Source: Maker Hero (2023)

Below is presented table 3 specifying the characteristics of the Arduino UNO module.

Table 3 - Technical specification of the Arduino UNO

Characteristics	Value
Microcontroller	ATmega328P
Digital Input/Output Pins	14 (of which 6 can be used as PWM outputs)
Analog Input Pins	6
Crystal oscillator	16 Mhz
Input Voltage (nominal)	7-12 V
Output Voltage	5V
Maximum Current per I/O Pin	20 mA
Maximum Current	200 mA
Clock Speed	16 MHz
Memory	2KB SRAM, 32KB FLASH, 1KB EEPROM

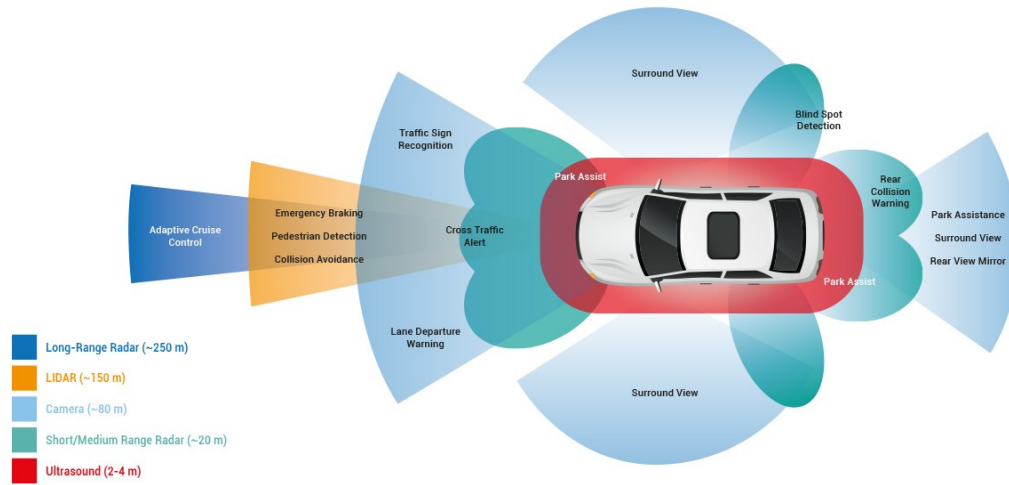
Source: Arduino (2023)

2.4 Advanced Driver Assistance System

With the aim of improving safety and efficiency in driving, the automotive industry has been investing more and more in ADAS technologies. These systems use sensors, cameras, radars, and software to collect information about the vehicle's environment and help the driver make informed decisions to avoid accidents (RAJAMANI, 2011). With the increasing number of traffic accidents worldwide, ADAS systems have become an important tool to reduce collisions and save lives. In this perspective, it is essential to understand how these systems work and their importance in the current context of the automotive industry. Figure 9 presents various ADAS applications and their respective sensors. According to Synopsis (2023), some of these applications are essential for safety and include:

- Pedestrian detection and avoidance;
- Departure warning/correction;
- Recognition of traffic signs;
- Automatic emergency braking;
- Blind spot detection.

Figure 9 - ADAS functions and sensors



Source: Texa (2023)

These systems are the key to the success of ADAS applications. They incorporate the latest interface standard models and use multiple vision-based algorithms to support real-time multimedia, vision co-processing, and sensor fusion subsystems. The modernization of ADAS applications is the first step towards understanding autonomous vehicles (RAJAMANI, 2011).

2.4.1 Advanced Emergency Braking

Also known as AEB aims to brake the vehicle or alert the driver in order to avoid potential collisions. The system can use various types of sensors, such as vision, radar, and Lidar (ZIEBINSKI *et al.*, 2017), that monitor the distance between the vehicle and other objects or vehicles on the road, thus having the potential to save lives and reduce serious injuries.

2.4.2 All-round Collision Warning

All-round Collision Warning stands for ACW, also known as a collision warning system in all directions. ACW is an automotive safety technology that uses sensors, such as radar or cameras, to monitor the area around the vehicle and alert the driver to possible collision risks, including vehicles, objects, and pedestrians approaching the vehicle from any direction. This technology is particularly useful in parking maneuvers as it helps to avoid collisions with other vehicles or objects that may be outside the driver's field of view (RAJAMANI, 2011).

2.4.3 Adaptive Cruise Control

Adaptive Cruise Control, or ACC, is a vehicle safety feature that uses radar or camera technology to maintain a safe distance between the user's vehicle and the vehicle in front, automatically adjusting the speed of the vehicle according to traffic flow. ACC is an extension of traditional cruise control, which allows the driver to set a constant speed for the vehicle. However, this system is able to monitor the distance and speed of the vehicle in front and adjust the speed of the user's vehicle accordingly, in order to maintain a safe distance and prevent collisions. ACC is an important ADAS technology that can help improve road safety, reduce accidents, and driver fatigue (ZIEBINSKI *et al.*, 2017).

2.4.4 Active Park Assist

In the automotive context, APC stands for "Active Park Assist", which is a safety feature that uses ultrasound sensors and cameras to help the driver park the vehicle more easily and accurately. When activated, the APC system can detect available parking spaces around the vehicle and then guide the driver through visual and audible instructions to park the vehicle in the space. The APC can also automatically control the vehicle's steering wheel during the parking maneuver, allowing the driver to focus on controlling the accelerator, brake, and transmission. The APC is a useful technology for drivers who have difficulty parking in tight spaces or to avoid collisions during parking maneuvers, and it is a common feature in newer vehicles and is often included as part of an ADAS feature package (RAJAMANI, 2011).

2.4.5 Autonomous Parking System

Also known as APS, is a car safety feature that uses sensors, cameras, and software to help the driver park the vehicle more easily and accurately. APS includes various features, such as "Automatic Parking Assistant" (or Park Assist), which allows the vehicle to park automatically in a spot identified by the system. Park Assist typically uses ultrasonic sensors to detect the size of the spot. Another feature of APS is the "Exit Parking Assistant" (or Exit Assist), which alerts the driver about the presence of pedestrians, cyclists, or other vehicles when leaving a parking spot. Exit Assist can also monitor approaching traffic and alert the driver about possible collisions during the exit from the parking spot.

2.4.6 Other systems

In addition to the systems mentioned in the subsections of section 2.4, there are also several others that are discussed in the literature. Some examples are present in the works of (Choi *et al.*, 2016; Kiencke & Nielsen, 2005; Rajamani, 2011; Synopsis, 2023; Ziebinski *et al.*, 2017), these are presented in the works listed in frame 1:

Frame 1 - Other ADAS systems

System	Acronym	Description
Lane Departure Warning	LDW	This system uses sensors to detect the lane marking lines on the road, and if the vehicle deviates from its lane, the system issues an audible or visual warning to alert the driver.
Lane Keeping Assist	LKA	This system works in conjunction with the LDW to correct the vehicle's deviation from the lane by applying a small torque to the steering wheel to keep the vehicle in the correct position.
Blind Spot Detection	BSD	This system uses sensors to detect the presence of other vehicles in the driver's blind spot. The system issues an audible or visual warning to alert the driver to the presence of the vehicle in the blind spot.
Rear Cross Traffic Alert	RCTA	This system uses sensors to detect vehicles approaching the car when it is backing out of a parking spot. The system issues an audible or visual alert to warn the driver about the approaching vehicle.
Advanced Parking Assist	APA	This system helps the driver to park the vehicle more easily and accurately, using sensors to detect the size of the parking space and guide the driver through visual and auditory instructions to park the vehicle properly.

Source: Adapted from Choi *et al.* (2016), Kiencke & Nielsen (2005), Rajamani (2011), Synopsis, (2023) and Ziebinski *et al.* (2017)

These are just a few examples of the types of ADAS systems that exist, and new technologies are constantly being developed to help improve safety on the roads.

2.5 Simulation Software

Software plays a crucial role in the simulation of autonomous parking systems. It is used to create and run the virtual environment in which the simulation takes place, as well as to control the behavior and movements of the simulated autonomous vehicle. There are several types of software that may be used in the simulation of autonomous parking systems. One type is computer-aided design (CAD) software, which is used to create a virtual model of the parking lot or garage in which the simulation will take place. This may include the layout of the space, the location of parking spaces, and the presence of any obstacles or hazards.

Another type of software that may be used is simulation software, which is used to run the simulation itself. This software typically includes algorithms and logic to control the movements of the simulated autonomous vehicle, as well as sensors and other data inputs to help the vehicle navigate and avoid obstacles. Finally, analysis software may be used to review the results of the simulation and identify any areas for improvement. This may include analyzing data on the vehicle's speed, accuracy of parking, and other performance metrics. Overall, the use of software in the simulation of autonomous parking systems allows for the testing and optimization of these systems in a controlled and safe environment before they are deployed in real-world scenarios.

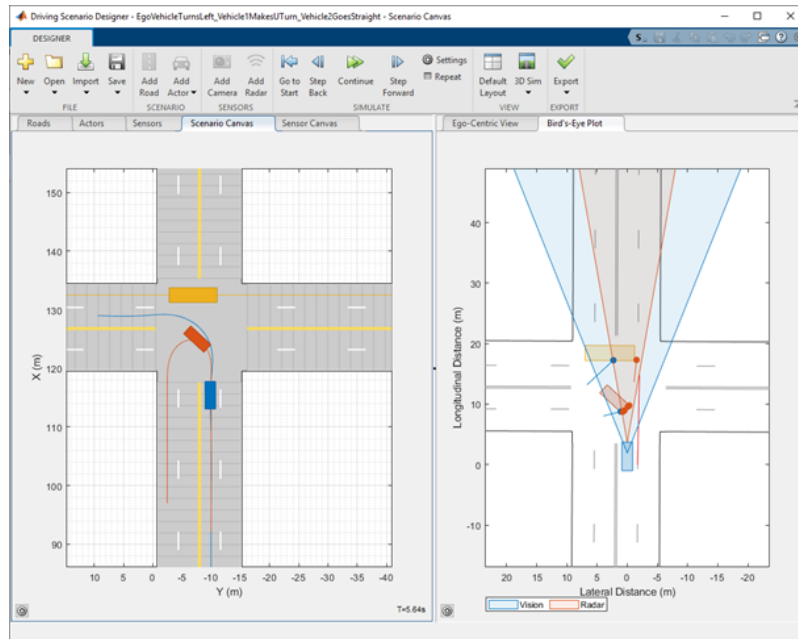
2.5.1 MATLAB and Driving Scenario

MATLAB (MATrix LABoratory) is a high-level programming language and computing environment developed by MathWorks. It is widely used in the field of engineering and scientific computing, particularly in areas such as image and signal processing, control systems design, and data analysis. The software includes a wide range of tools and functions that can be used to perform various tasks such as data visualization, numerical computation, and model-based design. It also provides a flexible and interactive programming environment with features such as debugging, code highlighting, and interactive graphics.

In the context of a driving scenario, MATLAB can be used to analyze and simulate various aspects of the scenario, such as vehicle dynamics, traffic flow, and sensor data. For example, it can be used to model and analyze the behavior of vehicles and pedestrians in a traffic environment, or to design and test control systems for

autonomous vehicles. This is possible using the driving scenario module in MATLAB, a toolbox that provides a set of functions and tools for modeling and simulating driving scenarios. The figure 10 exemplifies its use showing a car, represented in blue, which contains a vision sensor and a radar, capable of identifying objects in front of the car.

Figure 10 - Driving Scenario Designer with a project running



Source: Mathworks (2023)

It includes a library of vehicle, road and traffic models that can be used to create and simulate a variety of driving scenarios. The module also provides tools for visualizing and analyzing the results of the simulation, such as plotting vehicle trajectories and generating performance metrics. On this car the multiPlus identification points of the sensors are shown. These signals are very useful bases for the development of autonomous automotive systems, considering the easy modeling obtained from often complex systems. Overall, the driving scenario module in MATLAB is a useful tool for researchers and engineers working on the development and analysis of driving systems and technologies. It can be used to design, test, and optimize the performance of control systems, sensors, and other components of a driving system, and to evaluate the behavior and performance of a vehicle in various driving scenarios.

2.5.1.1 Introduction to driving scenario designer

Driving Scenario Designer is a powerful and versatile tool in the field of autonomous vehicle simulation and development. It is a specialized add-on for the

widely-used Matlab software, developed by MathWorks, that enables engineers, researchers, and developers to create realistic driving scenarios for a variety of applications.

In the pursuit of safer and more efficient autonomous driving systems, simulations play a crucial role in testing and validating algorithms and control strategies. The Driving Scenario Designer empowers users to design complex and dynamic driving scenarios that mimic real-world conditions, providing a safe and controlled environment to assess the performance of autonomous vehicles.

This tool offers a user-friendly graphical interface that allows users to define various elements of a driving scenario, such as roads, vehicles, pedestrians, traffic signs, and traffic signals, among others. Users can customize the behavior and motion of each element, allowing for the simulation of diverse traffic scenarios, urban environments, highways, and more.

With the Driving Scenario Designer, users can explore different traffic scenarios, test the performance of autonomous systems under various conditions, and evaluate the robustness of control algorithms. Additionally, the software facilitates the integration of simulated scenarios with control algorithms and vehicle models, creating a comprehensive simulation environment for testing and development.

It is also possible to understand the operation of the created scenario through the code, optimizing the development of scenarios and allowing the use of programmable logic to perform actions directly in the scenario. Figure 11 represents the respective code generated by exporting the scenario.

Figure 11 - Code generated from a driving scenario in MATLAB

```

1 function [scenario, egoVehicle] = createDrivingScenario()
2 % createDrivingScenario Returns the drivingScenario defined in the Designer
3
4 % Generated by MATLAB(R) 9.13 (R2022b) and Automated Driving Toolbox 3.6 (R2022b).
5 % Generated on: 20-Dec-2022 10:57:52
6
7 % Construct a drivingScenario object.
8 scenario = drivingScenario;
9
10 % Add all road segments
11 roadCenters = [0 0 0;
12               30 0 0];
13 marking = [laneMarking('Solid', 'Color', [0.98 0.86 0.36])
14           laneMarking('DoubleSolid', 'Color', [1 1 0.1])
15           laneMarking('Solid')
16           laneMarking('Solid', 'Color', [1 1 0.1])];
17 laneSpecification = lanespec(3, 'Marking', marking);
18 road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Road');
19
20 % Add the ego vehicle
21 egoVehicle = vehicle(scenario, ...
22                   'ClassID', 1, ...
23                   'Position', [3.3 -3.6 0], ...
24                   'Mesh', driving.scenario.carMesh, ...
25                   'Name', 'Car');
26
27 % Add the non-ego actors
28 vehicle(scenario, ...
29         'ClassID', 1, ...
30         'Position', [15.3 -3.8 0], ...
31         'Mesh', driving.scenario.carMesh, ...
32         'Name', 'Car1');

```

Source: Own authorship (2023)

There are the settings related to the road and the actors (cars, trucks, cyclists, pedestrians, or barriers). Sensors can also be added to the ego vehicle, the vehicle used for testing, allowing the positioning, angle of vision, range, and other attributes to be adjusted. In figure 11 it is possible to identify that there are comments delimiting the different implementations, such as the specification of the road segments and the added vehicles. There is an ego vehicle and other non-ego vehicles.

2.5.2 Vector's DYNA4

Vector's DYNA4 software is a simulation tool used to design and test dynamic systems, including automotive systems such as autonomous parking systems. It is based on the multi-body dynamics (MBD) approach, which allows for the simulation of complex interactions between multiple moving bodies and their environments. One key feature of DYNA4 is its ability to simulate the motion of bodies using both kinematic and kinetic analyses. This means that it can model both the position and velocity of bodies as well as the forces acting on them, allowing for a more accurate and detailed simulation of their behavior. An example of a simulation can be seen in figure 12.

Figure 12 - Example of a DYNA4 Simulation



Source: Own authorship (2023)

DYNA4 also includes a range of tools for model development and analysis, including tools for creating and editing models, visualizing simulation results, and performing parametric studies. It also includes interfaces for data exchange with other software tools and the ability to generate reports and documentation. In summary, Vector's DYNA4 software is a comprehensive and powerful tool for simulating the

behavior of dynamic systems, including autonomous parking systems. Its use of the MBD approach and wide range of analysis and model development tools make it a valuable resource for designers and engineers working in this field.

2.6 Motion Planning

While humans can easily move from one place to another, this is a major challenge for robots. Path planning is a central problem in autonomous robotics, where the goal is to find a safe and efficient path for a robot to navigate from a starting position to a desired goal position. This problem is relevant for a wide range of robotic applications, including vacuum cleaning robots, robotic arms, and even flying objects. The problem typically involves finding a path that avoids obstacles and satisfies various constraints on the robot's motion, such as its maximum speed or acceleration. The literature offers multiple approaches to address this problem, which depend on various factors such as the environment model, the type of robot, and the specific application (KOUBAA *et al.*, 2018).

Motion planning commonly involves generating a sequence of valid movements for a robot in a 2D or 3D world that may contain obstacles, where the robot may represent an actual robot or any other collection of moving bodies. The goal is to determine the appropriate motions for the robot to navigate to a desired goal state without colliding with obstacles. These algorithms typically take into account the geometry of the robot and its surroundings, as well as any constraints on the robot's movement, such as maximum speed or acceleration limits. The goal of motion planning is to generate a collision-free path for the robot to follow while minimizing the time and energy required to complete the task (LAVALLE, 2006).

To solve a navigation problem, a typical robot needs to know its location, where it should go, and how to get there. For this there are three stages, the location, mapping and path planning or motion planning.

Localization: sensors are usually used to identify the environment around the robot, such as ultrasonics, cameras, GPS, laser rangefinder. Location can be expressed locally (e.g., right side of a room), topologically (e.g., in Room 13), or absolutely (e.g., latitude, longitude, altitude) (LAVALLE, 2006).

Mapping: For a robot to effectively navigate in its environment, it needs to have a map that allows it to keep track of its location and the directions it has traveled. This map can be created manually and stored in the robot's memory in the form of a graph

or matrix representation, or it can be built incrementally as the robot explores new areas. By having an accurate and up-to-date map of its environment, a robot can more efficiently plan its movements and avoid collisions with obstacles (LLUVIA; LAZKANO; ANSUATEGI, 2021).

Motion planning or path planning: To enable a mobile robot to find a path from its current location to a target position, the robot must have prior knowledge of the target location. An effective way to provide this information to the robot is through an addressing scheme that the robot can interpret and follow. This addressing scheme specifies the destination relative to the robot's initial position, allowing it to understand where it needs to go. For example, in a parking lot, the robot can navigate to the destination only with the number of the space it must find. In other scenarios, the addressing scheme may use absolute or relative coordinates to indicate the target location. By utilizing an appropriate addressing scheme, the robot can navigate efficiently to the desired location while avoiding collisions with obstacles and minimizing the risk of getting lost. The addressing scheme is an essential element of the path planning process, helping the robot to accurately interpret its surroundings and successfully navigate to the target position (LAVALLE, 2006). Next the main solutions will be explained.

2.6.1 Classical Approaches

Classical path planning methods aim to either find a solution or prove that no solution exists. However, these methods have some drawbacks, including their high computational complexity and their inability to handle uncertainty, which makes them less suitable for real-world applications. This is due to the unpredictable and uncertain nature of many real-world scenarios. Classical path planning methods, such as Cell Decomposition (CD), Potential Field (PF) and Road Map fall into this category of methods (ATYABI, 2013).

Cell decomposition: is a classical method of path planning that involves dividing the free space of a robot's configuration into smaller regions called cells to generate a connectivity graph. This graph represents the adjacency between each cell, allowing for a continuous path to be determined by following adjacent free cells from the initial point to the goal point. The free space is decomposed into trapezoidal and triangular cells by drawing parallel line segments from each vertex of each interior polygon in the configuration space to the exterior boundary. Each cell is numbered and

represented as a node in the connectivity graph, with adjacent nodes linked in the graph. Finally, a free path is constructed by connecting the initial and goal configurations through the midpoints of the intersections of the adjacent cells. The cell decomposition method has several variants, including exact and approximate cell decomposition and the wave front planner (LAVALLE, 2006).

Potential Field: is a method that models a robot as a particle moving under the influence of a potential field. This field is determined by the obstacles and the target destination, where obstacles are assigned repulsive forces and the goal is assigned an attractive force. This approach allows the robot to move toward the target while avoiding obstacles. Different variants of PFM have been proposed to improve path planning, such as an improved PFM and genetic algorithm. In unknown environments, a new formula for repelling potential is used to reduce oscillations and conflicts when obstacles are near the target. In addition, a framework based on PFM is proposed to escape from a local minimum location of a robot path that may occur under narrow passages or similar scenarios (KOUBAA *et al.*, 2018).

Road map: also known as the Retraction, Skeleton or Highway approach, involves constructing a network of collision-free paths for motion planning. Path planning is then achieved by finding the shortest path between possible paths from the starting position to the goal position using the roadmap network. The two popular methods for developing road-maps are visibility and Voronoi graphs. Visibility graphs are graphs whose vertices consist of the start, target, and vertices of polygonal obstacles, with edges joining all pairs of vertices that can see each other. While the resulting path is usually the minimum-length solution, a disadvantage is that the shortest paths can touch obstacles at vertices or edges, making them unsafe. Voronoi diagrams address this issue by producing collision-free paths by dividing the free space around obstacles into regions (ATYABI, 2013).

2.6.2 Graph Search Approaches

Graph search approaches are a class of motion planning algorithms that represent the robot's environment as a graph of nodes and edges. The goal is to find a path from the initial position to the goal position on the graph while avoiding collisions with obstacles. Graph search approaches are typically categorized into two groups: search-based and sampling-based (LAVALLE, 2006).

2.6.2.1 A* algorithm

A* (pronounced "A star") is a widely used graph search algorithm in the field of artificial intelligence and robotics. It is an extension of the more basic Dijkstra's algorithm, with the addition of a heuristic function that guides the search towards the goal node and makes it more efficient. The A* algorithm is a search-based approach, meaning it explores the graph in a systematic way to find the shortest path. The search algorithm typically uses heuristics to guide the search and avoid exploring areas of the graph that are unlikely to lead to a solution. These algorithms are often used in situations where the environment is relatively simple and the computational resources are sufficient to explore the entire state space (GARCÍA, 2022).

To search for the shortest path each cell is evaluated according to equation 1. Where $g(n)$ is the accumulated cost of reaching the current cell n from the start position S , $h(n)$ is the estimated cost of the missing path to reach the goal, called heuristic. This can be defined as the Euclidean distance from the current cell to the target. For the algorithm to work faster, the Tie-breaking factor is used, which multiplies the value $h(n)$. This factor, when used, favors one direction over another in the event of a tie (LIN *et al.*, 2022). It's shown in the equation 2.

$$f(n) = h(n) + g(n)$$

$$g(n) = \begin{cases} g(S) = 0 \\ g(\text{parent}(n)) + \text{dist}(\text{parent}(n), n) \end{cases} \quad (1)$$

$$tBreak = 1 + \frac{1}{(\text{length}(\text{Grid}) + \text{width}(\text{Grid}))} \quad (2)$$

It maintains two lists: the open list, which contains nodes that have been visited but not yet fully explored, and the closed list, which contains nodes that have already been fully explored. The algorithm continues to expand nodes on the open list until the goal node is reached or the open list is empty. Each cell saved in the list has five attributes: ID, parentCell, g_Cost, h_Cost and f_Cost.

The algorithm starts by expanding the neighbor cells of the starting position. The neighbor cell with the lowest f_cost is chosen from the open list, added to the closed list, and then expanded in each iteration. However, the algorithm verifies some conditions before exploring the neighbor cells of the current cell. The conditions include ignoring cells that already exist in the closed list and comparing the g_cost of this path

to the neighbor cell and the g_cost of the old path to the neighbor cell if it already exists in the open list. If the g_cost of the new path is lower, the parent cell of the neighbor cell is changed to the current cell, and the g , h , and f costs of the neighbor cell are recalculated. This process is repeated until the algorithm reaches the goal position. Finally, the algorithm works backward from the goal cell, going from each cell to its parent cell until it reaches the starting cell to find the shortest path in the grid map (KOUBAA *et al.*, 2018).

A* is a popular and effective algorithm due to its optimality and completeness. It returns the optimal path if one exists, and it is guaranteed to find a path if one exists. However, the effectiveness of the algorithm can depend on the quality of the heuristic function used, which must be both admissible and consistent.

2.6.2.2 RRT*

RRT* (Rapidly-exploring Random Trees) is a sampling-based motion planning algorithm that can generate a probabilistically complete roadmap of the free configuration space of a robot. The algorithm incrementally grows a tree rooted at the starting configuration of the robot by randomly sampling new configurations and attempting to connect them to the tree. In RRT*, each node represents a configuration of the robot, and the edges of the tree represent feasible paths between these configurations. Each configuration is associated with a cost-to-come, which is the cost of the shortest path from the start configuration to that configuration (XINYU *et al.*, 2019).

The algorithm uses a cost function that assigns a cost to each edge in the tree, aiming to assess the quality of connections between the nodes. The cost of an edge is the Euclidean distance between the two configurations that it connects. The cost-to-come of a node is updated by finding the lowest-cost edge that can be used to reach the node. At each iteration, a new configuration is sampled randomly from the free space, and the nearest node in the tree is found. A new configuration is then added to the tree by connecting it to the nearest node. The algorithm then re-wires the tree by checking if any of the existing nodes can be reached from the new node by a path with a lower cost. If so, the tree is reconnected to reduce the overall cost of the tree (LAVALLE, 2006).

The algorithm continues to iteratively sample new configurations and add them to the tree until a path from the start configuration to the goal configuration is found.

RRT* is an asymptotically optimal algorithm, meaning that as the number of iterations approaches infinity, the algorithm is guaranteed to find the optimal path between the start and goal configurations, given certain assumptions about the cost function (LAVALLE, 2006). The figure 13 denotes the functioning of the algorithm.

Figure 13 - Algorithm RRT*

Algorithm 1 *RRT**

```

1:  $V \leftarrow start\_node; E \leftarrow \emptyset; T \leftarrow (V, E);$ 
2: for  $ind = 0 \rightarrow N$  do
3:    $new\_node \leftarrow sample(ind);$ 
4:    $near\_node \leftarrow near\_node(new\_node, T);$ 
5:   if  $near\_node = \emptyset$  then
6:      $near\_node \leftarrow nearest\_node(new\_node, T);$ 
7:   end if
8:    $L \leftarrow insert\_node(new\_node, near\_node);$ 
9:    $min\_node \leftarrow chooseparent(L);$ 
10:  if  $min\_node = \emptyset$  then
11:     $T \leftarrow insert\_vertex(new\_node, min\_node, T);$ 
12:     $T \leftarrow rewire(new\_node, L, E);$ 
13:  end if
14: end for
15: return  $T = (V, E);$ 

```

Source: Xinyu et al. (2019)

2.6.3 Cost Map

In path planning, a cost map is a grid-based representation of the environment where each cell is assigned a cost based on its characteristics such as traversability, distance to obstacles, or other factors that affect the robot's movement. The cost map is a useful tool for path planning algorithms as it provides a way to represent the environment and determine the optimal path through it (SUH; OH, 2012).

The cost map is typically generated by processing sensory data from sensors such as cameras, LiDAR, or sonar, and assigning a cost to each cell based on the data. For example, cells that are occupied by obstacles are assigned a high cost, while cells that are free of obstacles are assigned a low cost. The cost map can be updated in real-time as the robot moves through the environment and new sensory data becomes available (FERGUSON; LIKHACHEV, 2008).

Cost maps can be used in both search-based and sampling-based algorithms to determine the optimal path through the environment. In search-based algorithms,

the cost map is used to guide the search process and determine the optimal path, while in sampling-based algorithms, the cost map is used to bias the sampling process towards areas of low cost, which can improve the efficiency of the algorithm (FERGUSON; LIKHACHEV, 2008).

2.7 Vehicle Dynamics

The concept of vehicle dynamics corresponds to the description of the movement of vehicles, which can be automobiles, trucks, buses, and other special types of vehicles. It is interesting to identify and describe them to then obtain the movements of interest, such as acceleration and braking, ride, and turning (GILLESPIE, 2021). This concept can be considered in a rigid body, using the principles of Newton and Euler equations, describing force and moment (JAZAR, 2009).

2.7.1 Coordinate Systems

Firstly, it is necessary to describe the coordinates used by the system to describe the vehicle. There are two models that are commonly used: vehicle-fixed coordinate system and earth-fixed coordinate system. Both are important to describe in vehicle dynamics.

2.7.1.1 Vehicle-fixed coordinate systems

At vehicle-fixed coordinate systems exists two standards commonly used of the SAE and the ISO, in both the systems the coordinates originate at the center of gravity (C.G.) and travels with the vehicle (GILLESPIE, 2021). The SAE convention utilizes a right-hand orthogonal coordinate system, seen in the figure 14, and describe as follows:

x – Forward and on the longitudinal plane of symmetry

y – Lateral out the right side of the vehicle

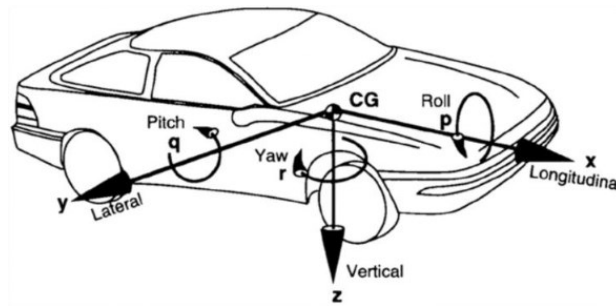
z – Downward with respect to the vehicle

p – Roll velocity about the x-axis

q – Pitch velocity about the y-axis

r – Yaw velocity about the z-axis

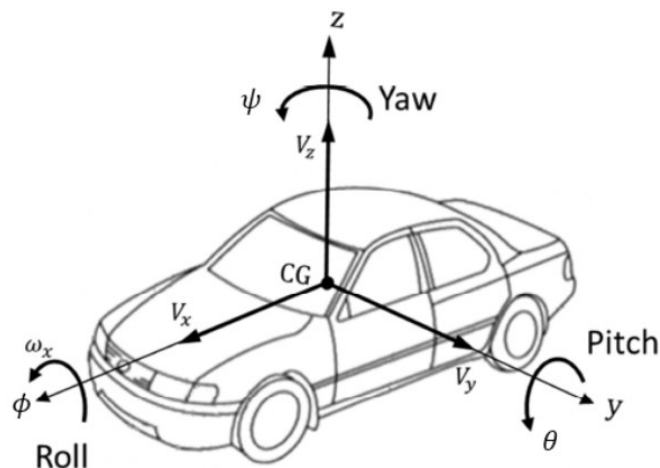
Figure 14 - SAE vehicle axis system



Source: Gillespie (2021)

Otherwise, the ISO standard defines the coordinate system differently, where the z-axis points up from the ground and the y-axis points to the left of the car, in the figure 15 the system is shown.

Figure 15 - ISO vehicle axis system



Source: Kissai et al. (2019)

2.7.1.2 Earth fixed coordinate system

It is important to define the earth fixed coordinate system, because this coordinate system is used to know the car position related to the world and it provides a reference frame for navigation and localization systems. Therefore, the applications like vehicle control and autonomous driving could be accomplish.

The standard is the use of a right-hand orthogonal axis system fixed on the earth. Commonly the axis is positioned at the start point of a maneuver and consequently coincide with the vehicle fixed coordinate system. The convention of the axis is shown in the figure 16, and the description is as follow:

X – Forward movement;

Y – Lateral movement (to the right is positive);

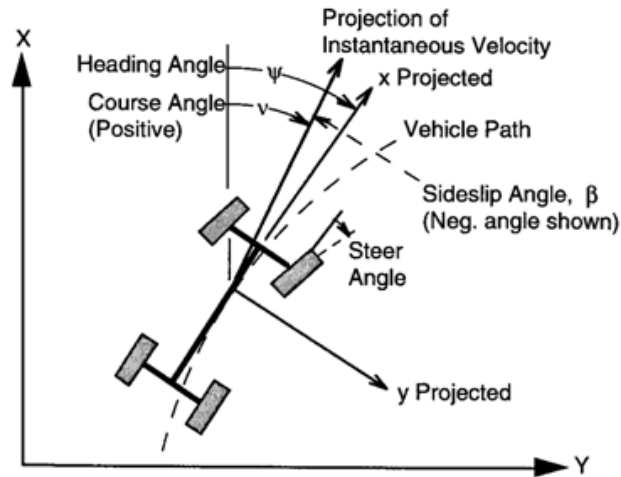
Z – Vertical movement;

ψ – Heading angle (angle between x and X in the ground plane);

ν – Course angle (angle between the vehicle's velocity vector and X axis);

β – Sideslip angle (angle between x axis and the vehicle velocity vector).

Figure 16 - Vehicle in an Earth Fixed Coordinate System

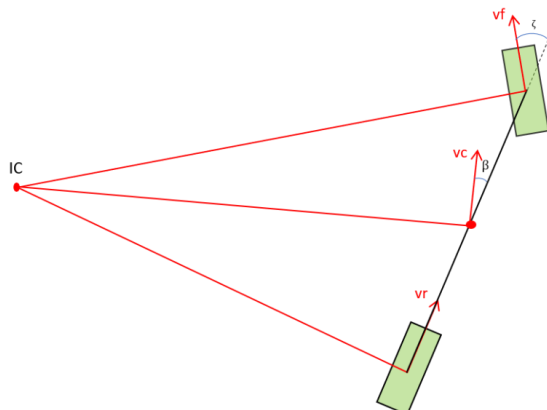


Source: Gillespie (2021)

2.7.2 Instantaneous center of rotation (ICR)

Given a rigid body, the instant velocity of any point of the body could be expressed as a result of a rotation about an axis perpendicular to the plane (JAZAR, 2009). Where the lines of the given points intercept are known as the instantaneous center of rotation or ICR, in the figure 17 could be seen the lines originate from the front and rear wheel instant velocity and from the center of mass instant velocity.

Figure 17 - Instantaneous center of rotation

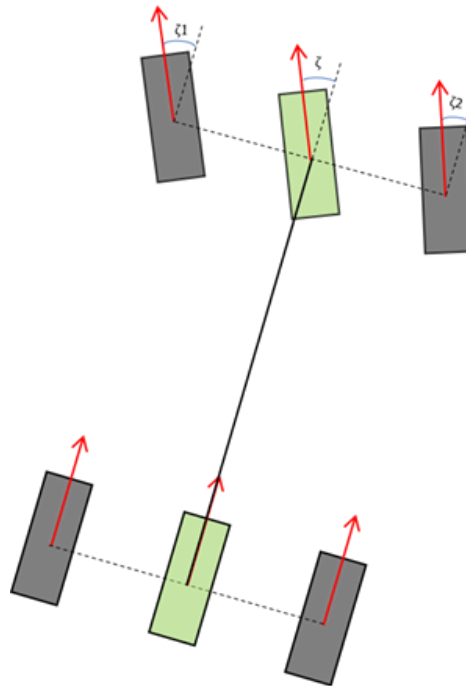


Source: Michael (2020)

2.7.3 Kinematic Bicycle Model

The kinematic bicycle model is a simplified mathematical representation of a bicycle that is used to analyze its dynamic behavior. The model could represent a car as a rigid body with two wheels and a steer. This is a classic model that does very well at capturing vehicle motion in normal driving conditions (DING, 2021). The model simplifies the vehicle dynamics combining the two front wheels and the two rear wheels into one front and one rear wheel respectively. This combination forms the two-wheeled model or most commonly named bicycle model. Therefore instead of two steering angle, the system only have one (MICHAEL, 2020). The figure 18 describe the model and show the steering angles.

Figure 18 - Two-wheel representation of a four-wheel system

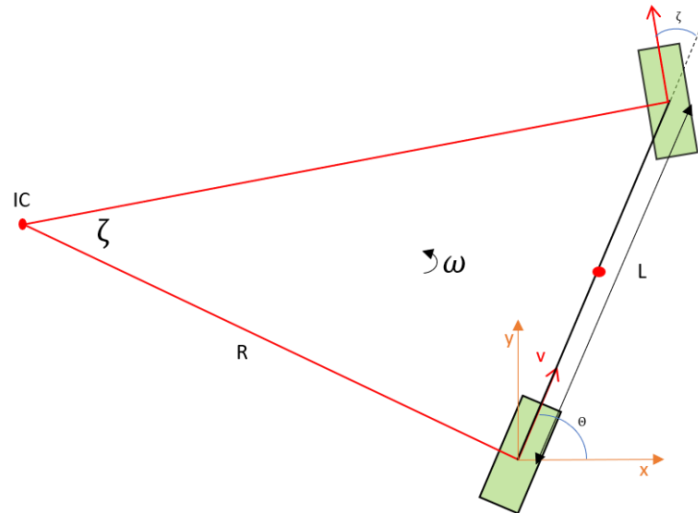


Source: Michael (2020)

Using the ICR model is possible to derive practical formulas for the kinematic bicycle model (THEERS; SINGH, 2023). The objective of the modeling is to find the equation to describe the behavior. Therefore, these parameters can feed a control system.

Utilizing the rear-axle as reference point, the direction of the vehicle's velocity is the same as the angle of the rear wheel, named θ , with respect to the x axis. The figure 19 denotes that.

Figure 19 - Kinematic bicycle model with the rear axle reference point



Source: Michael (2020)

It's necessary to find the velocity into components of x and y. Where \dot{x}_r are the v_x , the velocity in the x axis and \dot{y}_r are the v_y , the velocity in the y axis, the equation 3 denotes that, using trigonometrical relations:

$$\begin{aligned}\dot{x}_r &= v * \cos(\theta) \\ \dot{y}_r &= v * \sin(\theta)\end{aligned}\tag{3}$$

Next is important to define the angular velocity ω , this show how the model is changing his heading. Using the linear velocity (v) and the radius to the IC point (R), the angular velocity could be obtained (YOUNG; FREEDMAN; FORD, 2012). In equation 4 could be seen the relation.

$$\dot{\theta} = \omega = \frac{v}{R}\tag{4}$$

It's possible to obtain R using trigonometry, to do that is necessary find the angle x shown in the figure 21. It's known that the angle a and c is equal to 90° , therefore using the principle that: the internal sum of the angles of a triangle must be equal to 180° , it's obtained the equations 5 and 6. And subtracting equation 6 from 5 we get equation 7.

$$a + b + \zeta = 180^\circ \quad (5)$$

$$x + b + c = 180^\circ \quad (6)$$

$$x = \zeta \quad (7)$$

Knowing that the angle x has the same value as the angle ζ , referring to the steering of the model, it is possible to deduce equation 8. Then it is only necessary to isolate R .

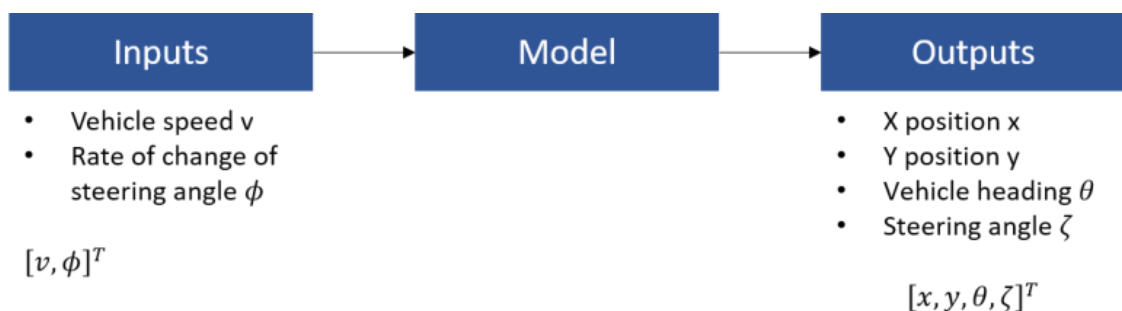
$$\tan(\zeta) = \frac{L}{R} \quad (8)$$

$$R = \frac{L}{\tan(\zeta)} \quad (9)$$

It is then possible to obtain the final equations shown in 10. And the figure 20 summarizes the model, showing the inputs and outputs.

$$\begin{aligned} \dot{x}_r &= v * \cos(\theta) \\ \dot{y}_r &= v * \sin(\theta) \\ \dot{\theta} = \omega &= v * \frac{\tan(\zeta)}{L} \end{aligned} \quad (10)$$

Figure 20 - Inputs and outputs of the kinematic bicycle model



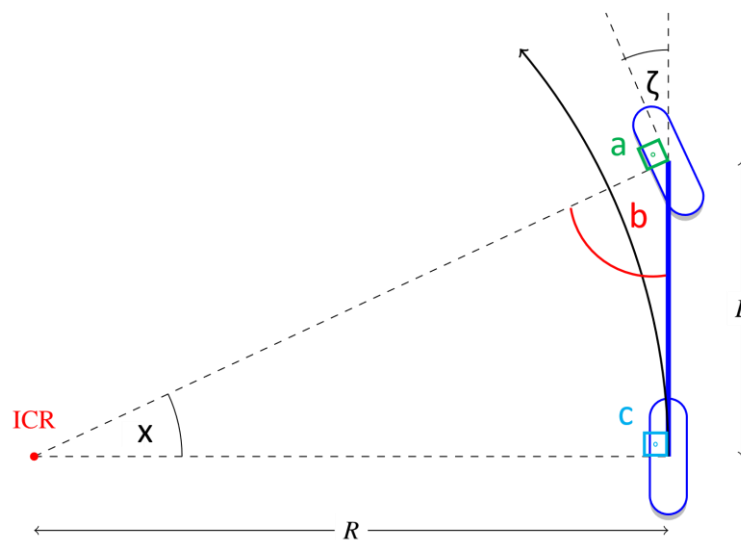
Source: Michael (2020)

To bring this model closer to its use in practice, the discretized model is shown with the equations in 11. The use of the rate of change of steering angle instead of the steering angle can be noticed, this is used considering that the vehicle cannot turn the steering wheel instantly.

$$\begin{aligned}
 x(t+1) &= x(t) + \dot{x} * \Delta t \\
 y(t+1) &= y(t) + \dot{y} * \Delta t \\
 \theta(t+1) &= \theta(t) + \dot{\theta} * \Delta t \\
 \zeta(t+1) &= \zeta(t) + \dot{\zeta} * \Delta t
 \end{aligned}
 \tag{11}$$

In this way, the x and y positions, vehicle heading θ , and steering angle ζ are obtained, as can be observed in figure 21, with inputs being the vehicle speed v and the rate of change of the steering angle Φ .

Figure 21 - Steering angle in bicycle model



Source: Adapted from Theers and Singh (2023)

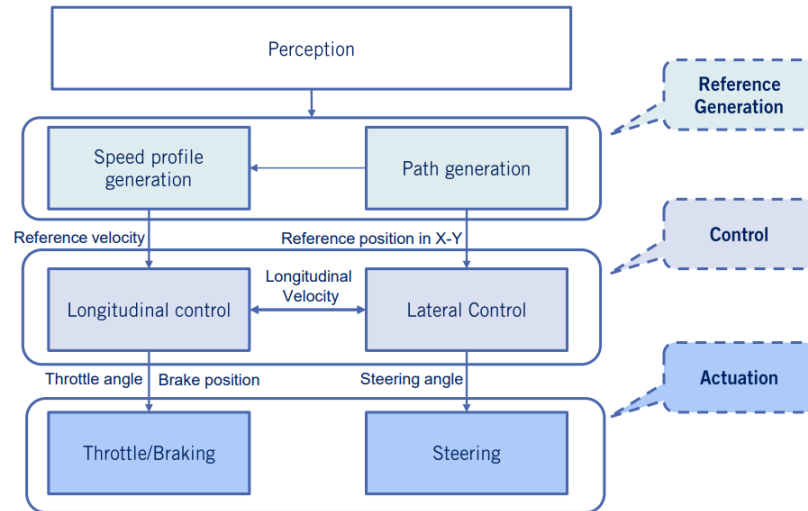
2.8 Control Strategies

Automotive control involves the use of control theory and engineering to improve the performance and safety of automobiles. Control systems are used to regulate and optimize various aspects of vehicle behavior, including acceleration, braking, steering, and stability. The goal of automotive control is to develop control systems that can improve vehicle performance, enhance driver safety, and reduce fuel consumption and emissions (KIENCKE; NIELSEN, 2005).

Recent advances in automotive control have been driven by the increasing use of electronic and computer-based systems in modern vehicles. These systems allow for more precise and efficient control of vehicle behavior, and have enabled the development of advanced driver assistance systems (ADAS) and autonomous driving

technologies (CHOI *et al.*, 2016). The figure 22 shows an example of an automotive control architecture.

Figure 22 - An example of a control architecture



Source: Waslander (2018c)

2.8.1 Transfer Function

The transfer function is a fundamental concept in control theory that describes the relationship between the input and output of a control system in the frequency domain. The transfer function of a control system is a mathematical representation of how the system responds to different inputs and disturbances (OGATA, 2009-).

The transfer function is usually expressed in terms of Laplace transforms, which are a mathematical tool used to analyze linear time-invariant systems. The transfer function is defined as the ratio of the Laplace transform of the output to the Laplace transform of the input, assuming zero initial conditions. In other words, it gives the relationship between the input and output signals in the Laplace domain (NISE, 2017).

The transfer function is typically denoted by $G(s)$, where “s” is the Laplace variable. The transfer function can be derived from the differential equations that describe the system dynamics, using techniques such as Laplace transforms and partial fraction expansion (OGATA, 2009-). A transfer function represented in the Laplace domain is shown in the equation 12, where G is the ratio between input U and output Y . The “s” variable is defined in equation 13 according to the laplace transformation.

$$Y(s) = G(s)U(s) \quad (12)$$

$$s = \sigma + j\omega \quad (13)$$

The Laplace transform is a great tool for analyzing the input-output relationship and for a better understanding of the performance of the control system. As well as the roots of the numerator and denominator which provide insight into a system's response to input functions, zeros being the roots of the numerator and the poles roots of the denominator (WASLANDER, 2018c).

2.8.2 Logitudinal Control

When it comes to controlling the vehicle's longitudinal motion, which includes its longitudinal velocity, acceleration, or distance from another preceding vehicle in the same lane on the highway, the phrase "longitudinal controller" is commonly used. The actuators responsible for carrying out longitudinal control are the throttle and brakes (RAJAMANI, 2011).

According to vehicle modeling, the acceleration a is responsible for the change in speed v . The relationship between v and a is typically represented as $\dot{v} = a$, which is modeled using an integrator in control theory. This integrator can be controlled with a PID controller. However, since the acceleration of the vehicle is often constrained due to mechanical limitations and passenger comfort, it is important to design the longitudinal controller with an input saturation (OLSSON, 2015).

A commonly known example of longitudinal control is demonstrated by the cruise control system that is present in most vehicles. The driver sets a constant desired speed for the vehicle, and the cruise control system automatically adjusts the throttle to maintain that speed. Nevertheless, the driver must ensure that the vehicle can safely travel at that speed on the highway. If a slower preceding vehicle appears on the highway or if the ego vehicle gets too close, the driver must apply the brakes, if necessary, which disengages the cruise control system and returns control of the throttle to the driver (RAJAMANI, 2011).

2.8.2.1 Proportional–integral–derivative (PID) control

PID control is a type of feedback control that is commonly used in engineering and industrial applications to regulate a process. PID stands for Proportional-Integral-

Derivative, which are the three types of control actions that the controller can use to adjust the process output. The proportional action is proportional to the current error, the integral action is proportional to the cumulative error over time, and the derivative action is proportional to the rate of change of the error. By combining these three actions, the PID controller can quickly respond to changes in the process input and maintain a stable output. PID control is used in a wide range of applications, including temperature control, motion control, and industrial process control. It is a widely used and well-understood control technique, and it has been the subject of extensive research and development over the years (NISE, 2017). Mathematically the PID control can be represented in the time domain, as shown in equation 14, or more commonly it can be represented in the s domain shown in equation 15.

$$u(t) = K_P e(t) + K_I \int_0^t e(t) dt + K_D \dot{e}(t) \quad (14)$$

$$U(s) = G_C(s)E(s) = \left(K_P + \frac{K_I}{s} + K_D s \right) E(s) = \left(\frac{K_D s^2 + K_P s + K_I}{s} \right) E(s) \quad (15)$$

Where $u(t)$ is the control input at time t , $e(t)$ is the error between the setpoint and the process output at time t , and K_P , K_I , and K_D are the proportional, integral, and derivative gains, respectively. The function in the laplace domain have the same principles. It is possible to select the positioning of the zeros by selecting the K_P , K_I and K_D gains, in the case of the equation 16 the pole is at the origin.

$$G_C(s) = \frac{K_D s^2 + K_P s + K_I}{s} \quad (16)$$

The proportional term K_P is a constant gain that determines the strength of the correction applied to the control input based on the current error $E(s)$. A larger value of K_P leads to a more aggressive correction, while a smaller value leads to a more gradual correction. The integral term K_I is a constant gain that determines the strength of the correction applied based on the accumulated error over time. The integral term sums up the error over a specified time interval (usually from the start of the control process to the present time), and multiplies it by the constant gain K_I . This helps to correct for any steady-state errors that may be present. The derivative term K_D is a

constant gain that determines the strength of the correction based on the rate of change of the error. The derivative term takes the rate of change of the error over time into account, and multiplies it by the constant gain K_D . This helps to prevent overshooting and oscillations in the control process (OGATA, 2009-). The frame 2 summarizes the concept.

Frame 2 - Changes in gains of the PID controller and their potential effects by properly tuning the PID gains

Closed Loop Response	Rise Time	Overshoot	Settling Time	Steady State Error
Increase K_P	Decrease	Increase	Small change	Decrease
Increase K_I	Decrease	Increase	Increase	Eliminate
Increase K_D	Small change	Decrease	Decrease	Small change

Source: Waslander (2018a)

The transfer function of the plant being controlled is typically represented by $G(s)$. The closed-loop transfer function of the system with PID control is given by the equation 17.

$$\frac{C(s)}{R(s)} = \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} \quad (17)$$

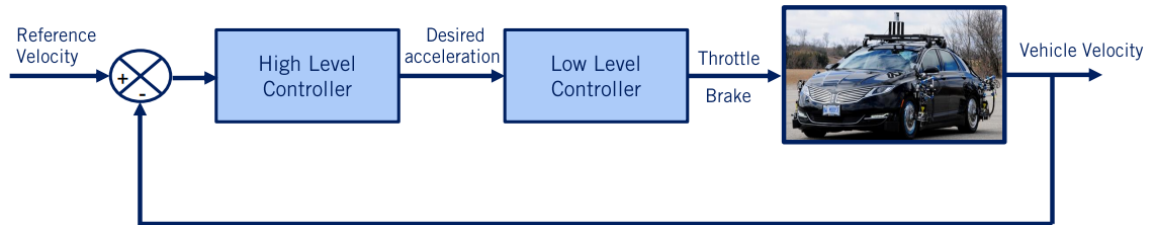
Where $C(s)$ is the output of the controller, $R(s)$ is the reference signal or setpoint, $G(s)$ is the transfer function of the plant and $G_c(s)$ is the PID transfer function. The closed-loop transfer function can be analyzed to determine the stability, steady-state error, and response characteristics of the system. The gains K_P , K_I and K_D must be tuned to optimize the control performance for a given system. This can be done through manual trial-and-error, or using more advanced methods such as Ziegler-Nichols or Cohen-Coon tuning methods.

2.8.2.1.1 Cruise control

One of the main examples of using a PID controller is in the making of a cruise control, in which the speed of the car is controlled through the accelerator and brake to stay within the established reference speed. The controller could be split in two level, a high-level controller and a low-level controller. The high-level controller receives the reference velocity and the actual velocity of the car, and computes a desired

acceleration to the low-level controller, that outputs a throttle and brake commands. In the figure 23 is possible to see an example of controllers to make this system.

Figure 23 - Cruise control schematic



Source: Waslander (2018c)

The desired acceleration can be found using the PID controller shown in figure 24.

Figure 24 - Equation to find the desired acceleration of a cruise control

$$\ddot{x}_{des} = K_P(\dot{x}_{ref} - \dot{x}) + K_I \int_0^t (\dot{x}_{ref} - \dot{x}) dt + K_D \frac{d(\dot{x}_{ref} - \dot{x})}{dt}$$

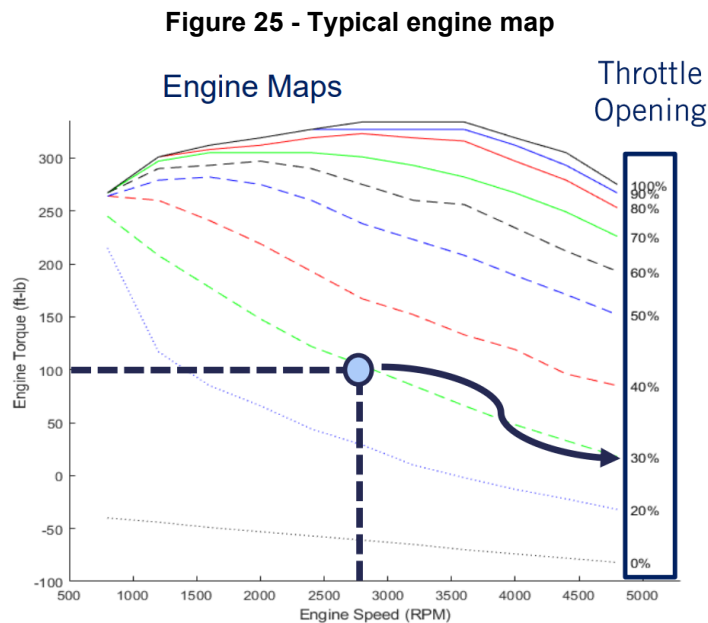
PID Gains

Source: Waslander (2018c)

In the dynamics of mechanical systems, equation 18 describes the relationship between the engine torque (T_{Engine}), the system's moment of inertia (J_e), the distance from the rotation axis (r_{eff}), angular acceleration (GR), and linear acceleration (\ddot{x}_{des}). Using the kinematic bicycle model and the engine torque equations with the engine maps is possible to design a control to output the throttle and brake angle. In cruise control normally the output is only the throttle angle. With this assumption, considering the torque converter locked (gear 3+) and with the tire slip small (gentle longitudinal maneuvers) is possible to find the torque engine using the equation 18 with the desired acceleration, knowing that the other variables are constants in this system.

$$T_{Engine} = \frac{J_e}{(r_{eff})(GR)} \ddot{x}_{des} + T_{Load} \quad (18)$$

With the engine map in figure 25 the throttle could be find using the torque engine.



Source: Waslander (2018c)

2.8.3 Lateral Control

Vehicle lateral control refers to the process of maintaining a vehicle's lateral position or trajectory relative to a specific reference point while in motion. This involves controlling the vehicle's lateral movement, which is primarily achieved through the steering system (SOUDBAKHSI; ESKANDARIAN, 2012).

Lateral vehicle control, can be mathematically described as the process of regulating the lateral position of a vehicle with respect to a predefined reference trajectory or point, typically in the context of time.

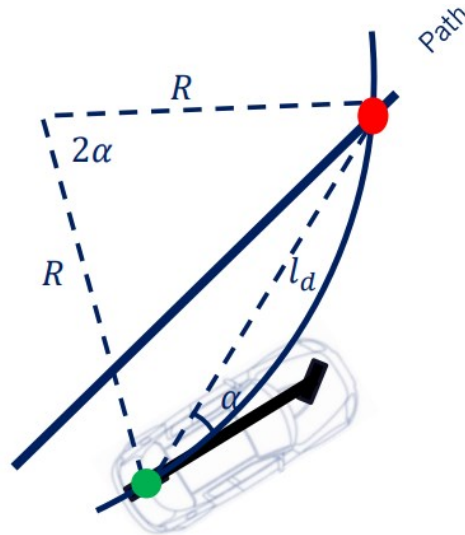
The goal of lateral control is to minimize this lateral error, by making steering adjustments to the vehicle's direction, which is achieved primarily through the steering angle applied to the front wheels. A common mathematical model used in lateral control is the bicycle model, where the lateral motion of the vehicle is typically represented using a combination of differential equations and control algorithms.

2.8.3.1 Pure Pursuit

One of the most commonly used techniques for path tracking in mobile robots is the pure pursuit method and its variations. This method involves calculating the curvature of a circular arc that connects the rear axle position to a goal point on the

path ahead of the robot. The goal point is determined by a look-ahead distance l_d from the current rear axle position to the desired path represented by the coordinates (g_x, g_y) , as shown in figure 26. By utilizing the location of the goal point and the angle α between the vehicle's heading and the look-ahead vector, the steering angle ζ of the robot can be calculated (SNIDER, 2009).

Figure 26 - Path following with pure pursuit controller



Source: Waslander (2018b)

In a Pure Pursuit control system, the variables play a fundamental role in guiding a vehicle toward a target point along a planned trajectory. The first variable, l_d , represents the distance from the vehicle to the target point along the reference path, determining when the vehicle should make steering adjustments. The angle α denotes the orientation angle that the vehicle needs to follow to align itself with the target point. Lastly, R signifies the radius of curvature of the reference path at the vehicle's current position, influencing the curvature of the path the vehicle must follow to reach the target point. The law of sines can be applied to figure 26 to derive the relationship in the equations 19.

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)}$$

$$\frac{l_d}{2 \sin(\alpha) \cos(\alpha)} = \frac{R}{\cos(\alpha)} \quad (19)$$

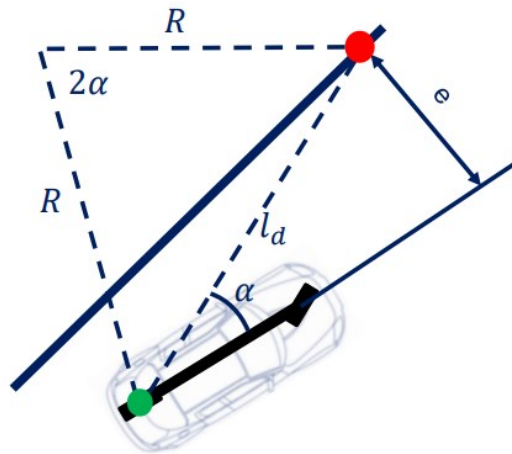
$$\frac{l_d}{\sin(\alpha)} = 2R$$

In other terms the answer can be expressed as path curvature κ , using equation 9 of the bicycle model, the relationship shown in equation 20 can be obtained to find the steering angle ζ , defining the pure pursuit control law.

$$\begin{aligned}\kappa &= \frac{1}{R} = \frac{2 \sin(\alpha)}{l_d} \\ \zeta &= \tan^{-1}(\kappa L) \\ \zeta &= \tan^{-1}\left(\frac{2L \sin(\alpha)}{l_d}\right)\end{aligned}\quad (20)$$

The error of this controller is called the cross-track error e and is defined as the lateral distance between the heading vector and the target point, as could be seen in figure 27.

Figure 27 - Pure pursuit controller with cross track error e



Source: Waslander (2018b)

In the equation 21 the cross-track error is calculated.

$$\begin{aligned}\sin(\alpha) &= \frac{e}{l_d} \\ \kappa &= \frac{2 \sin(\alpha)}{l_d} = \frac{2}{l_d^2} e\end{aligned}\quad (21)$$

This controller is proportional and can vary according to vehicle speed, for this reason the variable l_d can be associated with vehicle speed as shown in equation 22.

$$\begin{aligned} l_d &= K_{dd}v_f \\ \zeta &= \tan^{-1}\left(\frac{2L \sin(\alpha)}{K_{dd}v_f}\right) \end{aligned} \quad (22)$$

2.9 Chapter's considerations

In this chapter, fundamental concepts for understanding parking systems and ADAS systems were introduced. Within the fundamentals, it was found that parking systems are typically developed using ultrasonic sensors because they provide essential advantages for this type of system, such as reduced cost, short-range detection, commercial availability, and fast data processing. Regarding motion planning methods, RRT* was found to be the most interesting, as it provides a quick response, generates faster and more reliable routes than other methods, such as A*. It was also demonstrated that the MATLAB software is an appropriate platform for developing an all-in-one solution incorporating the main requirements for parking systems. Additionally, control systems used in the automotive industry, which are good pathways for the initial implementation of the system, were discussed.

3 METHODOLOGY

The methodology section of this report outlines the process used to design and simulate autonomous parking systems. It covers the use of a driving scenario designer, the identification of parking spaces, the maneuvering of the vehicle into and out of parking spaces, and the simulation of both perpendicular and oblique parking.

3.1 Bibliography Analysis and Literature Review

Seeking to acquire the necessary bibliographic portfolio for research development, the PRISMA methodology was used. The PRISMA methodology (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) is a guide for the development of systematic reviews and meta-analyses, which is a method used to synthesize and summarize the available evidence on a specific research question. Its objective is to help authors conduct a systematic review and meta-analysis transparently and standardized, providing a checklist for reporting these studies. The methodology consists of a set of guidelines for the stages of study selection, data extraction, study quality evaluation, and data synthesis, which are:

- **Identification:** Search of selected databases according to the research axes and/or search terms defined in the research protocol;
- **Selection:** Filtering of articles following the reading criteria of title, abstract, and keywords proposed by the methodology, seeking to verify the alignment of the article proposal with the researched topic;
- **Eligibility:** Complete reading of pre-selected articles to assign the final research portfolio. If incompatibility with the topic is verified, the article is excluded;
- **Inclusion:** Phase of including other studies by qualitative analysis to compose the final research portfolio.

First, it was necessary to identify the keywords on the topic and define the search criteria in the selected databases. The search terms were defined according to the different synonyms of the concept of APS, using the Boolean operator "OR". The parameters used are listed in frame 3.

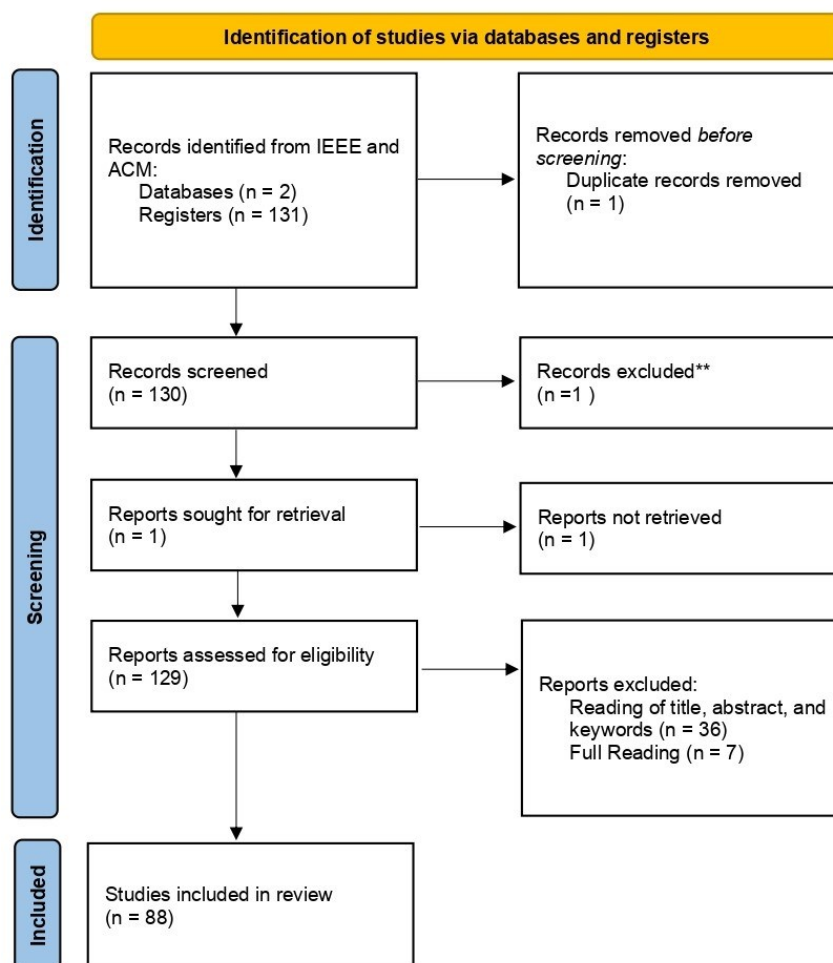
Frame 3 - Research Protocol

Topic	Autonomous Parking
Method	Prisma
Type	All
Language	English
Axis	("Motion Planning" OR "path tracking" OR "path-planning") AND ("Automatic parking system" OR "parallel parking" OR "Parking scenarios" OR "perpendicular parking")

Source: Own autorship (2023)

The PRISMA includes a flowchart that clearly describes the study selection process, and a checklist that helps ensure that all critical elements of the systematic review and meta-analysis are reported adequately and transparently. The methodology development flowchart can be viewed in figure 28.

Figure 28 - PRISMA flow diagram



Source: adapted from the PRISMA Statement (2023)

A final portfolio of 88 studies was obtained, which were submitted to bibliometric analysis. Using VOSVIEWER and Microsoft Excel software, it was possible to understand the network of relationships present in the categories:

- Keywords;
- Publications per year;
- Authors.

VOSviewer is a tool for visualizing and analyzing bibliometric networks and maps, which is used to analyze and visualize bibliographic data such as scientific articles, journals, authors, and other academic publications. It allows the analysis of large volumes of data, such as bibliographic references collected in a systematic review or meta-analysis, and the identification of patterns and trends in the data. VOSviewer was developed at Leiden University in the Netherlands and is a free and open-source tool that can be used to create visualizations of networks, maps, and bibliographic co-citation, co-authorship, and co-citation graphs. It is very useful for researchers who wish to analyze scientific production in a particular area of research or in several related areas.

3.2 Simulation

In this section, we will present the methodologies used to develop the simulation of the proposed APS (Automated Parking System). Simulation plays a crucial role in the APS development process, allowing for a thorough analysis of its capabilities, efficiency, and safety in a controlled environment. The objective of this simulation is to evaluate the performance of the APS in different parking scenarios, taking into consideration factors such as spot detection, route planning, and parking maneuvers. Through simulation, valuable insights are gained about the system's behavior, identifying potential areas for improvement.

By utilizing a virtual environment, a wide variety of scenarios can be created, ranging from simple parking lots to tight spaces with additional challenges. With simulation, the APS's success rate in autonomously finding and parking in spots can be evaluated, as well as the time required to complete the entire process. It is also possible to analyze the system's efficiency in utilizing available space, aiming to optimize spot occupancy by minimizing wasted space and maximizing parking capacity.

3.2.1 Software selection for simulation

Initially, a literature review was conducted to situate oneself on the theme and begin planning the simulation with a focus on finding autonomous parking methods and programs to realize the simulation. During the research, it was identified that in a large part of the work, simulation is carried out on unspecified software. There were also works using Mathworks software such as Simulink and Matlab with the add-on driving scenario designer. There was also the possibility of using the DYNA4 software from Vector for the simulation, which has great advantages in relation to creating high-fidelity simulations with reality. However, due to the complexity of the software and the lack of tutorials and guides for its use, it was decided that Matlab would be the best solution, as it provides various libraries of content that can be used to take the first steps in new topics, which can otherwise become complex.

3.2.2 Scenario creation

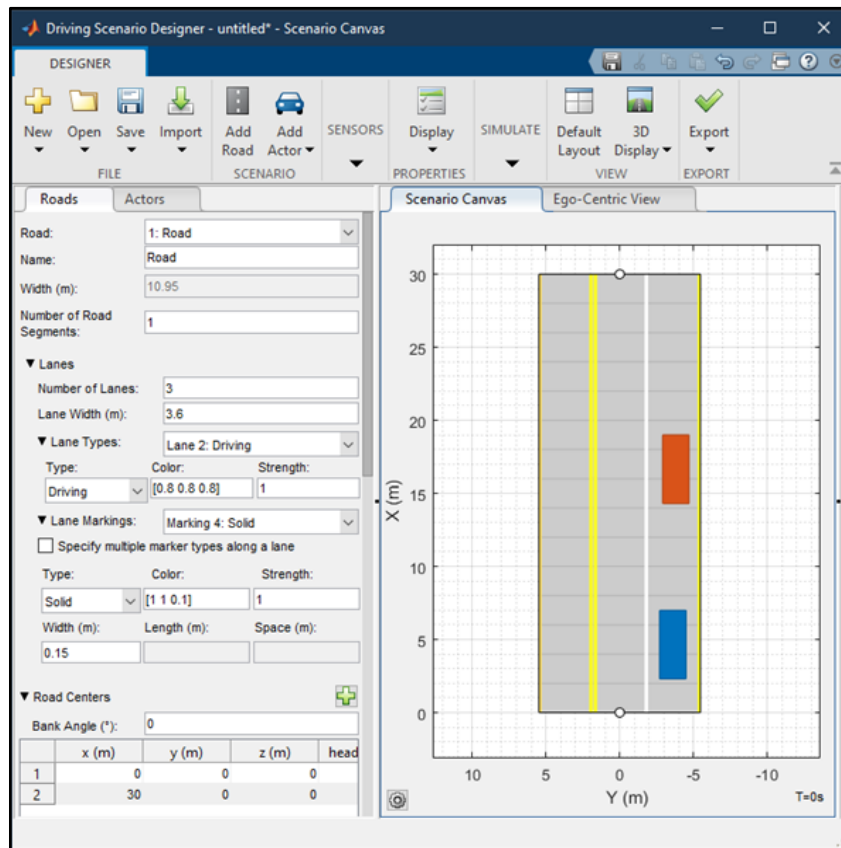
In order to enhance the efficiency of simulation development, a deliberate decision was made to employ the driving scenario add-on within the Matlab software. This strategic choice was motivated by the desire to optimize resources and time, thereby enabling a more focused approach to refining the core aspects of the research study.

Testing the driving scenario simulation began to better understand its operation. Mathworks' first steps documentation, explanatory videos, and examples from their website were used for reference. This validation process provided valuable insights into the add-on's capabilities and facilitated accurate simulation outcomes. Then, the scenario for parallel parking simulation was created, using the following logic:

- 1) Develop scenario in the driving scenario designer add-on;
- 2) Export as "MATLAB Function";
- 3) Open the generated file with MATLAB.

In figure 29, it is possible to identify the creation of a scenario with two parked cars on a street.

Figure 29 - Creation of a driving scenario with two cars parked



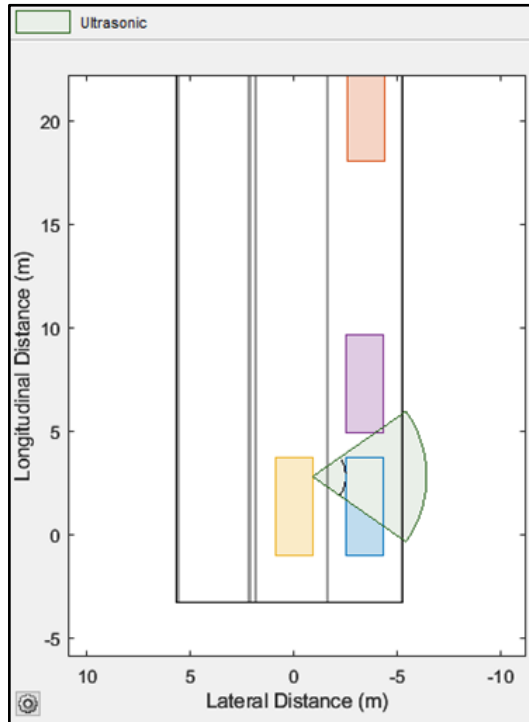
Source: Own authorship (2023)

3.2.3 Identification of the parking space

The first step to achieving the goal of autonomous parking is the identification of the parking space. In this projects, ultrasonic and vision sensors are used. Ultrasonic sensors have desirable characteristics such as low acquisition cost, ease of use, and operation independent of lighting conditions, factors that make them ideal for use in parking assist systems and automated parking systems. Vision sensors have good performance when used with ultrasonic sensors in the parking space identification phase and obstacle identification.

As the use of machine learning and sensor fusion makes it possible to obtain accurate data on the size of the measured spaces and detected obstacles. Initially, a scenario was created as demonstrated in section 3.2.2 with three cars and one parking space between the second and third, with the ego vehicle placed in the center of the central lane with an ultrasonic sensor in the front front of it, in the perpendicular direction, with the aim of measuring whether the space between the cars is sufficient for parking. The scenario can be seen in figure 30.

Figure 30 - Base scenario for parking space detection



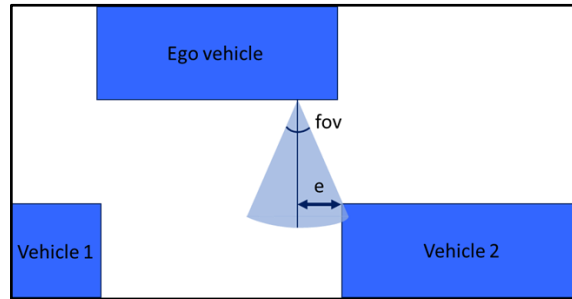
Source: Own authorship (2023)

To identify the parking space, the following logic is used: when there is no more signal, a possible parking space is identified for parking, and the size of the space is then counted by incrementing the variable "lengthVaga" (L_v), adding itself to the result of dividing the current speed of the ego vehicle by the sensor's update rate. Equation 23 represents this operation.

$$L_{v2} = L_{v1} + \frac{V_e}{F_s} \quad (23)$$

It is important to mention a problem inherent to ultrasonic sensors in the identification and measurement of a possible parking space. The operation of this type of sensor only allows the identification that there is an obstruction of it at a certain distance, being possible to calculate it using time of flight concepts. Figure 31 demonstrates this problem, the detected object can be in any position parallel to the car within the light blue band. The size of this area depends on the FOV of the ultrasonic sensor. Thus, in the measurement of the parking space, an error occurs, marked in the figure with the letter e , the size of the parking space is calculated smaller than the real size.

Figure 31 - Error in identifying the real size of the parking space



Source: Own authorship (2023)

Some solutions to this problem are:

- 1) Use of an ultrasonic sensor with a small field of view;
- 2) Development of software to mitigate the error calculated through the use of secondary and tertiary echoes received;
- 3) Use of more ultrasonic sensors and with the help of an algorithm to more accurately identify the transition points;
- 4) Use of sensor fusion with other sensors, such as cameras, radar, lidar, among others.

3.2.4 Maneuver of entering and exiting the parking space

With the correct identification of parking spaces, the next step was to conduct a literature review in order to find methods for the car to maneuver into and then out of a parking space, performing the parking and exit from the space with a continuous movement and in a fast, efficient and safe way. Some of the main methods were identified in the literature, one of which was mathematical, as in the work of Paromtchik and Laugier (1996), which presents geometric relationships for parking in a specified parking space. Another method is the use of path planning algorithms such as A*, RRT*, Bi-RRT*, among others, which is the most chosen approach in recent works because, despite its higher computational cost, it has an advantage in the generalization of the parking place. The use of Bi-RRT* can be observed in the work of Jhang and Lian (2020), where good results were obtained in a real test environment, as the algorithm finding free paths with human-like movements.

It is interesting to mention the series of videos titled "matlab tech talks" within the category "autonomous navigation," which provides a good theoretical basis for entering the world of autonomous vehicles and provide an important foundation for understanding and implementing path planning algorithms. Due to reliability,

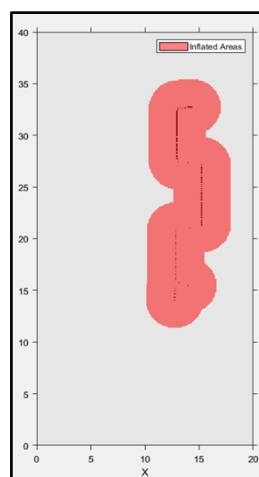
processing time, and ease of implementation, the model chosen to perform the simulation was the RRT* algorithm, which has an implementation for vehicles within the "automated driving toolbox" of MATLAB. The function "pathPlannerRRT" is used for this purpose. It was decided that for parallel and perpendicular parking, the car should enter with the rear, as this allows the parking space to be identified with the sensors as it passes through it. In oblique parking, however, this is not possible, as the only way to park is frontally.

3.2.5 Developing a method to create a cost map from the driving scenario

For the path planning algorithm to work, a cost map is needed, so that the algorithm can know the occupied spaces on the map, places where the calculated trajectory cannot pass through. As the parking space detection had already been carried out by the code developed in the work, there was work to identify the necessary data, already collected by the sensor, to generate the cost map. It was identified that it would be necessary to have values in the form of two-dimensional coordinates (x, y), in addition to the transposition of the map from the driving scenario add-on to the automated driving toolbox.

Initially, a blank cost map was created, and then a logic was developed that transposed the objects detected by the sensor, using the ego vehicle's location and the sensor's position as a base. In this way, positions were obtained in relation to the cost map, and with them it was possible to create the positions where objects detected by the sensor were located, as can be seen in figure 32, created by transforming points from the ego vehicle's coordinates to the coordinates of the automated driving toolbox.

Figure 32 - Generated cost map



Source: Own authorship (2023)

3.2.6 Path planning

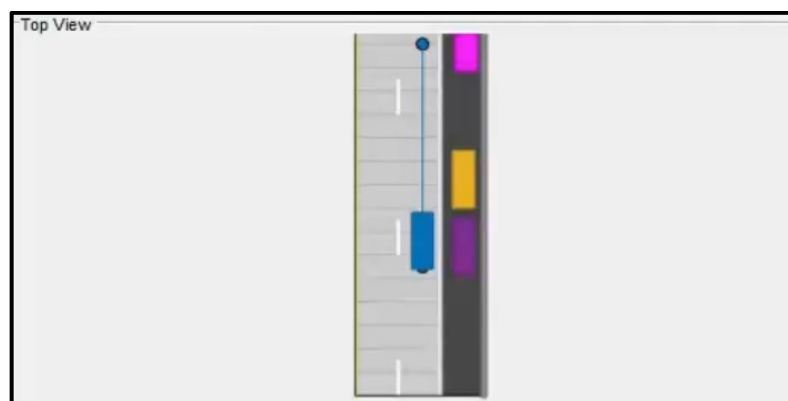
After creating the cost map, it was tested in a scenario with several parked cars on the right lane, in an environment for the parallel parking test. For simplicity, the car's initial trajectory was defined to stop right after the parking space, and the path planning objective was calculated when the parking space had the minimum viable size for parking. From the sensor data and the car's position, the coordinates for parking the car in the center of the parking space are obtained. Then the path planning for parking is performed, the cost map, the vehicle's initial position, and the parking position are passed. After a few seconds, the trajectory that the car should follow is calculated and stored in a variable called waypoints that must be passed.

These waypoints contain coordinates (x, y, z) and the corresponding angle of the car. Then the data from this variable is transformed into the ego vehicle's coordinate base and then passed back to the simulation through the "trajectory" command. The inverse trajectory is also passed so that the car can exit the parking space in sequence.

3.2.7 Parallel parking

With the development of the cost map and the adaptation of the path planning, a parallel parking scenario was created to test the system. A street and some parked cars were created using the driving scenario. With this scenario, code was generated automatically, allowing for modifications in specific parameters of the scenario. After that, a path was programmed for the ego vehicle to follow, as shown in figure 33. During the course, the ego vehicle collected data with its sensors, which were processed in other classes to perform the parking maneuver.

Figure 33 - Top view from the road of the parallel parking simulation

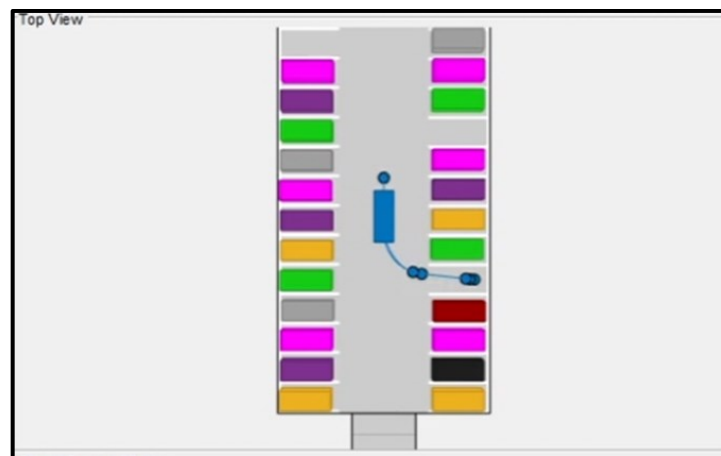


Source: Own authorship (2023)

3.2.8 Perpendicular parking

For the simulation in a perpendicular parking lot, it was necessary to create a new map using the parkingLot function to quickly and automatically create the parking spaces, then several cars were added to populate the parking lot. Parameters were modified such as the positioning of the car and its initial trajectory. The size of the sought parking space was also modified, instead of using the length of the car to compare with the parking space, the width was used, as this is the limiting factor in this type of parking. In figure 34, the result can be identified.

Figure 34 - Top view with the waypoints generated for perpendicular parking



Source: Own authorship (2023)

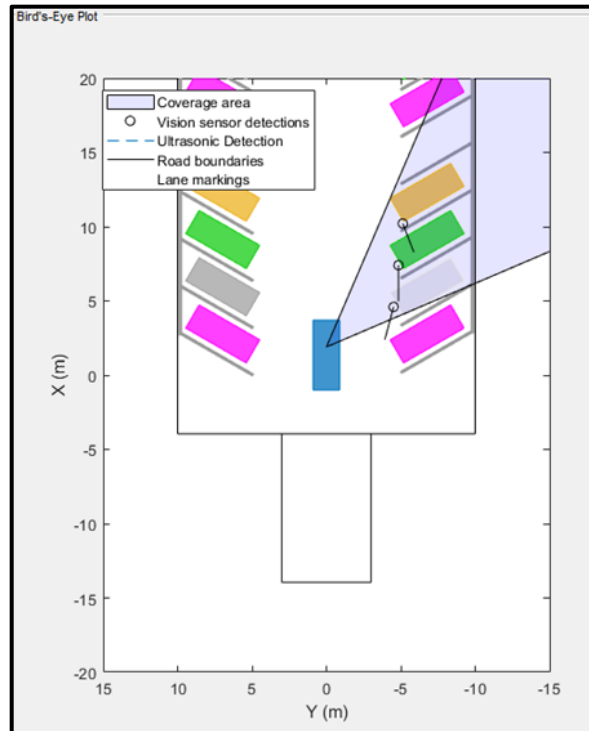
3.2.9 Angled parking

At the beginning, a map was created with 60° angled parking spaces using the parkingLot function. The map was then populated with cars and the ego vehicle was positioned. To perform the simulation with the goal of performing angled parking, it is necessary to rethink the logic of identifying and parking spaces, as it is not feasible to identify them using ultrasonic sensors. This type of parking space needs to be identified beforehand, before the car passes through it, in order to perform the parking maneuver frontally. In the simulation, a camera was used for identification, however, a machine learning system was not implemented, using the measurements and tracking carried out automatically by the driving scenario instead. These measurements return coordinates (x, y) that can be used to create the cost map. The cost map was created through a build in function on matlab.

With the cost map created, it was then necessary to rethink the logic of movement and the waypoints passed to the trajectory function. The other parkings

occurred in the opposite direction, with the car in reverse, however this car needs to enter the space frontally. The driving scenario reverses the angles depending on the direction of movement, so the waypoints, angles, and speeds passed to the trajectory function were modified. It was possible to perform the parking and exit from the space, figure 35 shows the vehicle entering a space.

Figure 35 - Angle parking in bird's-eye plot



Source: Own authorship (2023)

3.3 Small scale prototype

The small-scale prototype was designed to be a testing and validation platform with the aim of empirically testing the simulations and codes developed for the APS.

3.3.1 Hardware

This section outlines the process used to design and prototype autonomous parking systems. It covers the plan, design and implementation of a small-scale prototype, as well as the programming of the prototype side and computer side design to filter and use the data acquired. The small-scale prototype was designed to be a testing and validation platform with the aim of empirically testing the simulations and codes developed for the APS.

3.3.1.1 Arduino Uno

To connect the sensors and actuators, a microcontroller is required. In this case, an Arduino Uno was used due to its availability and the option to validate the system on a low-cost platform with limited processing capabilities. To operate the system, the Arduino was connected to a computer via USB, which provides power and data connection. Information is transferred via the USB port using the Serial protocol. Therefore, a program had to be developed to receive, filter, and process the data on the computer.

The Arduino Uno acts as an interface between the sensors and actuators, allowing for real-time data collection and transmission. Sensors such as cameras and ultrasonic sensors are connected to the Arduino's analog and digital inputs, while actuators like steering control motors are connected to the outputs. The microcontroller is responsible for reading the sensor signals, processing them, and sending appropriate commands to the actuators.

3.3.1.2 Motor's driver

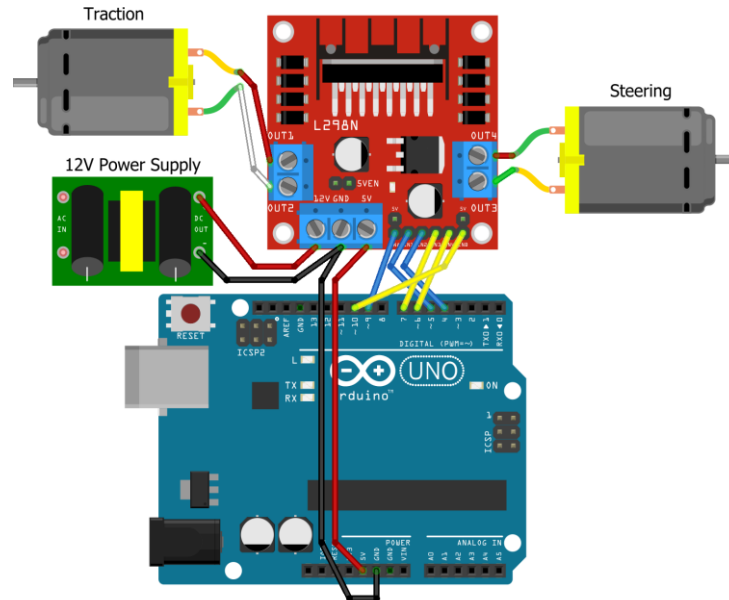
The prototype of the autonomous parking system was constructed using a remote-control car equipped with two motors, one for rear-wheel drive and another for steering. This choice was motivated by the availability of the car in the laboratory, making it a cost-effective solution for the initial development stages. The car had been previously used in other projects, and its reuse allowed for efficient utilization of resources.

To evaluate the performance of the traction and steering motors, they were initially connected to a dual H-bridge L298N module. This module enabled the testing of motor control capabilities and ensured the motors could operate within their specified parameters. An external 12V power source was employed to supply power to the motors, preventing any potential overload on the Arduino.

Subsequently, a C code was developed for the Arduino microcontroller. This code facilitated the control of both motors by setting the minimum achievable speed for the traction motor, which had inherent limitations due to its quality. The Arduino's PWM (Pulse Width Modulation) pins were utilized to regulate the power supplied to the motors, with pin 9 assigned to the traction motor and pin 10 dedicated to the steering

motor. Figure 36 depicts the interconnections between the two motors, the L298N module, the Arduino Uno, and the 12V power source.

Figure 36 - Schematic with L298N



Source: Own authorship (2023)

In the project, the PWM values were configured within a range of 0 to 255, where 0 represented 0% power and 255 represented 100% power. To alter the direction of both the traction and steering motors, the digital output activation pins, mla and mlb, were alternated accordingly. The schematic illustrated in figure 37 remained unchanged throughout the subsequent stages of the project. Tests were performed to determine the minimum PWM signal required for the proper functioning of the traction and steering motors. Incremental values were added, starting from 0, until the car exhibited uniform and stable motion.

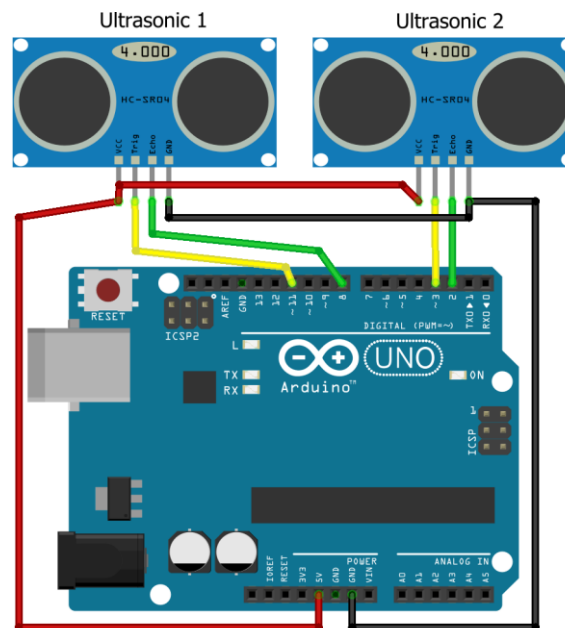
3.3.1.3 Ultrasonic sensors

The initial phase of the research involved the utilization of two ultrasonic sensors (HC-SR04) connected to the Arduino board, as illustrated in figure 37. These sensors are popular due to their low cost and good performance in short-range applications. The system was planned with two sensors to facilitate the utilization of multiple ultrasonic sensors simultaneously in future steps of the project, allowing for system validation.

The utilization of ultrasonic sensors enables a comparative analysis with the LiDAR sensor. By acquiring data from both sensors and conducting a thorough

analytical comparison, it is possible to evaluate the performance of the LiDAR sensor. The schematic in the figure 37 illustrate the idea. This comparative analysis assesses the accuracy, precision, and reliability of the LiDAR sensor in detecting and measuring distances, particularly in comparison to the measurements obtained from the ultrasonic sensors. Such a validation process aids in determining the suitability of the LiDAR sensor for the specific requirements and objectives of the autonomous parking system, which are precise parking spot detection, measurement, and classification.

Figure 37 - Schematic with HC-SR04 ultrasonic sensors



Source: Own authorship (2023)

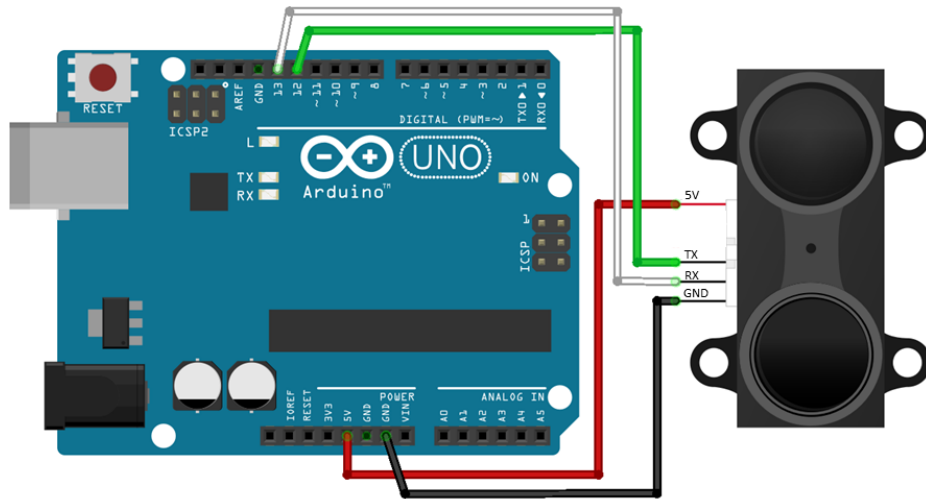
To ensure the proper utilization of the sensor, a code was developed to enable interrupt-driven functionality specifically designed for the ultrasonic sensors. This was achieved by utilizing the NewPing library, which provides a convenient and efficient way to handle the ultrasonic data. The code implementation allows for timely and accurate measurements, optimizing the overall performance of the ultrasonic sensors in the system. It is important to note that, in subsequent sections of this document, the disconnection of the ultrasonic sensor 2 after the initial tests, as well as the connection of an encoder to the same pins, will be further discussed.

3.3.1.4 LiDAR sensor

The LiDAR sensor used is the TF Mini Plus model manufactured by Benewake. It was specifically selected for its ability to provide highly accurate distance readings of

up to 12 meters and its IP67-rated enclosure. The IP67 rating is particularly beneficial for the project, as it ensures the sensor's protection against dust and water ingress, making it suitable for testing and implementation in real-world vehicle scenarios.

Figure 38 - Schematic with the TF Mini Plus LIDAR sensor



Source: Own authorship (2023)

Figure 38 showcases the schematic representation of the TF Mini Plus LIDAR sensor's connection configuration. In this setup, the sensor operates in serial communication mode, with the green and white output wires connected to pins 12 and 13, respectively, of the Arduino Uno. Another method of connecting the TF Mini Plus LIDAR sensor is by directly linking it to the computer using a TTL to serial converter board connected to the USB-A port. This connection setup allows for seamless communication between the sensor and the computer, facilitating the use of Benewake's software for testing, configuration, and validation of the sensor.

3.3.1.5 Rotary encoder

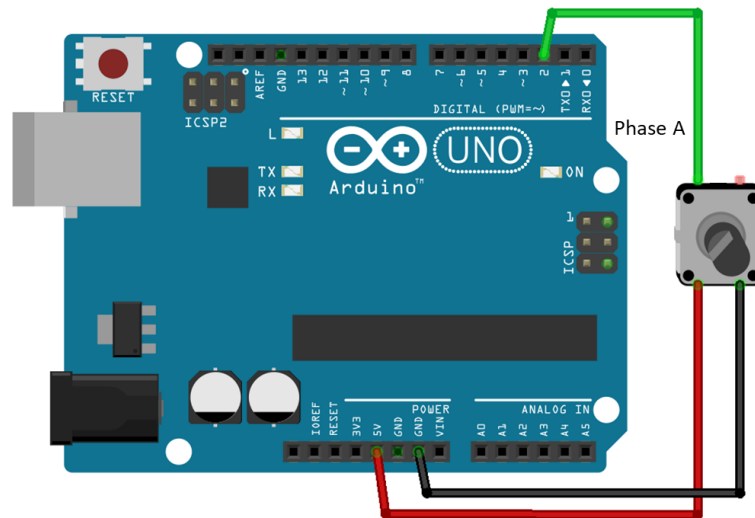
The LPD3806 encoder plays a crucial role in measuring the rotation of the vehicle's wheels. It is responsible for providing precise information about the angular velocity of the wheels, enabling the system to calculate the distance traveled and, consequently, determine the size of the parking space.

The connection of the LPD3806 encoder to the Arduino is simple and direct. The phase A of the encoder is connected to pin 2 of the Arduino, while phase B remains disconnected. In this specific context, there is no need to identify the direction of movement, and therefore, phase B is not used. The identification of the direction of

movement can be performed later, if necessary, using the information from phases A and B. The encoder can be powered by the Arduino, considering its operating range of 5-24V.

Figure 39 illustrates the wiring diagram of the LPD3806 encoder to the Arduino, providing a clear visual reference for cable positioning and the correct connection to pin 2 of the Arduino, which should be set to pull-up in the code.

Figure 39 - Schematic with the Arduino uno and an encoder



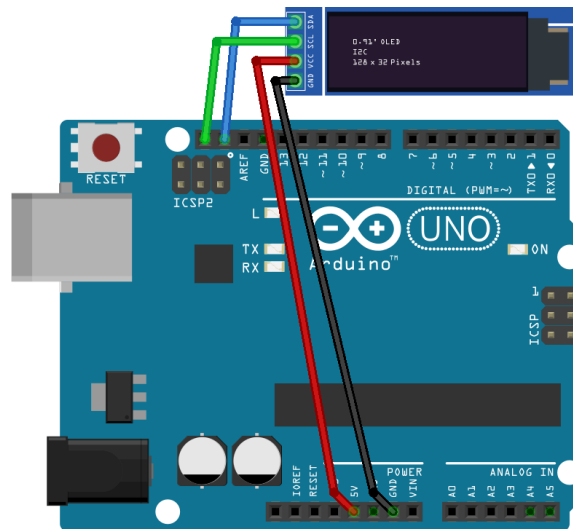
Source: Own authorship (2023)

This simplified configuration of the encoder in the prototype is suitable for the current needs of the developing APS. The rotation of the wheels is measured by the encoder through phase A, which generates electrical pulses proportional to the rotational movement. These pulses can be counted and processed by the Arduino to determine the distance traveled and adjust the vehicle's maneuvers.

3.3.1.6 OLED display

To facilitate real-time visualization and easy comparison of the measured distances, an OLED 128x32 display was integrated into the system. The display provides a convenient way to visually monitor the sensor readings. For this purpose, the Adafruit_SSD1306 and Adafruit_GFX libraries were employed. The OLED display utilizes the I2C connection protocol, ensuring efficient and reliable communication. The connection scheme between the display and the Arduino Uno is illustrated in figure 40, providing a clear reference for the correct wiring configuration.

Figure 40 - Schematic with OLED display



Source: Own authorship (2023)

3.3.1.7 Arduino code

The Arduino code was developed in a Arduino variant of C++, integrating the modules previously. The code contains detailed comments explaining the purpose and operation of each section. It defines the necessary variables and allows for the customization of parameters such as traction motor speed, activation time, and sensor acquisition time through adjustable predefined values.

Within the main loop of the code, there is a movement function that activates the traction motor. An interrupt-driven approach was implemented to obtain data from the ultrasonic and LiDAR sensors, ensuring data acquisition occurs at a frequency preconfigured within the code. This frequency can be adjusted to suit the specific requirements of the system. It is important to note that the code handles the number of ultrasonic sensors used as a parameter to avoid cross-interference between devices.

The default frequency used in the code was set to 20 Hz. This frequency provides a sufficient amount of information from the sensors to perform the required functions, considering that the prototype operates at low speeds during parking maneuvers. The choice of this frequency was also influenced by the Arduino Uno microcontroller's capability to acquire data at much higher frequencies consistently, allowing for potential scalability in future developments.

To activate the motors, the appropriate output variable (mla or mlb) is set to a high logic level, with the direction defined accordingly. It is ensured that only one of

these variables is set to high at a time to prevent conflicts. A function is responsible for verifying the reception of the return signal by the ultrasonic sensors and calculating the distance to the detected object in centimeters.

Another function consolidates the relevant variables for serial transmission at a predefined interval determined by a variable. Additionally, a condition is included to send the "END" signal after a specified duration. This allows the computer program to identify the completion of the data acquisition transmission.

3.3.1.8 Testing scenario

A testing scenario was carefully designed to evaluate the prototype's capability to detect and measure three types of parking spaces: parallel, perpendicular, and angled. Photograph 9 illustrates the test environment that was specifically created for this purpose. The primary objective of this testing was to assess the prototype's performance in accurately identifying these parking spaces on a small-scale basis and accurately measuring their length. By conducting these tests, valuable insights were gained regarding the prototype's effectiveness and its potential for real-world applications.

In addition, the testing scenario was also created to validate the parking space classification process. By detecting and accurately classifying the different types of parking spaces, the prototype's ability to differentiate between parallel, perpendicular, and angled parking spots was assessed. This validation step was crucial in ensuring that the autonomous parking system could correctly identify and categorize parking spaces in real-world scenarios. It provided valuable data and feedback to refine and improve the classification algorithm, ultimately enhancing the overall performance and reliability of the system.

Photograph 9 - Types of parking spaces in the testing scenario



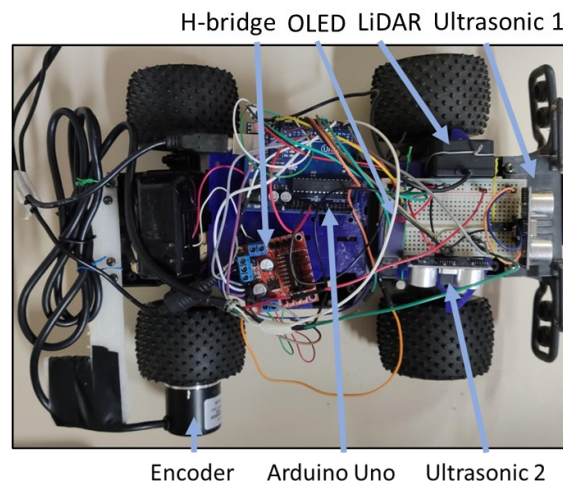
Source: Own authorship (2023)

3.3.1.9 Components positioning

The positioning and installation of components in the prototype were carefully considered to ensure optimal performance and functionality. Each component was strategically placed to fulfill its designated role in the autonomous parking system.

Starting with the ultrasonic sensors, they were initially positioned as shown in photograph 10, one at the front of the vehicle, simulating an obstacle detection sensor, and another at the front right, simulating a parking space detection sensor. This configuration allowed for simultaneous testing of both functionalities.

Photograph 10 - The initial arrangement of components in the small-scale prototype



Source: Own authorship (2023)

The LiDAR sensor, known for its high-precision distance measurement capabilities, was strategically positioned on the front right side of the prototype. This specific placement was carefully chosen to ensure optimal data acquisition for detecting parking spots. By being situated in this position, the LiDAR sensor effectively scanned the surrounding environment, capturing detailed information that played a critical role in the autonomous parking system. This data was utilized to create a cost map and generate a virtual representation of the environment, both of which were essential for the subsequent stages of the project.

The encoder was meticulously positioned on the left rear wheel of the prototype, utilizing a custom-designed 3D-printed plastic adapter. This adapter served the dual purpose of securely mounting the encoder and facilitating precise measurement of wheel rotation. The choice to install the encoder on the rear wheel was primarily based on its ease of attachment and stability. This location ensured that the encoder accurately captured the rotation of the wheels, providing essential

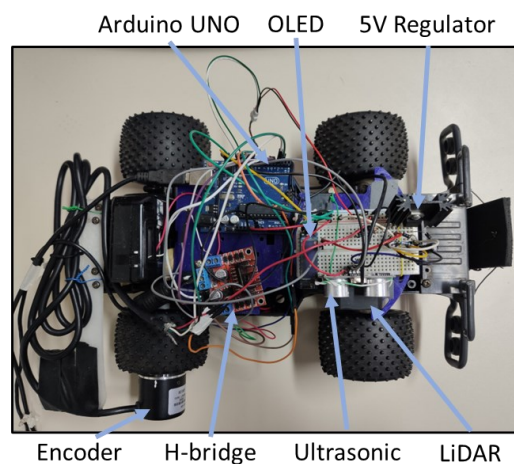
feedback for the autonomous parking system's navigation and motion control algorithms. It should be noted that due to the limited number of digital ports available on the Arduino Uno, it was necessary to disconnect and remove one of the ultrasonic sensors in order to accommodate the connection of the encoder. This trade-off was made to ensure the successful integration of all components within the system's hardware constraints.

The Arduino was placed in a convenient and accessible location, ensuring ease of programming and monitoring during the development and testing stages. This positioning enhanced the efficiency and flexibility of the system, allowing for efficient debugging, adjustments, and improvements as needed. Additionally other essential electronic components were strategically positioned on the central plate of the vehicle. This placement allowed easy access to all the components, facilitating quick architectural modifications if necessary. Careful attention was given to establishing proper electrical connections to enable smooth communication between the components and the control system.

Furthermore, the OLED display, which provided real-time distance visualization, was strategically positioned next to the sensors, ensuring clear line of sight. This placement facilitated easy monitoring and debugging of the detected distances from the sensors. By having the display in close proximity to the sensors, it allowed for convenient observation of the real-time data, ensuring accurate assessment of the system's performance.

All the mentioned components can be seen in photograph 11 in their final positioning.

Photograph 11 - Final prototype view



Source: Own authorship (2023)

3.3.2 PC software

Aiming to process data obtained from sensors, perform parking detection, measurement, and classification, as well as path planning and all necessary logic, a software written in Python has been developed. This software offers an intuitive approach for testing, allowing for analysis and saving of captured data for later examination. With the purpose of optimizing the performance of the parking assistance system, the software becomes a fundamental component for extracting valuable information from the sensors. By processing data efficiently and reliably, it enables accurate parking spot detection, precise measurement of their dimensions, and appropriate classification of available parking types.

Written in Python, a widely-used programming language, the software provides an intuitive interface that facilitates testing and experimentation. This intuitive approach allows for easier interaction with the system, enabling users to conduct tests, capture data, and analyze it later. Additionally, the ability to save captured data is an important functionality of the software. This allows for detailed and in-depth analysis of the data at a later time, aiding in system refinement and obtaining valuable insights for future improvements.

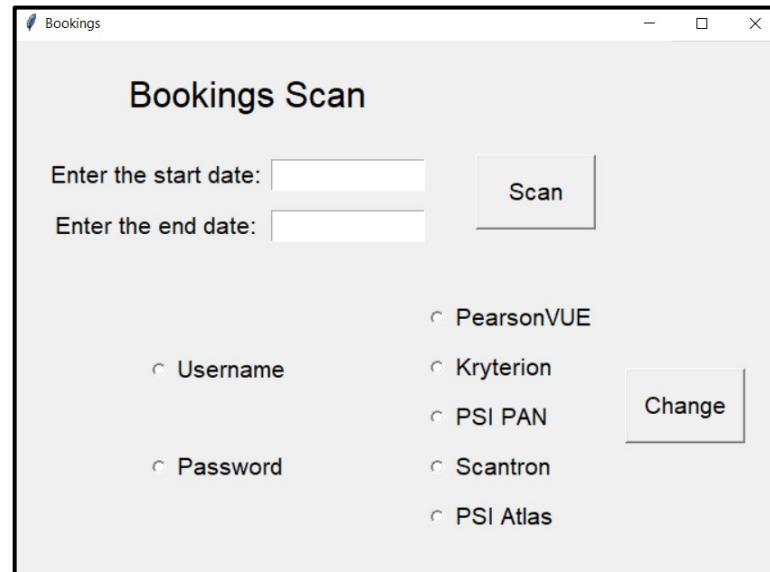
With the development of this software, the aim is to provide an efficient and flexible tool for data processing, parking spot detection, and path planning. It becomes an essential component to enhance the accuracy and efficiency of the parking assistance system, enabling a more intuitive and reliable experience for users.

In the following sections, we will delve deeper into the functionalities and benefits of this Python-written software, highlighting how it contributes to intuitive testing, data capture and analysis, as well as its fundamental role in the processing and logic of the parking assistance system.

3.3.2.1 Interface

The interface is an essential part of many software applications, and in the current program, it was developed to facilitate data acquisition, representation, and saving, as well as to simplify the testing process with the prototype. The *tkinter* library was used for developing the interface, which allows for easy addition of buttons, checkboxes, text boxes, and other elements, and subsequently associating functions with each element. An example could be seen in the figure 41.

Figure 41 - Tkinter example



Source: Adapted from Stack Overflow (2021)

The main class of the program is the "Interface" class, which serves as a central point from where all other program functions are called. Texts were instantiated to provide information, and buttons were created to trigger essential functions, such as reading data through the serial port. An option menu was also included to enable the selection of the serial port from which the information will be received.

To allow for the selection of sensor data to be collected, there are two checkboxes, one for selecting data from the ultrasonic sensor and another for selecting data from the LiDAR sensor. Additionally, to display the results of parking space detection, measurement, and classification, a table element was implemented. This table is used to represent the calculated values.

3.3.2.2 Data reception

One of the fundamental steps of the program is data reception. Through these classes, the data acquired by the prototype is received and organized into variables within the program, to be subsequently used by other classes and functions. The data reception process is divided into two classes: real-time data reception from the prototype, triggered by the "Read Data" button, and the reading of previously acquired data from a CSV file, triggered by the "Read CSV" button.

Firstly, the real-time data reception class establishes a connection with the Arduino UNO through the computer's USB port, utilizing the selected serial port specified by the user through the program's interface. The buffer is then cleared, and

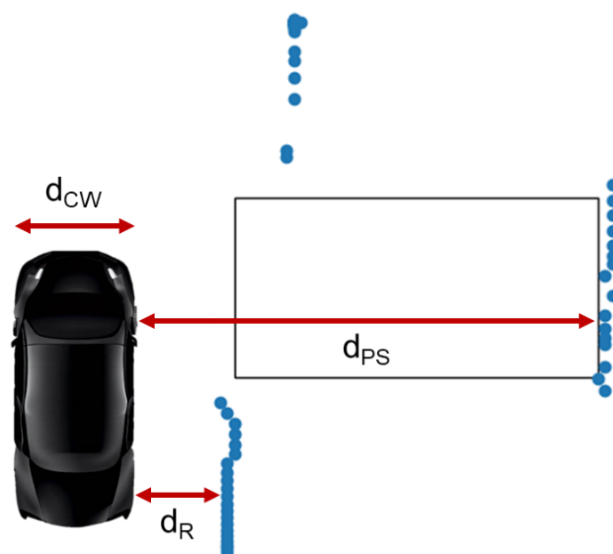
data reception begins. A function searches for arrow symbols "<>" since these symbols were defined in the Arduino code to encapsulate the messages, ensuring the reading of a complete message during each iteration.

Upon receiving a complete message, it is parsed and separated into variables using semicolons as delimiters. The received message includes the acquisition time in milliseconds, distance traveled in meters, distance detected by the LiDAR sensor in meters, and distance detected by the ultrasonic sensor in meters.

When it is desired to open a previously acquired file, the program reads the data from a CSV spreadsheet, separating it into variables in a similar manner to the real-time data acquisition class. Both classes enable the reception of data, either in real-time or from pre-acquired files, ensuring that the acquired information is properly organized and available for further processing and analysis within the program.

3.3.2.3 Parking spot detection

With the data stored in the software's memory, they are used as parameters for the parking detection class, which performs the detection using the logic of analyzing the acquired points. The first step is to identify the reference distance, which is defined as the initial distance from the car to the first obstacles, as shown in figure 42 as d_R .



Source: Own authorship (2023)

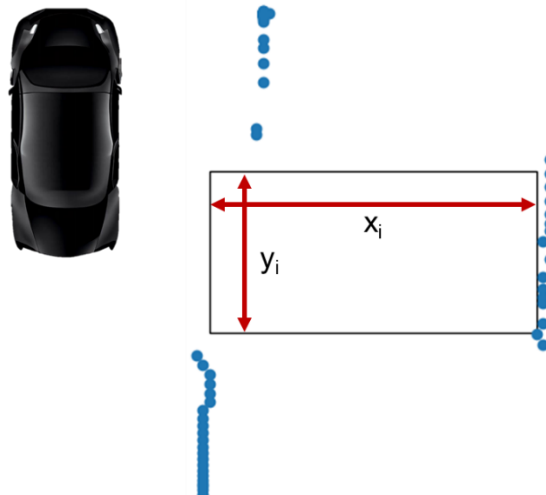
This distance is then used for comparison to check the condition of equation 23. If the vehicle width, reference distance, and safety distance are greater than the detected distance, a parking space is identified.

$$d_{PS} > d_R + d_{CW} + d_S \quad (23)$$

3.3.2.4 Parking spot measurement

After identifying a parking space, its components are measured both longitudinally and laterally, obtaining the values of x and y for the parking space. Figure 43 represents the values to be found.

Figure 43 - Measurement of parking spot size



Source: Own authorship (2023)

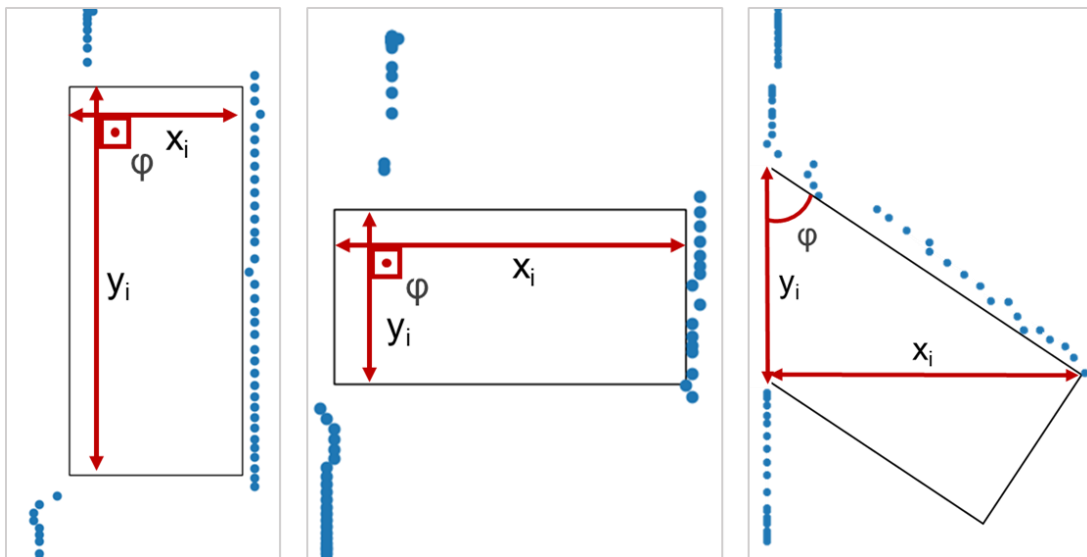
The lateral distance x_i , is calculated by averaging the internal distance values within the parking space measured by the sensor, values between the indices i_b e i_e . To measure the longitudinal distance y_i , it is calculated using encoder data by subtracting the displacement value at index i_e from the displacement value measured at index i_b . Equations 24 and 25 demonstrate the calculations.

$$x_i = \sum_{i_b}^{i_e} \frac{x}{i_e - i_b} \quad (24)$$

$$y_i = y_{ie} - y_{ib} \quad (25)$$

After obtaining the calculated measurements, a preliminary classification is necessary because the parking space size calculation is divided into two methodologies: one for parallel and perpendicular parking spaces and another for angled parking spaces. To select the appropriate methodology, the angle of the parking space distance data is calculated, as shown in figure 44. This angle is referred to as φ (phi).

Figure 44 - Angle φ in the three types of parking



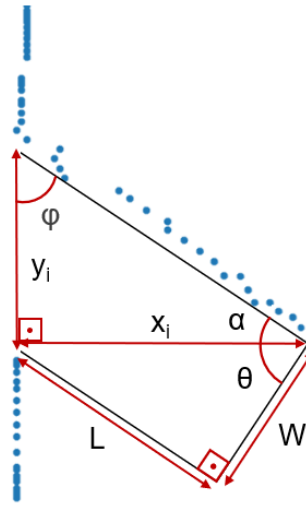
Source: Own authorship (2023)

The angle φ is calculated using the trigonometric relationship between the measurements x_i e y_i the arctangent function is applied, as shown in equation 26. When φ falls within the range of 20° to 70° , the parking space is classified as angled. When the angle is outside this range, the parking space proceeds to the next classification stage, which aims to identify whether it is parallel or perpendicular.

$$\varphi = \text{arc tg} \left(\frac{x_i}{y_i} \right) \quad (26)$$

When the parking space is classified as parallel or perpendicular, the parameters x_i e y_i represent the size of the space. However, when the parking space is classified as angled, additional calculations need to be performed to determine the size of the space. These calculations are illustrated in figure 45, where the variables L (length) and W (width) are introduced.

Figure 45 - Relevant measurements in the calculation of an angled parking space



Source: Own authorship (2023)

Equation 27 was used to find the relationship between φ and the angle α . Subsequently, by substituting α into the trigonometric relationship in equation 28, it is shown that the angles θ and φ are equal. Thus, by using the sine and cosine of the angle φ it is possible to obtain the measurements L and W as shown in equations 29 and 30.

$$\varphi + \alpha + 90 = 180 \quad (27)$$

$$\alpha = 180 - 90 - \varphi = 90 - \varphi$$

$$\theta + \alpha = 90$$

$$\theta = 90 - \alpha \quad (28)$$

$$\theta = 90 - 90 + \varphi$$

$$\theta = \varphi$$

$$\sin \varphi = \frac{x_i}{L} \quad (29)$$

$$L = \frac{x_i}{\sin \varphi}$$

$$\cos \varphi = \frac{W}{x_i} \quad (30)$$

$$W = x_i \cos \varphi$$

3.3.2.5 Parking spot classification

The classification of the parking space takes into account the obtained measurements x_i , y_i e φ . When the calculated angle φ is outside the range of 20° to 70° , the parking space can be parallel or perpendicular, as in these cases the angle φ would be 90° . Thus, the parking spaces are classified based on the relationship between x_i e y_i . When the ratio is less than 1, the parking space is classified as parallel, and when the ratio is greater than 1, the parking space is classified as perpendicular. Table 4 summarizes this idea.

Table 4 - Classification table of parking space types

Type of parking spot	Mathematical relation
Parallel	$(\varphi < 20 \text{ or } \varphi > 70) \text{ and } \frac{x_i}{y_i} < 1$
Perpendicular	$(\varphi < 20 \text{ or } \varphi > 70) \text{ and } \frac{x_i}{y_i} > 1$
Angled	$20 < \varphi < 70$

Source: Own authorship (2023)

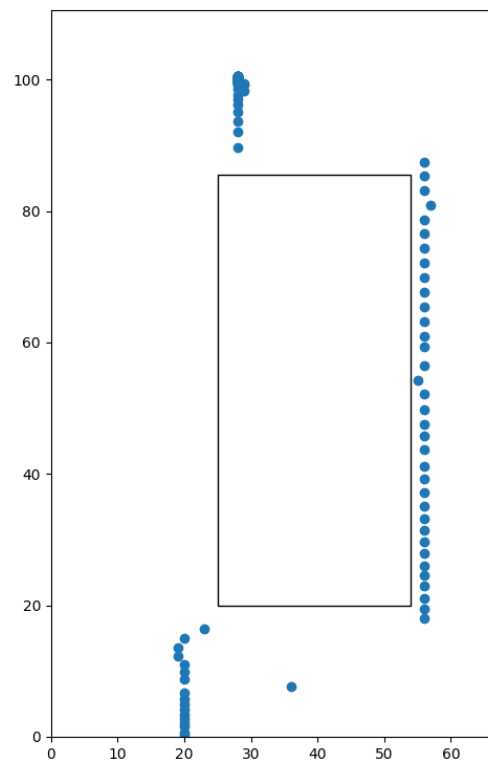
After the classification, calculations can be performed to determine whether the detected parking space has sufficient size for safe parking. The size of the parking space is compared with the size of the car, considering a safety coefficient. By adding the necessary safety margin to the measurements, the suitability of the parking space can be evaluated.

3.3.2.6 Plotting the data

To analyze the data, a plotting function was created using the matplotlib library. The class was designed to enable real-time plotting or plotting from previously saved data. The data is structured with the x-component representing distance measurements obtained from the distance detection sensors. It is possible to plot data from either the ultrasonic sensor, the LiDAR sensor, or both simultaneously. Two checkboxes in the interface are used for this selection.

When the "simulation" button on the interface is pressed, the preloaded data is plotted as points using the scatter function. Additionally, a custom function is called to visually represent parallel and perpendicular parking spaces by drawing rectangles with dimensions corresponding to the identified parking space. Figure 46 provides an example of a parallel parking space graphically represented by the simulation function of the program.

Figure 46 - Parallel parking space plot using real data



Source: Own authorship (2023)

3.3.2.7 Saving the data

One of the needs encountered during the project's development was to store the data acquired by the sensors for later analysis. The "save" class of the program serves this purpose by saving the data in a CSV file format, which is widely accepted by various programs, thus facilitating data analysis. The data is saved with the file name being the current date and time obtained from the computer system on which the software is being executed.

3.4 Chapter's considerations

Chapter 3.1 described the Bibliography Analysis and Literature Review methodology using the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) methodology to obtain a relevant bibliographic portfolio for the research. The stages of identification, selection, eligibility, and inclusion of studies were conducted, resulting in a total of 88 studies. These studies underwent bibliometric analysis, revealing publication trends over the years. The most frequent keywords were identified, as well as the key authors in the field. These findings provide valuable insights for future research related to the topic of autonomous parking.

Chapter 3.2 discussed the simulation of the proposed APS. Simulation plays a crucial role in the development of the APS, allowing for a thorough analysis of its capabilities, efficiency, and safety in a controlled environment. The chapter covers software selection, parking space identification, maneuvering in and out of parking spaces, creating a cost map, and path planning for different parking scenarios.

Chapter 3.3 presented the small-scale prototype, showcasing the hardware and software requirements. It demonstrates the development methodology of a testing platform, explaining how each component of the prototype was planned. Additionally, the methodology for developing the parking calculation software is exposed, which was implemented in Python. The software aims to provide identification, measurement, and classification of parking spaces, as well as display them in the interface and save the relevant data.

4 RESULTS

In this section, the results obtained by applying the mentioned methodology will be presented, considering all the stages of the APS.

4.1 Bibliography Analysis and Literature Review

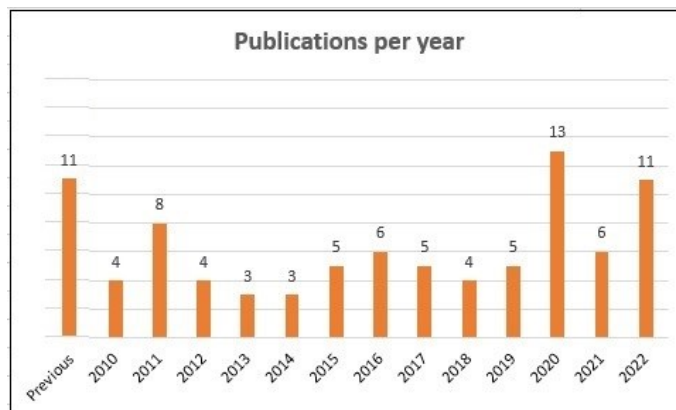
Through a meticulous bibliography analysis, the researcher identifies and selects key works that have played a pivotal role in shaping the research field. These sources serve as the foundation for building the theoretical framework and contextualizing the research within the existing body of knowledge.

The literature review aims to critically assess the findings, methodologies, and theoretical perspectives presented in the selected sources. It involves synthesizing the information from various studies and identifying common themes, patterns, and gaps in the literature. By doing so, the researcher establishes the basis for the current study, highlighting its significance and contribution to the field.

4.1.1 Publications per year analysis

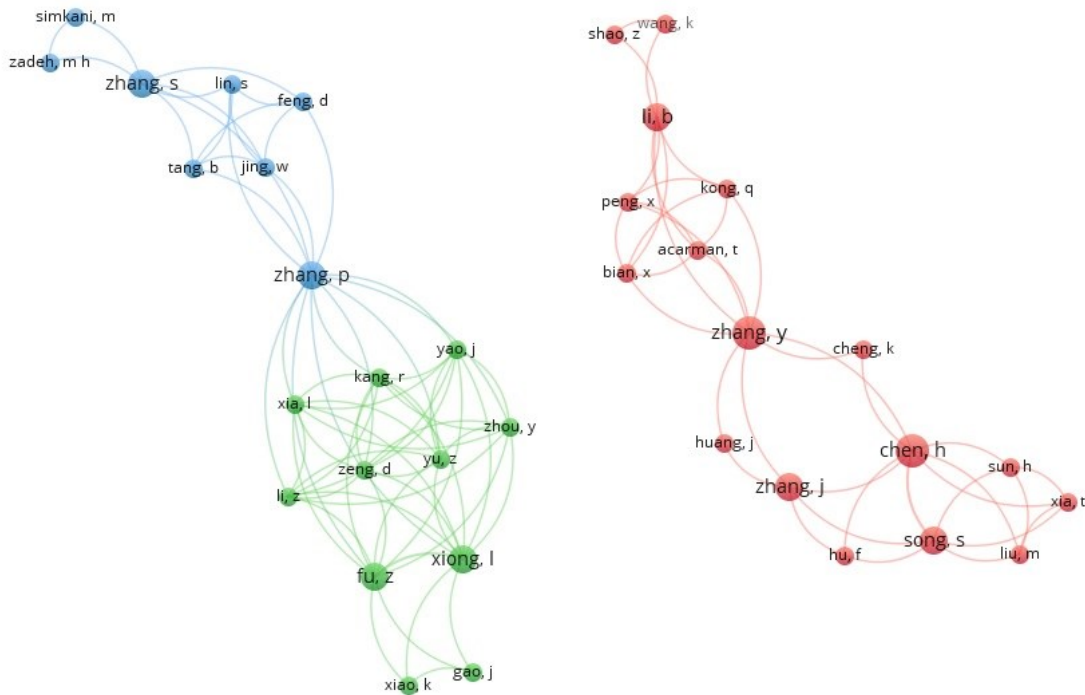
Using Microsoft Excel software, it was possible to map the development of the theme over the years. Figure 47 represents this analysis. It is possible to perceive that the topic has taken on a greater proportion from 2020. The more recent articles have an approach more related to the use of Artificial Intelligence (AI), Motion Planning, and Algorithms such as RRT*, while the older ones aimed at classical methods, mathematical and geometric approaches to path planning.

Figure 47 - Publications per year



Source: Own authorship (2023)

Figure 49 - Author Analysis



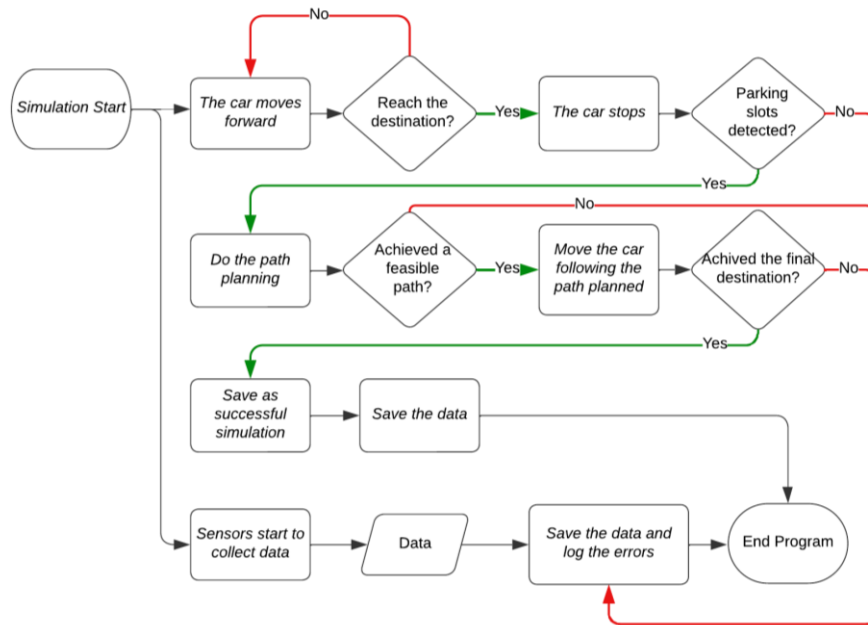
Source: Own authorship (2023)

4.2 Simulation

The comprehensive simulation of the complete system allows for a thorough analysis of the performance and effectiveness of the APS in different scenarios and conditions. By analyzing the results obtained from the simulation, a clear understanding of the APS performance can be gained, enabling the identification of strengths and areas that require improvement. This information is valuable for the continuous development of the system, ensuring its efficiency, reliability, and safety.

Figure 50 presents a state diagram that illustrates the simulation process of an APS. The diagram consists of a series of states, represented by nodes, connected by transitions, represented by directional arrows. Each state represents a specific step in the autonomous parking process, while the transitions indicate the state changes that occur as the system progresses.

Figure 50 - Diagram of the APS simulation process

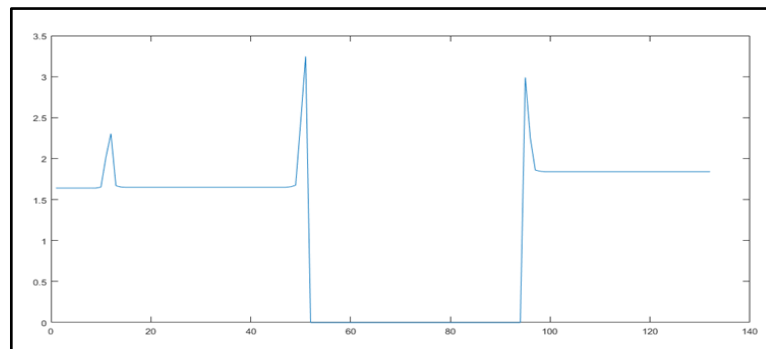


Source: Own authorship (2023)

4.2.1 Parking spot identification

As a result of the scenario created in section 3.2.2, data regarding the distance measured with the simulated ultrasonic sensor were obtained. In this way, the distance between the side of the ego car and the nearest object to the right is obtained. In figure 51, the waveform generated by the sensor when passing through the first two cars, the parking space, and then another car can be observed. The X-axis represents the number of samples and the Y-axis represents how many meters the sensor identified an obstruction.

Figure 51 - Sensor data with longitudinal movement of the ego vehicle



Source: Own authorship (2023)

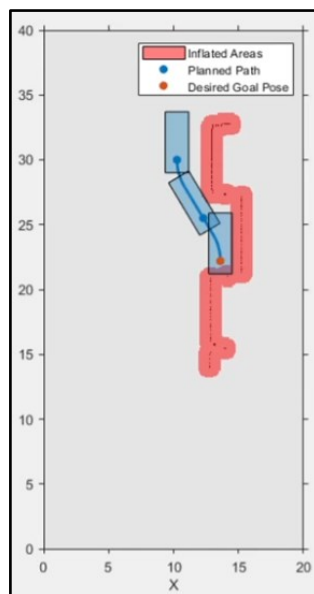
In figure 51, the first plateaus in the signals can be observed, separated by a peak. These are then classified by the code as the first two parked cars. Then there is

a new peak in the detection and then a large drop, showing the value zero. This signal represents that the sensor is not identifying any objects and is therefore classified as a parking space. After the parking space, there is a peak and another higher plateau, close to 2 meters, which is the third parked car. A problem was identified with the sensor, because in short, the non-identification of obstruction implies that the variable is not returned, which is then defined as zero by the code. In this way, it is not possible to identify whether the sensor is detecting an empty space or if it is experiencing problems, not returning any variable.

4.2.2 Path planning

During and after the development of the merge algorithm between the native path planning system of MATLAB and the simulation in the driving scenario, tests were carried out. It was identified that sometimes the trajectory was too complex to follow, with no natural movement. This problem was solved by increasing the minimum and maximum number of attempts of the path planning algorithm, as this allows more trajectories to be found, and through the internal optimization of the algorithm, human-like trajectories are generated. The result can be seen in figure 52, with the trajectory to a parallel parking space.

Figure 52 - Trajectory generated for parking a vehicle in parallel



Source: Own authorship (2023)

Thus, the parking occurred as expected, and after a few seconds, the car left the parking space. However, various bugs and lack of optimization hindered this step

in particular. The "trajectory" command has several problems, and a programming logic was implemented to avoid values that caused error for no apparent reason during execution.

4.2.3 Parallel parking

Through previous tests, concise results could be identified in the utilization of the described data acquisition and path planning methods to identify, measure, and classify a parallel parking space, as well as to perform the parking using path planning. Some of the identified problems consist of incorrect measurement of the parking space size due to the considerably high FOV of the ultrasonic sensor. In addition, erratic execution of path planning paths was observed.

Such problems can be resolved by selecting a sensor with a smaller FOV for the measurement of the parking space size. For path planning, other configurations were experimented with, which reduced the variations in the generated paths.

4.2.4 Perpendicular parking

In the perpendicular parking test, it is understood that choosing the parameter of the car's width, that the measurement error will make up a significantly larger portion of the measure. As the measured space parallel to the car is smaller, the error remains proportional. Solutions found in the literature for this problem are as follows:

- 1) Use of cameras and radars to perform sensor fusion and reduce measurement error;
- 2) Replacement of the ultrasonic sensor used for parking space measurement with a lidar sensor.

Due to the possibility of a very low viewing angle of some lidar sensors, this characteristic theoretically allows almost complete elimination of the error in measuring parking space size, as sensors of this type with a field of view of less than two degrees can be found, which compared to the fifteen degrees of ultrasonics represents a significant improvement.

These solutions are also effective against the problems encountered in parallel parking tests, as these problems are the same.

4.2.5 Angled parking

For the angled parking scenario, it was necessary to adapt the simulation model as shown in section 3.2.9 of the methodology. The ultrasonic parking detection sensor was replaced with a vision sensor. However, using the vision sensor of the driving scenario, a cohesive identification of the space was not obtained, as the software sends few points of the position of the identified cars, and there is no configuration to change this behavior. Therefore, it would be necessary to use another sensor in the simulation, such as radar or lidar. This falls outside the scope determined for the identification of the space, being a software problem.

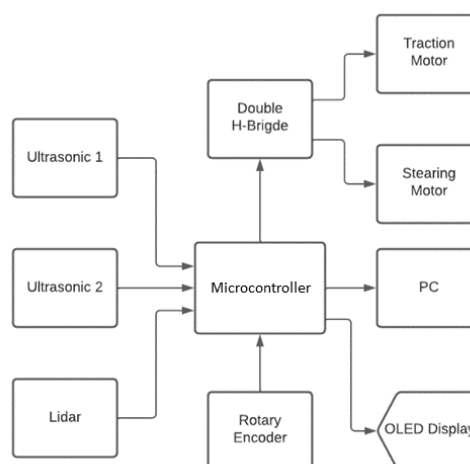
4.3 Small scale prototype

Conducting tests and obtaining results in a small-scale prototype is of utmost importance, both in terms of hardware and software. These tests allow for the evaluation of system performance and effectiveness in a controlled environment, identifying potential issues and enabling improvements before large-scale implementation.

4.3.1 Hardware

By integrating the sensors and actuators separately, it was possible to perform the necessary tests to validate each component of the prototype, as well as the codes for the Arduino microcontroller. Figure 53 illustrates the entire system in a block diagram format.

Figure 53 - System block diagram



Source: Own authorship (2023)

4.3.1.1 Electric motors

In order to ensure the proper functioning of the prototype, tests were conducted to evaluate the traction and steering motors. For the traction motor, the objective was to determine the minimum PWM value required to achieve a consistent and uniform movement of the car. Through the tests, it was discovered that a PWM value of 40, equivalent to 15.69% of the total motor force, represented the minimum activation speed at which the car would start moving.

A similar test was carried out with the steering motor. However, it was found that only the maximum PWM activation value of 255 was capable of achieving the desired result of effectively steering the front axle.

4.3.1.2 Ultrasonic sensors

With the aim of validating the minimum and maximum detection distances with ultrasonic sensors, tests were conducted. It was possible to identify that there is high precision in identifying distances from 10 cm to 2 m, with data becoming increasingly uncertain beyond two meters, with inconsistent readings.

Thus, it is understood that ultrasonic sensors may have lower limits than those found in datasheets, since the HC-SR04 ultrasonic sensor has a maximum measuring distance of 4 meters. Through tests, it was found that this information does not correspond to reality. This is due to the quality of the sensors, as there is no renowned manufacturer that produces this model of ultrasonic sensor.

Tests were conducted using two ultrasonic sensors at the same time and data was obtained that corresponds to their correct operation.

4.3.1.3 TFmini Plus LiDAR sensor

The sensor's functionality was evaluated by connecting it to a computer via a serial conversion board and utilizing the manufacturer's dedicated software, Benewake. This software provides the capability to test the ultrasonic sensor under its factory conditions and offers the flexibility to adjust various settings, including frame rate (ranging from 1 to 1000 Hz), trigger detection, output format, baud rate, and more. The conducted tests were performed using the sensor's default settings.

4.3.1.4 LPD3806 rotary encoder

Tests were conducted with the LPD3806 encoder to validate the developed code. Initially, the goal was to test if the encoder's rotation count was accurate. To do this, the prototype's tire was marked with paint, allowing manual rotations to be performed and the code's output to be verified. It was found that the Arduino's output in relation to the rotations was correct.

Next, the calculated distance traveled by the Arduino code was tested and compared to the physically measured distance. It was observed that the error was below 3%, indicating acceptable values for further calculations.

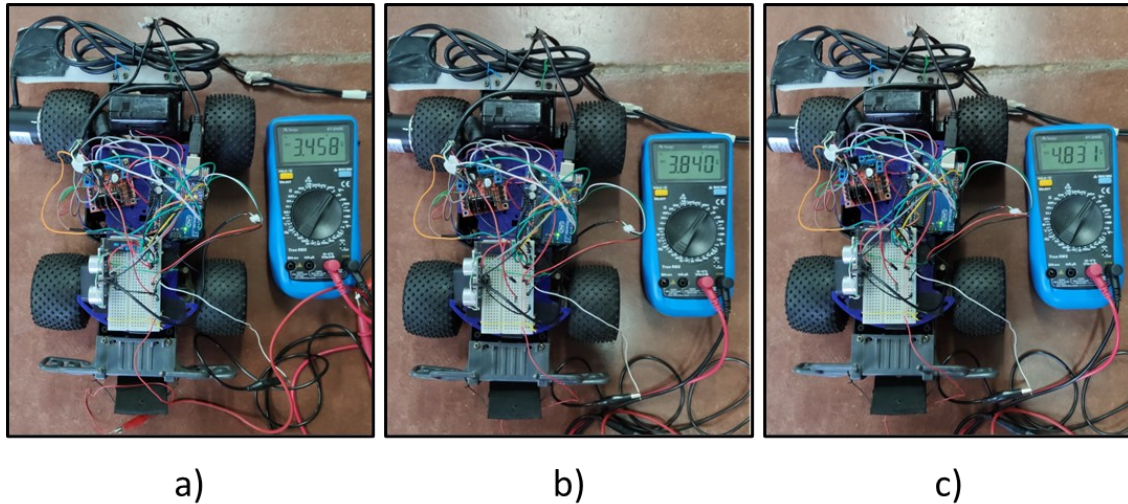
4.3.1.5 Measurement Error

When connecting the LiDAR sensor TFmini Plus to the Arduino UNO, measurement errors were encountered during tests. These errors occurred when attempting to measure the distance to an object with a matte black surface. This type of object is known to be one of the most challenging to measure accurately with a LiDAR sensor, as the reflectivity of the black paint can be below 5% (CASTRO *et al.*, 2008).

The values obtained from the sensor differed from the actual measurement by more than 30%, which is not an expected result according to the product datasheet. Therefore, hardware and software checks were conducted to identify the cause of the incorrect measurements. It was discovered that there was a problem with the output voltage of the Arduino, which powers the TFmini Plus sensor. The voltage was below the minimum recommended for the sensor's operation, as the current required by the sensor exceeds the maximum current of the Arduino UNO.

It was observed that the voltage on the Arduino's 5V output was significantly lower than expected. Photograph 12 depicts a) the voltage with all devices connected, b) the voltage with only the TFmini Plus sensor connected, and c) the voltage measured when no devices were connected to the Arduino's 5V output.

Photograph 12 - Voltages at the 5V output of the Arduino



Source: Own authorship (2023)

The measured voltage was approximately 3.458V when all devices were connected, 3.840V when only the TFmini Plus was connected, and 4.831V at the Arduino's output when no devices were connected. This test shows a voltage drop of approximately 1V when connecting the LiDAR sensor. It is understood that there is a limitation on the Arduino's ability to provide the required power to the systems, as there was also a voltage drop of approximately 0.4V when connecting the other devices.

The measured voltages are insufficient to properly power the devices. According to the datasheet, the TFmini Plus sensor requires a power supply of $5V \pm 0.5V$, and the other devices have similar operating conditions. The main reason for the voltage drop is that the LiDAR sensor consumes a maximum current of 500mA, which exceeds the 200mA limit supported by the Arduino UNO. This causes a voltage drop at the Arduino's output, as it cannot provide the necessary power to the sensor, resulting in incorrect measurements.

To solve this problem, a circuit with a voltage regulator using the LM317T was proposed. This regulator can provide an adjustable and stable voltage to the sensor. The input to the regulator is a 12V source that is already being used in the prototype to power an H-bridge that drives the two motors of the prototype.

The LM317T regulator has three terminals: input (IN), output (OUT), and adjustment (ADJ). The output voltage is determined by the equation 23:

$$V_{OUT} = V_{REF} \left(1 + \frac{R_2}{R_1} \right) + I_{ADJ} R_2 \quad (23)$$

Where:

- V_{OUT} is the voltage at the regulator's output;
- V_{REF} is the internal reference voltage of the regulator (approximately 1.25V);
- R_1 and R_2 are the resistors in the voltage divider;
- I_{ADJ} is the current at the ADJ terminal of the regulator (typically 50 μ A).

To obtain an output voltage of 5V, the values of the resistors can be calculated using the above formula. The term $I_{ADJ}R_2$, can be disregarded since the value of I_{ADJ} is very small. With $R_1 = 330 \Omega$, we have the equation 24:

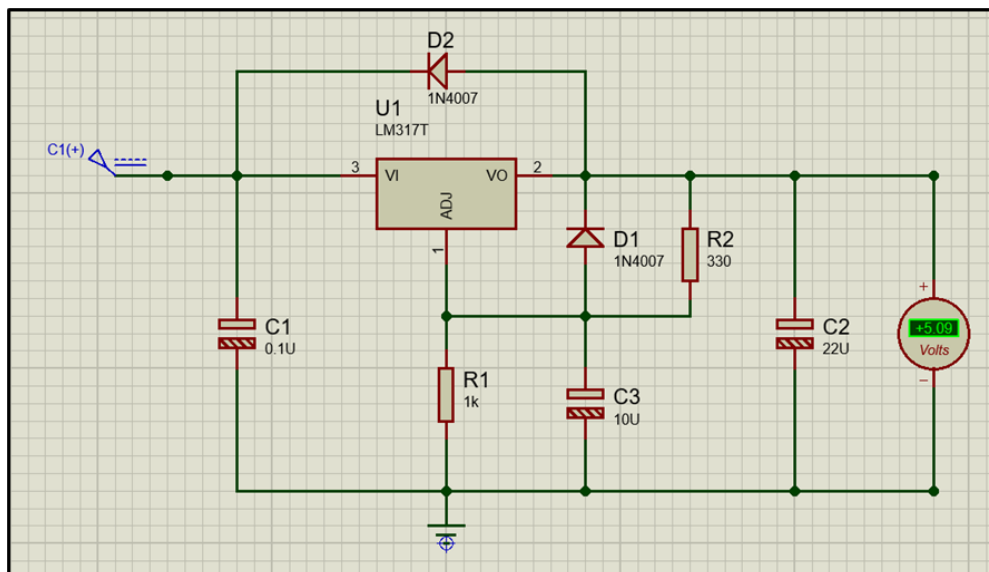
$$5 = 1,25 \left(1 + \frac{R_2}{330} \right) \quad (24)$$

Solving for R_2 , we have the value shown in equation 25:

$$R_2 = 990 \Omega \quad (25)$$

Therefore, a resistor close to this value can be used to obtain an output voltage close to 5 V. In this case, a commercial resistor of 1 k Ω was used, resulting in an output voltage of approximately 5.09 V, as shown in the diagram in figure 54.

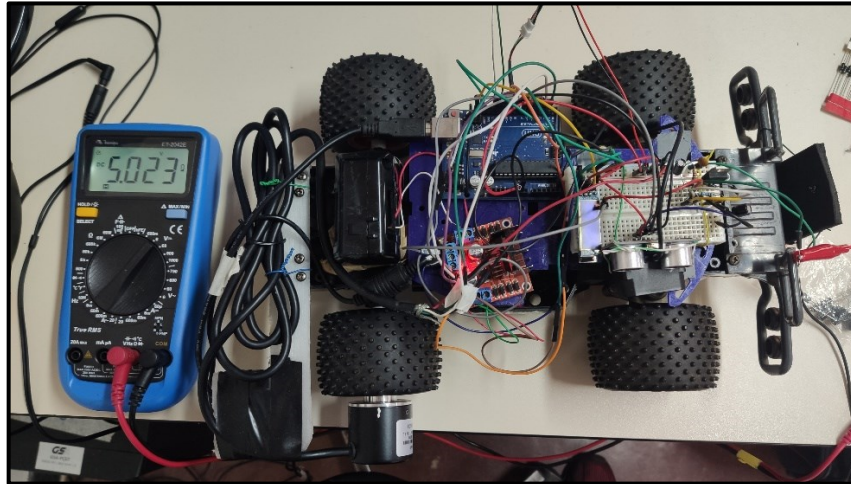
Figure 54 - Circuit with the LM317T voltage regulator



Source: Own authorship (2023)

With this circuit, it was possible to power the TFmini Plus LiDAR sensor with a suitable and independent voltage from the Arduino's output. Photograph 13 shows the voltage measured with a digital multimeter when no devices were connected to the circuit.

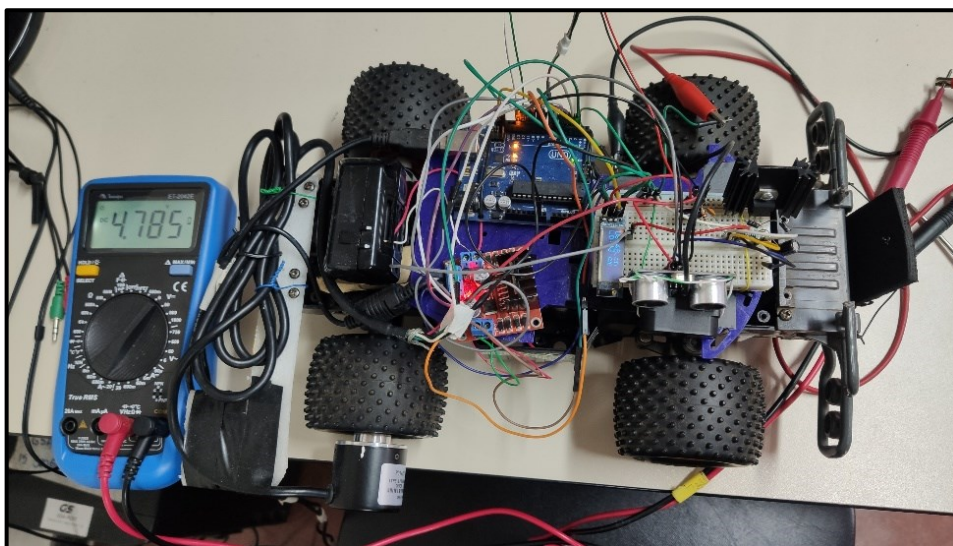
Photograph 13 - Voltage at the output of the circuit with the devices disconnected



Source: Own authorship (2023)

It can be observed that the voltage difference between the simulated and actual designed circuit was approximately 1.32%, with a voltage of 5.023 V being suitable for the circuit. After this test, the modules and sensors were connected, and photograph 14 shows the obtained values.

Photograph 14 - Voltage at the output of the circuit with the sensors and modules connected



Source: Own authorship (2023)

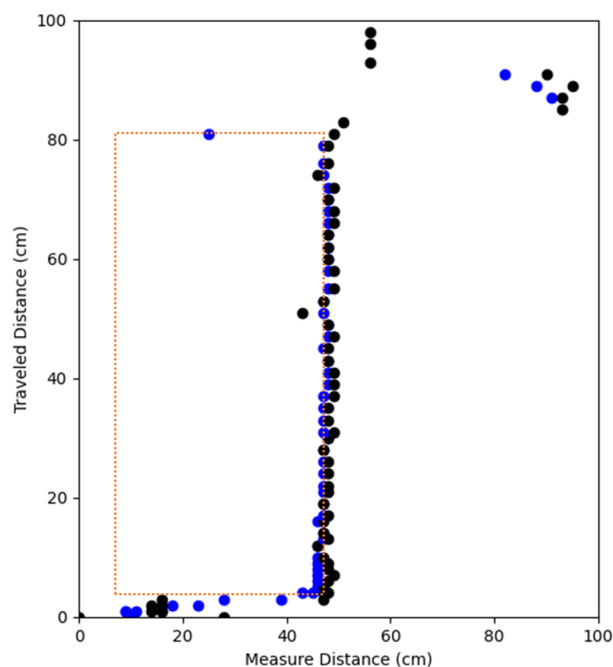
It can be observed in photograph 14 that there was a decrease in voltage from 5.023 V to 4.785 V. However, this is a normal variation due to the current consumption of the system. The voltage of 4.785 V is within the operating range of the used systems.

4.3.1.6 Sensor's comparison

The inclusion of the ultrasonic sensors served the purpose of conducting a comparative analysis with the LiDAR sensor within the context of the autonomous parking system. The tests conducted allowed for the evaluation of the effectiveness of each sensor type in obstacle detection during the parking maneuver. The obtained results provided valuable insights into the performance and limitations of each sensor, facilitating the selection of the most suitable sensor for the system.

Figure 55 depicts one of the performed tests, where the actual parking space is shown in dashed lines. The blue dots represent data from the LiDAR sensor, while the black dots represent data from the ultrasonic sensor. It can be observed from the figure that the LiDAR sensor displays fewer variations in its data. It is also interesting to note the gradient displayed at the beginning of the parking space identification, indicating that the actual boundaries of the space were not straight. At the end of the space, a dot can be seen, which represents the final boundary of the parking spot that was not detected by the ultrasonic sensor.

Figure 55 - Acquisition of a parallel parking spot performed by the ultrasonic sensor and LiDAR



Source: Own authorship (2023)

According to the conducted tests, the LiDAR sensor exhibited higher precision and consistency in the acquired data. It is also worth mentioning that ultrasonic sensors have higher precision in short-range measurements, able to measure up to approximately 4 meters, while the utilized LiDAR sensor has the capability to measure up to 12 meters.

As a way to validate the claims of superior performance of the LiDAR sensor compared to the ultrasonic sensor, comparative tests were conducted in the three types of parking spaces: parallel, perpendicular, and angled. All tests were conducted by placing the small-scale prototype a few centimeters before the beginning of the parking space. The motor of the prototype was then automatically activated, and both LiDAR and ultrasonic sensors were simultaneously used to acquire data, which was sent to the computer. The computer performed the necessary filtering and processing of the data.

4.3.1.6.1 Parallel parking spot

In this series of tests, the performance of the LiDAR and Ultrasonic sensors in measuring width and length in a parking space was evaluated, the results obtained have been summarized in the table 5. The real values for width and length were consistent at 35 cm and 70 cm, respectively.

Table 5 - Comparative tests of LiDAR and ultrasonic sensors in parallel parking

	LiDAR Width (cm)	Ultrasonic Width (cm)	LiDAR Length (cm)	Ultrasonic Length (cm)
Real Value	35	35	70	70
Test 1	34.48	31.44	76.00	1.00
Test 2	35.87	19.78	71.00	1.00
Test 3	34.13	15.22	71.00	3.00
Test 4	33.81	18.47	71.00	38.60
Test 5	31.81	15.69	67.42	36.72

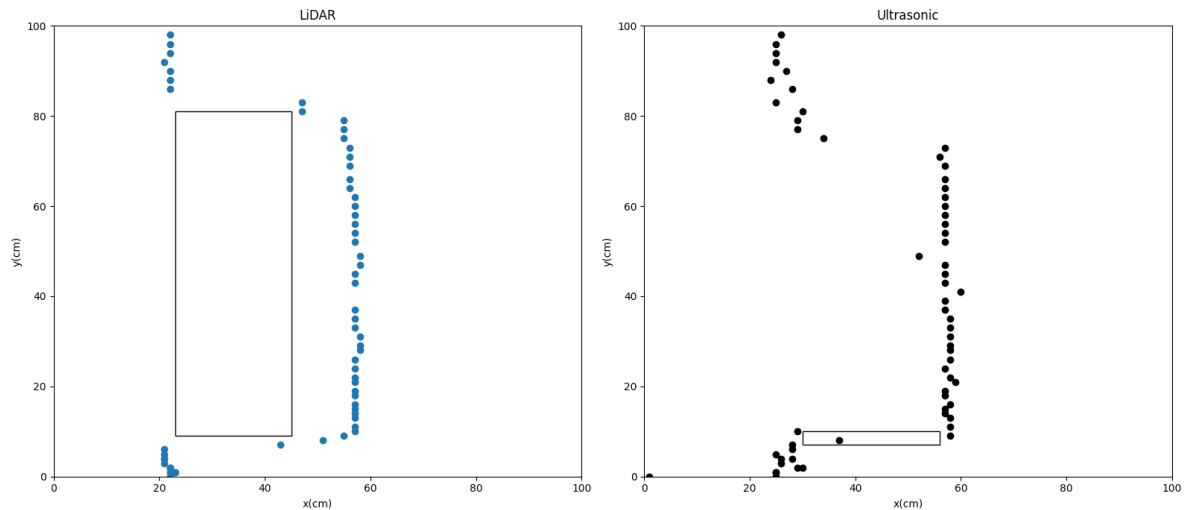
Source: Own authorship (2023)

The data revealed some interesting findings. The LiDAR measurements for width ranged from 31.44 cm to 35.87 cm, with test 1 showing the closest measurement to the real width value at 34.48 cm. On the other hand, the Ultrasonic measurements

for width varied between 15.22 cm and 31.44 cm, indicating higher variability and potentially less accuracy.

The length measurements using the LiDAR sensor showed little variation, with only the measurement of 76 cm in test 1 being an outlier. However, the measurements obtained from the ultrasonic sensor proved to be unreliable, which can be explained by the way the parking space measurement logic was developed. The calibration and design of rules and algorithms took into account the characteristics of the LiDAR sensor, known for its higher reliability and fewer outliers. Figure 56 illustrates the graph comparing test 1.

Figure 56 - Test 1 comparing LiDAR and ultrasonic sensors in a parallel parking space



Source: Own authorship (2023)

It is possible to identify the discrepancy in the data between the sensors in figure 56, focusing on the edges of the parking spaces. In these positions, there are many variations in values for the ultrasonic sensor, which appears to identify points before reaching the actual end of the parking space.

4.3.1.6.2 *Perpendicular parking spot*

Subsequently, tests were conducted to identify a perpendicular parking space, and the results can be observed in table 6. The real values for the size of the parking space are 52 cm in width and 36 cm in length.

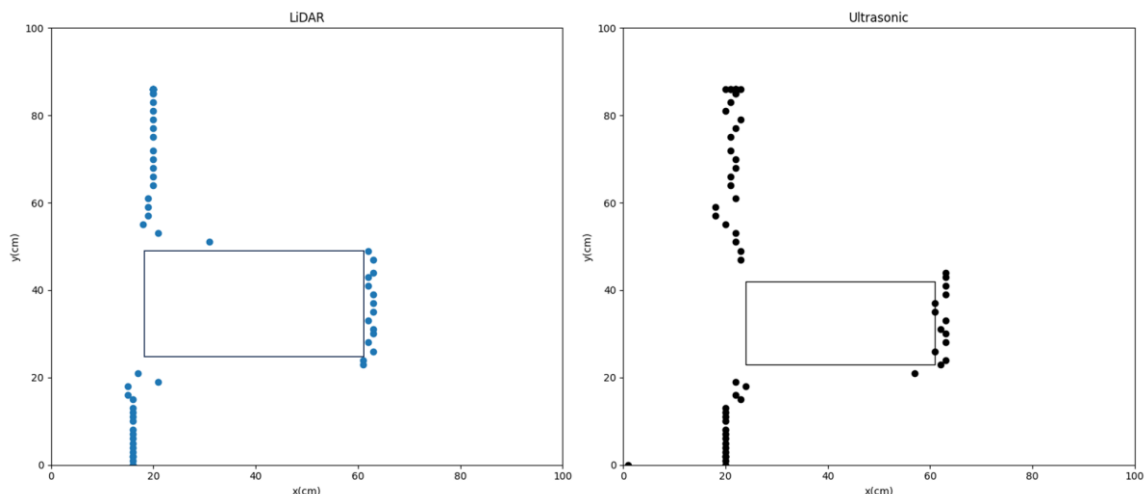
Table 6 - Comparative tests of LiDAR and ultrasonic sensors in perpendicular parking

	LiDAR Width (cm)	Ultrasonic Width (cm)	LiDAR Length (cm)	Ultrasonic Length (cm)
Real Value	52	52	36	36
Test 1	44.42	18.13	36.00	47.42
Test 2	42.53	14.91	35.00	44.01
Test 3	40.68	18.67	35.00	5.00
Test 4	42.11	5.78	33.00	25.00
Test 5	44.40	42.38	28.00	23.00

Source: Own authorship (2023)

As tests for the parallel parking space, the LiDAR sensor obtained consistent results, showing values with little variation in each test, both for width and length calculations, with the exception being seen in the length calculated in test 5, which presented a higher variation.

On the other hand, the ultrasonic sensor did not achieve consistent results, obtaining a large percentage of error for both width and length measurements in several tests. The exception was test 5, where the values were closer to the real values and also closer to the LiDAR sensor results. However, even in this case, it was possible to identify that the calculated parking space size was smaller than the real size. This is likely due to the error caused by the ultrasonic sensor's larger FOV compared to the LiDAR sensor. Figure 57 shows a side-by-side comparison of performance.

Figure 57 - Test 5 comparing LiDAR and ultrasonic sensors in a perpendicular parking space

Source: Own authorship (2023)

It is possible to identify that in Figure 69, both the LiDAR and ultrasonic sensors exhibited similar behaviors in reconstructing the parking space. However, it is noteworthy that the LiDAR sensor's points show more continuity and a reduced number of outliers. It can be understood that it is possible to identify, measure, and classify a parking space using only an ultrasonic sensor, as shown in figure 57. However, the result obtained is less precise and requires more sophisticated algorithms to attempt to compensate for the lack of consistency in the data obtained.

4.3.1.6.3 Angle parking spot

To assess the capability of LiDAR and ultrasonic sensors in identifying an angled parking space, the same comparative tests conducted previously for parallel and perpendicular parking spaces were carried out. The actual values for the width, length and angle of the angled parking space were provided as reference, measuring 34.4 cm, 62 cm and 55 degrees, respectively.

The results of these tests are of utmost importance as they will provide valuable insights into the sensors performance in a more complex and realistic parking scenario, such as the angled parking space. It is expected that the angular geometry of this parking space will pose additional challenges to the sensors, requiring higher precision in detecting and measuring the space's dimensions. The results could be seen in table 7.

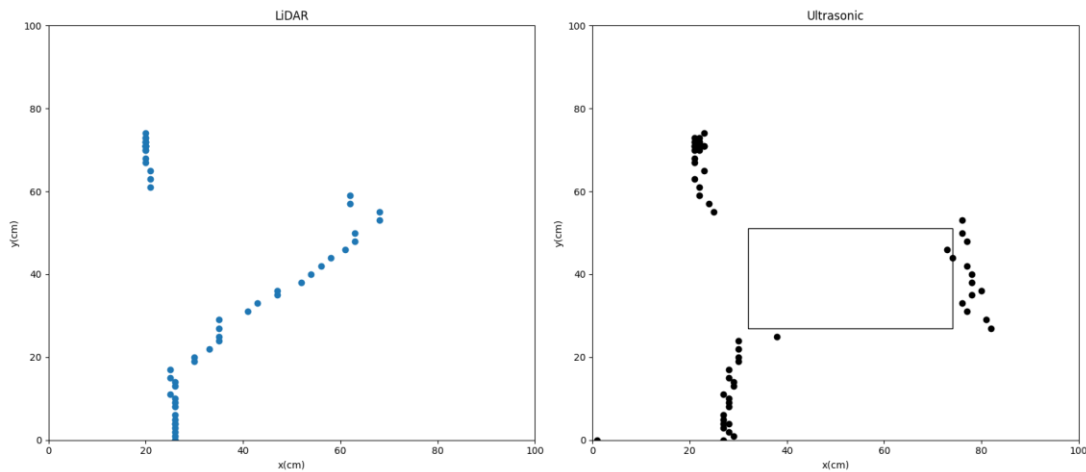
Table 7 - Comparative tests of LiDAR and ultrasonic sensors in angle parking

	LiDAR Width (cm)	Ultrasonic Width (cm)	LiDAR Length (cm)	Ultrasonic Length (cm)	LiDAR Angle (degrees)	Ultrasonic Angle (degrees)
Real Value	34.4	34.4	62	62	55	55
Test 1	25.74	10.78	51.66	28.00	47.35	70.43
Test 2	25.19	5.44	51.09	32.00	49.76	82.33
Test 3	27.05	49.03	54.56	31.00	48.72	86.82
Test 4	26.05	45.78	52.40	26.00	48.09	76.55
Test 5	25.34	49.80	50.99	28.00	48.18	84.51

Source: Own authorship (2023)

The results obtained from the tests indicate that both LiDAR and ultrasonic sensors face difficulties in accurately measuring the dimensions and angle of the angled parking space. However, the LiDAR sensor generally provides more consistent measurements, especially for the angle, where its results closely match the real value. Therefore, it is understood that improvements can be made to the results obtained by modifying the algorithm, making it more robust and accurate. On the other hand, the ultrasonic sensor's measurements display larger variations and inconsistencies, indicating its limitations in accurately capturing the parking space's dimensions and angle. Figure 58 presents the results of the sensors side by side.

Figure 58 - Test 5 comparing LiDAR and ultrasonic sensors in a perpendicular parking space



Source: Own authorship (2023)

It is possible to observe that the LiDAR sensor exhibits a pattern resembling an angled parking space in its plotted data, while the ultrasonic sensor shows data without any reference, obtaining erratic values.

4.3.1.6.4 Tests interpretation

With the tests, it was possible to understand the actual difference between the results obtained by the one-dimensional LiDAR sensor and the ultrasonic sensor. It was demonstrated that the LiDAR sensor provides greater precision and reliability in the data, while the ultrasonic sensor produced erratic measurements.

This behavior may be due to the nature of the ultrasonic sensor, which relies on sound waves to measure distances. Ultrasonic waves can bounce off surfaces, causing reflections and inaccuracies, especially in situations where the surface is irregular or has multiple reflective points, such as the edges of the parking space.

On the other hand, the LiDAR sensor uses laser beams to measure distances and is known for its ability to provide more accurate and reliable measurements, even in complex environments. It can better distinguish individual objects and surfaces, making it more suitable for precise measurements of the parking space boundaries.

As a result, the data from the ultrasonic sensor may exhibit more variations and inaccuracies, while the LiDAR sensor provides more consistent and trustworthy measurements, particularly at the edges of the parking space. This highlights the advantage of using LiDAR technology in such applications, where precision and reliability are crucial for parking and navigation systems.

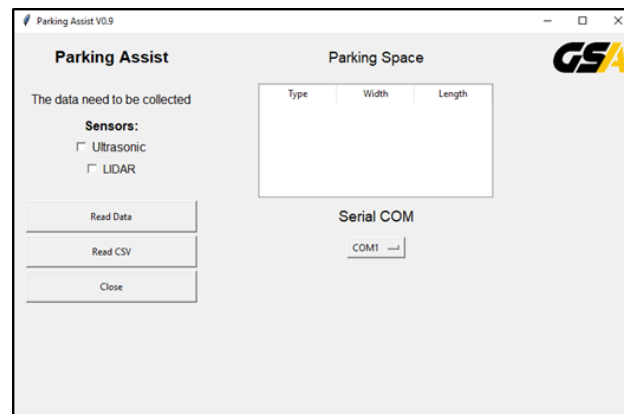
4.3.2 Software

This section presents the tests, results, problems, and potential improvements related to the Python software developed for an autonomous parking system. It highlights the objective of the software, discusses the outcomes of the tests conducted, and analyzes encountered issues. Additionally, it proposes possible enhancements to optimize the system's performance.

4.3.2.1 Interface

On the initial screen of the interface, it is possible to select the sensor to be used for data acquisition: ultrasound, LiDAR, or both for result comparison. Initially, the Serial COM needs to be defined, which represents the serial port through which the microcontroller sends data to the program. Then, you can press the "Read Data" button to acquire the data. Another option is to open a pre-recorded CSV file by pressing the "Read CSV" button. The initial screen is shown in figure 59.

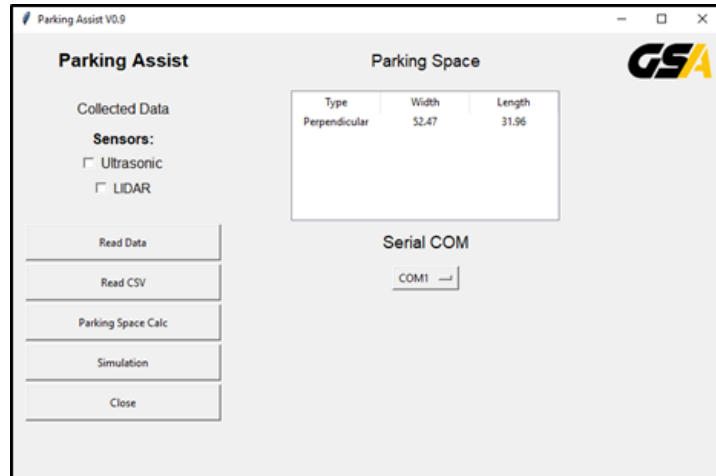
Figure 59 - Parking assistant program home screen



Source: Own authorship (2023)

After collecting the data, it is possible to calculate the size of the detected space by pressing the "Parking Space Calc" button. The value is displayed in the Parking Space table along with the type, width, and length, as shown in figure 60. If the data is read using the "Read Data" option, the user is given the option to save the data in CSV format.

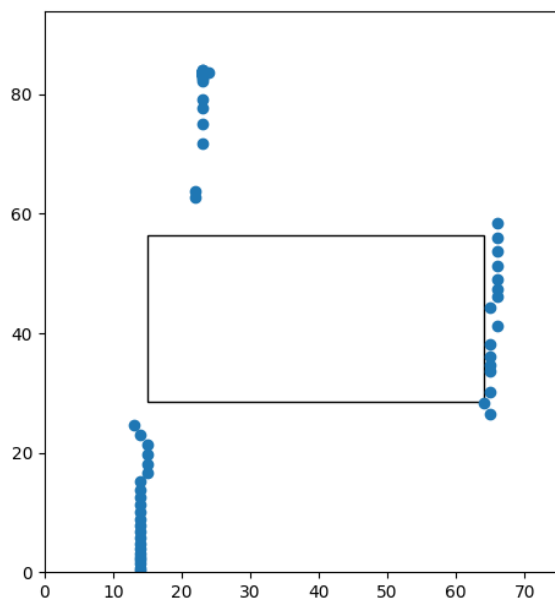
Figure 60 - Initial screen of the parking assistant program



Source: Own authorship (2023)

To demonstrate the size of the space, you can press the "Simulation" button. Afterward, a graph is plotted with the acquired points, and the parking space is shown with a rectangle, as depicted in figure 61.

Figure 61 - Detected parking space after pressing the simulation button



Source: Own authorship (2023)

If the first option chosen is "Read Data," at the end of the data reception process, the "Save" option is displayed. This option saves the received data to a CSV file within the "Save" folder located in the program's project directory.

4.3.2.2 Data reception

In order to validate the data reception process, tests were conducted to analyze the integrity of the data received through the serial connection. A small-scale prototype was used as the data emitter for these tests. Errors were identified in the serial port opening and data reading, particularly with the first data received by the software. To address these errors, error handling using try-except blocks was implemented. This ensures that the software does not crash when such errors occur.

To minimize these errors, a function was implemented to clear the serial port buffer. However, it was observed that these errors are stochastic, meaning they occur without a concrete pattern. After implementing the error handling function, the impact of these errors on the software and data collection is minimal. Typically, only one or two data points at the beginning of the data series are lost. Considering the sampling frequency of 20 Hz of the prototype's sensors, it is understood that these lost data points have negligible impact on future data analysis.

4.3.2.3 Parking spot detection

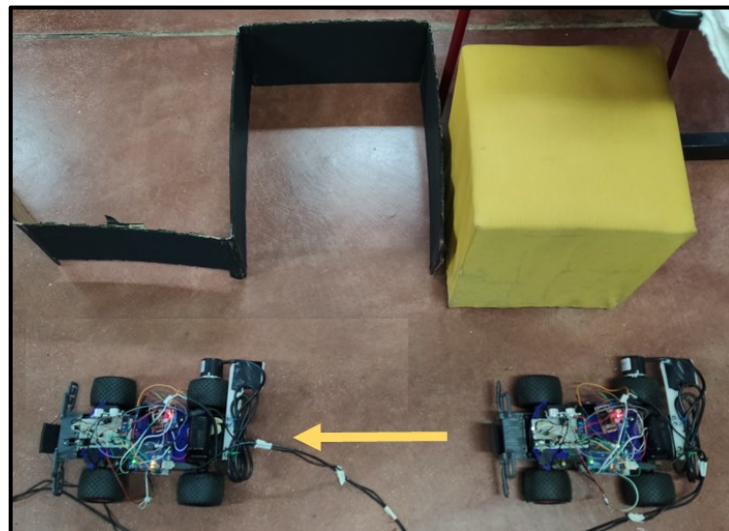
For the detection of parallel and perpendicular parking spaces, no significant issues were encountered as they exhibit a sharp change in the sensor data, making the detection algorithm reliable. However, when it comes to the detection of oblique parking spaces, problems arose due to the small variation in distance registered by the sensors, especially at the beginning of the space. As a result, the code captures only a portion of the values corresponding to the parking space. There is a need to improve the detection logic for this type of space to properly segregate the detected values belonging to the space, thus achieving a more robust system.

4.3.2.4 Parking spot measurement and classification

To validate the system, it was decided to perform five tests for each type of parking space. The tests were conducted on the constructed model of parallel, perpendicular, and oblique parking spaces. The small-scale prototype was positioned

just before the beginning of each space, as shown in **Error! Reference source not found.** Data was then acquired while the model moved forward until it reached a position just after the space, as also depicted in photograph 15. A screenshot of the synchronous plot, i.e., the data being plotted simultaneously with the test, was then taken. The "simulation" button in the software was pressed to open the plot with the drawn parking space and the calculated size of the space.

Photograph 15 - Representation of the prototype's position at a small scale at the beginning and end of the tests



Source: Own authorship (2023)

4.3.2.4.1 Parallel parking spot

The values obtained from the classification and measurement tests in the parallel parking space are shown in table 8, along with the actual measured value using a tape measure.

Table 8 - Tests of classification and measurement of parallel parking spaces

	Type	Width (cm)	Length (cm)
Real Value	Parallel	35	70
Test 1	Parallel	38.18	70.00
Test 2	Parallel	38.33	73.00
Test 3	Parallel	36.75	69.00
Test 4	Parallel	39.50	76.00
Test 5	Parallel	37.59	71.00

Source: Own authorship (2023)

It can be observed that in all tests, the parking space was correctly classified as parallel, and the calculated width values showed an average variation of 9.09%, while the length values showed an average variation of 1.43%. Most of the results deviate towards larger values than the actual measurements.

4.3.2.4.2 Perpendicular parking spot

Next, tests were conducted with the perpendicular parking space, and the results are shown in table 9.

Table 9 - Tests of classification and measurement of perpendicular parking spaces

	Type	Width (cm)	Length (cm)
Real Value	Perpendicular	52	36
Test 1	Perpendicular	55.56	33.00
Test 2	Perpendicular	54.32	38.00
Test 3	Perpendicular	53.61	35.00
Test 4	Perpendicular	56.28	34.00
Test 5	Angled	43.69	53.40

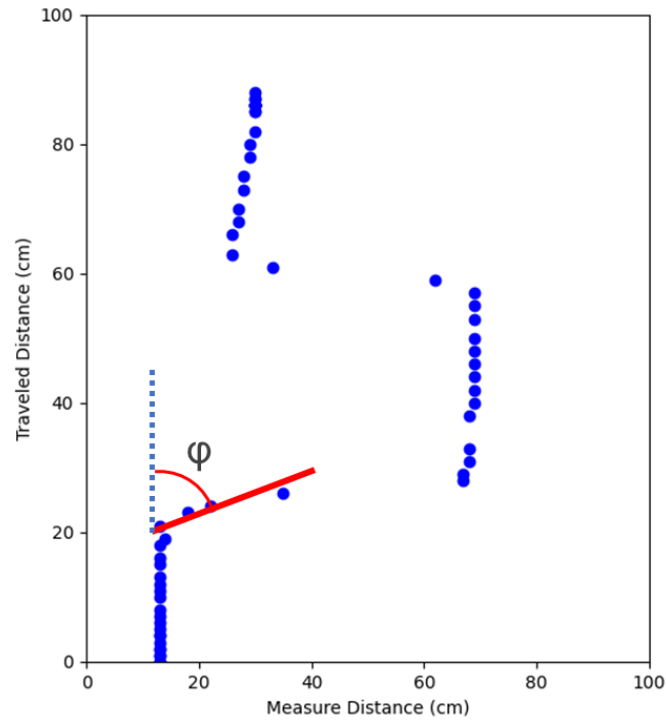
Source: Own authorship

Based on the tests conducted on the perpendicular parking space, it can be observed that the width measurements resulted in values above the actual measurement, similar to the previous tests with the parallel parking space. The average error in width measurements is calculated to be 6.85%. However, the length measurements showed a uniform distribution around the actual value, with an average error of 5.56%.

It is noteworthy that test 5 yielded an incorrect result from the algorithm, which classified the parking space as oblique and consequently calculated its size incorrectly. This error can be explained by analyzing figure 62, which presents the raw data from the prototype and the processed data from the software. It can be understood that the algorithm identified an angle smaller than 70 degrees, as indicated in figure 62, leading to a misclassification of the parking space as oblique.

This highlights the need for further improvement in the algorithm's logic for detecting oblique parking spaces to ensure accurate classification and measurement.

Figure 62 - Raw data from test 5 with the perpendicular parking space



Source: Own authorship (2023)

4.3.2.4.3 Angle parking spot

Then tests were performed with the oblique parking space, and the results are shown in table 10.

Table 10 - Tests of classification and measurement of angle parking spaces

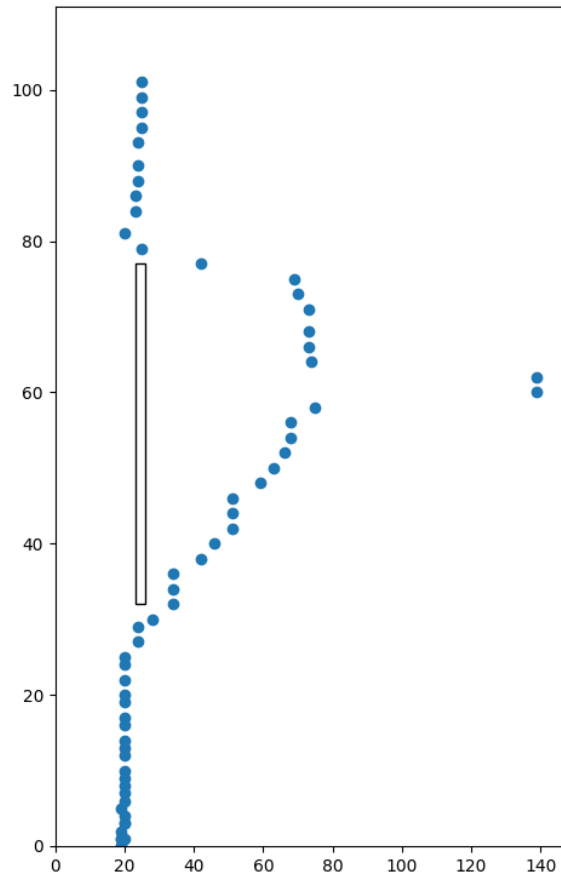
	Type	Width (cm)	Length (cm)	Angle (degrees)
Real Value	Angle	34.40	62.00	55.00
Test 1	Angle	31.95	65.44	51.20
Test 2	Angle	30.28	66.57	57.26
Test 3	Parallel	37.41	49.00	73.99
Test 4	Angle	31.70	64.66	50.65
Test 5	Angle	24.38	53.54	57.20

Source: Own authorship (2023)

It is possible to analyze that in all tests with the oblique parking space, correct classifications were obtained in tests 1, 2, 4, and 5, while only in test 3 did it obtain the classification as a parallel parking space, indicating an incorrect measurement of the

angle of the parking spot. Figure 63 presents the raw data from the test, and it can be observed that two outliers were considered for the angle calculation. This fact led to an incorrect calculation of the angle.

Figure 63 - Raw data from test 4 with the angle parking space showing two outliers

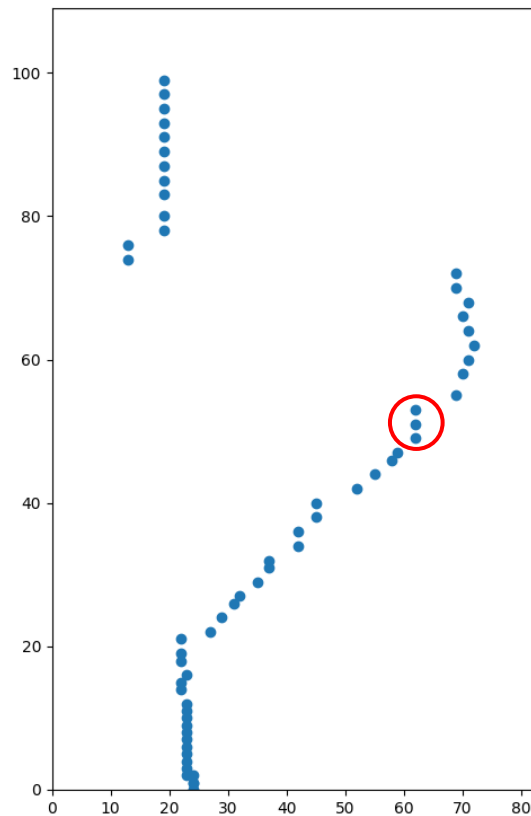


Source: Own authorship (2023)

An average error of 8.43% was found in measuring the width of the parking space, 7.37% in measuring the length, and 6.91% in measuring the angle. These measurement errors are greater than the errors obtained from the tests on the parallel and perpendicular parking spaces, which is likely due to calculations performed on the measurements that sometimes amplify errors.

This type of parking space proved to be challenging; however, an adequate level of accuracy was achieved. With further refinements to the system, it is considered possible to obtain even better data. Figure 64 presents the raw results from test 4, which had the lowest error count in the series.

Figure 64 - Raw data from test 4 with the angle parking space



Source: Own authorship (2023)

In the data from figure 64, it is possible to identify the classic shape of an oblique parking space. The presented data has been confirmed to have excellent average precision when compared to actual measurements of distance and size. There is an understanding of the need for improvement in the acquisition process to avoid repeated values, as indicated by the red circle in figure 64. This can be achieved through configuring the LiDAR sensor and adjusting the microcontroller code. By doing so, more accurate values can be obtained, thereby impacting the calculations performed.

4.4 Chapter's considerations

In chapter 4.1 was presented the results of the applied methodology for the APS, encompassing the bibliography analysis and literature review. By critically analyzing key works, the theoretical framework is established, and the significance of the study is highlighted. The Publications per Year analysis shows an increased focus on AI and motion planning in recent years. The Keywords Analysis identifies essential keywords for future research, and the Author Analysis highlights prominent

contributors in the field. These insights serve as a solid foundation for shaping the APS project and advancing knowledge in the area.

In chapter 4.2, the results of applying the proposed methodology in all stages of the APS simulation were presented. The comprehensive system simulation allowed for the analysis of the APS performance and effectiveness in different scenarios and conditions. Ultrasonic sensors were used for parking space identification; however, some detection and measurement issues were observed during the tests. Adjustments were made to improve trajectory planning and generate more natural movements. Parallel, perpendicular, and angled parking tests provided concise results, but also encountered difficulties such as measurement errors and erratic trajectory execution.

In chapter 4.3, the results of executing tests on the small-scale prototype were shown, validating the performance of hardware components like electric motors and sensors. During these tests, limitations in the ultrasonic sensors were identified, as well as the need for adjustments in the power supply for the LiDAR sensor. When comparing the sensors, it was observed that LiDAR exhibited greater precision and consistency in the acquired data. Lastly, the Python-developed software was tested and yielded satisfactory results, also identifying possible improvements to be implemented.

5 CONCLUSION

The work carried out in this master's thesis has significantly contributed to the development of an APS by addressing various aspects related to its implementation and performance evaluation. Through the utilization of simulations and the creation of a scaled-down prototype, the fundamental components and functionalities of the APS have been examined and refined. This preliminary testing phase has provided valuable insights into the system's behavior, allowing for improvements to be made before real-world implementation. By identifying and rectifying errors, as well as evaluating potential challenges and opportunities, the APS can be optimized to deliver superior performance.

One significant challenge encountered during the development process was the utilization of the LiDAR TFmini Plus sensor in conjunction with the Arduino UNO microcontroller. The high current consumption of the sensor resulted in a voltage drop at the Arduino's output, compromising its ability to power the sensor adequately. To overcome this issue, a circuit with the LM317T voltage regulator was introduced, enabling the provision of a stable and adjustable voltage supply for the sensor. This solution ensured the correct operation of the LiDAR sensor and other 5V devices within the prototype.

Additionally, a comparative analysis was conducted between ultrasonic and LiDAR sensors for measuring parking spaces. The results indicated that LiDAR sensors offer advantages over ultrasonic sensors, including improved measurement precision and a wider range. The developed APS prototype serves as a testing and validation platform, offering valuable insights into the system's performance and capabilities. While the prototype demonstrated the feasibility of detecting, measuring, and classifying parking spaces, further refinement and validation are necessary to ensure accuracy and reliability in real-world scenarios. Factors such as sensor calibration, environmental conditions, and trigonometric calculations for oblique parking spaces must be taken into account for future iterations.

In conclusion, the future advancement of the APS relies on several key areas of focus to enhance its accuracy, robustness, and adaptability in various parking scenarios. Exploring alternative sensing techniques, refining data processing algorithms, and conducting extensive testing on a larger scale prototype will contribute to a comprehensive evaluation under diverse environmental conditions and parking

space configurations. These efforts aim to develop a reliable and efficient autonomous parking solution that meets the demands of real-world applications.

In the next steps, the implementation of a path planning model based on RRT* in the small-scale prototype of the APS system will be pursued. With this trajectory planning model, efficient and safe routes for autonomous vehicle parking will be enabled by the system. The development of a control system will serve as a bridge between the path planning algorithm and the sensors and actuators of both the prototype and the vehicle. This control system will be responsible for orchestrating the execution of the planned parking trajectory by coordinating the actions of the vehicle's motors, brakes, and steering mechanism.

The control system will receive inputs from the APS sensors, including distance measurements, obstacle detection, and environmental conditions. It will process this information and generate appropriate commands to ensure precise and smooth execution of the parking maneuver. It will continuously monitor the vehicle's position and adjust its actions in real-time, responding to any changes or unexpected obstacles encountered during the parking process.

By seamlessly integrating the path planning algorithm with the sensors and actuators through the control system, the APS will be able to navigate complex parking scenarios with accuracy and safety. This integration will enable efficient coordination between the planned trajectory and the vehicle's physical. After these improvements are implemented in the prototype and extensive testing is conducted, the next step will involve the implementation of the system in a real vehicle model. This stage will require specific adaptations for the vehicle in question, as well as a comprehensive validation of the system under real operating conditions.

In summary, this master's thesis has made significant contributions to the field of autonomous parking systems. By integrating simulations, prototype development, and comprehensive testing, valuable insights have been gained regarding system performance, sensor integration, and measurement accuracy. The findings and methodologies presented in this research serve as a foundation for further advancements in autonomous parking technology, with the goal of improving urban mobility and addressing parking challenges.

REFERENCES

ALMEIDA, M. W. Z. **Carro não se constrói, compra-se: o empreendedor brasileiro na indústria automobilística entre os anos 70 e 90**. 2016. Porto Alegre, 2016. Disponível em: <<https://repositorio.pucrs.br/dspace/handle/10923/9552>>. Acesso em: 16 fev. 2023

ARDUINO. **UNO R3 | Arduino Documentation**. [S. l.], 2023. Disponível em: <<https://docs.arduino.cc/hardware/uno-rev3>>. Acesso em: 19 jun. 2023.

ATYABI, A. **Review of classical and heuristic-based navigation and path planning approaches**. [s. l.], 2013. Disponível em: <https://www.academia.edu/11873502/Review_of_classical_and_heuristic_based_navigation_and_path_planning_approaches>. Acesso em: 15 fev. 2023.

AYALA, K. J. **The 8051 microcontroller: architecture, programming, & applications**. [s. l.], p. 367, 1996. Disponível em: <https://books.google.com/books/about/The_8051_Microcontroller.html?hl=pt-BR&id=FyamZwEACAAJ>. Acesso em: 19 jun. 2023.

BANZI, M. **Getting started with arduino**. [S. l.]: O'Reilly Media, Inc, 2008. *E-book*. Disponível em: <<https://www.oreilly.com/library/view/getting-started-with/9780596155704/>>. Acesso em: 19 jun. 2023.

BENEWAKE. Datasheet: **TFmini Plus single-point ranging LiDAR**. Electronic Publication, 2023.

BENEWAKE. **TFmini Plus LiDAR module**. [S. l.], [s. d.]. Disponível em: <https://cdn.sparkfun.com/assets/2/b/0/3/8/TFmini_Plus-01-A02-Datasheet_EN.pdf>. Acesso em: 19 jun. 2023.

CARULLO, A.; *et al.* An ultrasonic sensor for distance measurement in automotive applications. **IEEE Sensors journal**, [s. l.], v. 1, n. 2, p. 143, 2001.

CASTRO, A. P. A. S.; *et al.* Medidas de refletância de cores de tintas através de análise espectral. **Ambiente Construído**, [s. l.], v. 3, n. 2, p. 69–76, 2008. Disponível em: <<https://seer.ufrgs.br/index.php/ambienteconstruido/article/view/3452>>. Acesso em: 19 jun. 2023.

CHOI, S.; *et al.* **Advanced driver-assistance systems: challenges and opportunities ahead.** [s. l.], 2016. Disponível em: <<https://www.mckinsey.com.br/industries/semiconductors/our-insights/advanced-driver-assistance-systems-challenges-and-opportunities-ahead>>. Acesso em: 20 fev. 2023.

CVIJETIC, N. **Searching for a parking spot? AI got it.** [S. l.], 2019. Disponível em: <<https://blogs.nvidia.com/blog/2019/09/11/drive-labs-ai-parking/>>. Acesso em: 16 fev. 2023.

DIETSCHE, K; KUHLGATZ, D. History of the automobile. *Em*: REIF, Konrad (org.). **Fundamentals of Automotive and Engine Technology.** Wiesbaden: Springer Fachmedien Wiesbaden, 2014. p. 1–7. *E-book.* Disponível em: <http://link.springer.com/10.1007/978-3-658-03972-1_1>. Acesso em: 16 fev. 2023.

DIGI-KEY ELECTRONICS. **Weighing the advantages and tradeoffs of encoder technologies.** [S.l.], 2020. Disponível em: <<https://www.digikey.be/nl/articles/weighing-the-advantages-and-tradeoffs-of-encoder-technologies>>. Acesso em 31 Jul. 2023.

DOUBLEHERO. **Amazon.com: DoUBLEHero Rotary Encoder 1pcs Rotary Encoder LPD3806-600bm G5-24CAB Two Phase 5-24V 600BM 400BM Pulses Incremental Optical (Color : 600BM) : Industrial & Scientific.** [S. l.], [s. d.]. Disponível em: <https://www.amazon.com/DoUBLEHero-Encoder-LPD3806-600bm-G5-24CAB-Incremental/dp/B0BCDYR73S?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&psc=1&smid=A2A09IGCZR9TX1>. Acesso em: 19 jun. 2023.

ELECTRONIC PRODUCTS. **Improving control with MEMS inertial sensors.** [S. l.], 2011. Disponível em: <<https://www.electronicproducts.com/improving-control-with-mems-inertial-sensors/>>. Acesso em: 16 fev. 2023.

FERGUSON, D; LIKHACHEV, M. **Efficiently using cost maps for planning complex maneuvers.** [s. l.], 2008. Disponível em: Acesso em: 15 fev. 2023.

FILGUEIRA, A.; *et al.* (2017). **Quantifying the influence of rain in LiDAR performance.** Measurement, 95, 143-148. ISSN 0263-2241.

FLINK, J. J. **The automobile age.** [S. l.: s. n.], 1990. *E-book.* Disponível em: <<https://quod.lib.umich.edu/cgi/t/text/text-idx?c=acls>>. Acesso em: 15 fev. 2023

G1. **Reprovações por baliza em exame de CNH representam 70% no Sul de MG.** [S. l.], 2016. Disponível em: <<https://g1.globo.com/mg/sul-de-minas/noticia/2016/04/reprovacoes-por-baliza-em-exame-de-cnh-e-sao-70-no-sul-de-minas.html>>. Acesso em: 16 fev. 2023.

GARCÍA, E. A. M. **Motion planning.** [S. l.]: IntechOpen, 2022. *E-book*. Disponível em: <<https://www.intechopen.com/books/10655>>. Acesso em: 15 fev. 2023.

GILLESPIE, T. **Fundamentals of vehicle dynamics.** [S. l.]: SAE International, 2021. *E-book*. Disponível em: <<https://books.google.com.br/books?id=LeybEAAAQBAJ>>. Acesso em: 15 fev. 2023.

GOODIN, C.; *et al.* (2019). **Predicting the Influence of Rain on LIDAR in ADAS.** *Electronics*, 8(1), 89. MDPI.

GUNTHER, T. **Triangulation.** [S. l.], 2022. Disponível em: <<https://education.nationalgeographic.org/resource/triangulation-sized/>>. Acesso em: 16 fev. 2023.

IBM. **IBM Global Parking Survey: Drivers Share Worldwide Parking Woes.** [S. l.], 2011. Disponível em: <<https://www.prnewswire.com/news-releases/ibm-global-parking-survey-drivers-share-worldwide-parking-woes-130694428.html>>. Acesso em: 16 fev. 2023.

IMAN-EINI, H.; *et al.* A modular power electronic transformer based on a cascaded H-bridge multilevel converter. **Electric Power Systems Research**, [s. l.], v. 79, n. 12, p. 1625–1637, 2009. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S0378779609001515>>. Acesso em: 16 fev. 2023.

INSTITUTO DIGITAL. **Driver Motor Ponte H L298N para Arduino - Instituto Digital.** [S. l.], [s. d.]. Disponível em: <<https://www.institutodigital.com.br/produto/driver-motor-ponte-h-l298n/>>. Acesso em: 19 jun. 2023.

INTELLIAS MOBILITY. **How Autonomous Vehicles Sensors Fusion Helps Avoid Deaths.** [S. l.], 2018. Disponível em: <<https://intellias.com/sensor-fusion-autonomous-cars-helps-avoid-deaths-road/>>. Acesso em: 16 fev. 2023.

JAZAR, R. N. **Vehicle dynamics: theory and applications**. Corrected. New York, NY: Springer, 2009. *E-book*. Disponível em: <<https://geumotorsports.files.wordpress.com/2016/08/vehicle-dynamics-theory-and-applications.pdf>>. Acesso em: 16 fev. 2023.

JHANG, J. H.; LIAN, F. L. An autonomous parking system of optimally integrating bidirectional rapidly-exploring random trees* and parking-oriented model predictive control. **IEEE Access**, [s. l.], v. 8, p. 163502–163523, 2020. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9183957>. Acesso em: 16 fev. 2023.

KIENCKE, U.; NIELSEN, LARS. **Automotive Control Systems: For Engine, Driveline, and Vehicle**. 2nd ed. ed. Berlin: Springer, 2005. *E-book*. Disponível em: <https://www.amazon.com.br/Automotive-Control-Systems-Driveline-Vehicle/dp/3540231390/ref=tmm_hrd_swatch_0?_encoding=UTF8&qid=&sr=>>. Acesso em: 16 fev. 2023.

KISSAI, M.; *et al.* Adaptive Robust Vehicle Motion Control for Future Over-Actuated Vehicles. **Machines**, [s. l.], v. 7, p. 26, 2019. Disponível em: <<https://www.mdpi.com/2075-1702/7/2/26/pdf?version=1556523658>>. Acesso em: 16 fev. 2023.

KOUBAA, A.; *et al.* **Robot Path Planning and Cooperation**. Cham: Springer International Publishing, 2018. (Studies in Computational Intelligence). v. 772 *E-book*. Disponível em: <<http://link.springer.com/10.1007/978-3-319-77042-0>>. Acesso em: 15 fev. 2023.

LAVALLE, S. M. **Planning algorithms**. [S. l.]: Cambridge University Press, 2006. *E-book*. Disponível em: <<https://www.cambridge.org/core/books/planning-algorithms/FC9CC7E67E851E40E3E45D6FE328B768>>. Acesso em: 15 fev. 2023.

LEREMY. **Formas E Métodos De Estacionamento De Carro**. [S. l.], 2017. Disponível em: <<https://www.istockphoto.com/br/vetor/formas-e-m%C3%A9todos-de-estacionamento-de-carro-gm817166910-132263807>>. Acesso em: 16 fev. 2023.

Li, N.; *et al.* A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors. **Laser & Photonics Reviews**, [s. l.], v. 16.11, n. 2100511, 2022.

LIN, S.; *et al.* A Review of Path-Planning Approaches for Multiple Mobile Robots. **Machines** 2022, Vol. 10, Page 773, [s. l.], v. 10, n. 9, 2022. Disponível em: <<https://www.mdpi.com/2075-1702/10/9/773/htm>>. Acesso em: 15 fev. 2023.

LIU, S. **Benewake TFmini vs. TFmini plus, different sensors? - ArduCopter / Copter 3.6 - ArduPilot Discourse**. [S. l.], 2019. Disponível em: <<https://discuss.ardupilot.org/t/benewake-tfmini-vs-tfmini-plus-different-sensors/37878/9>>. Acesso em: 19 jun. 2023.

LLUVIA, I.; LAZKANO, E.; ANSUATEGI, A. Active Mapping and Robot Exploration: A Survey. **Sensors** 2021, Vol. 21, Page 2445, [s. l.], v. 21, n. 7, 2021. Disponível em: <<https://www.mdpi.com/1424-8220/21/7/2445/htm>>. Acesso em: 15 fev. 2023.

MAKER HERO. **Placa Uno R3 + Cabo USB para Arduino - MakerHero**. [S. l.], 2023. Disponível em: <<https://www.makerhero.com/produto/placa-uno-r3-cabo-usb-para-arduino/>>. Acesso em: 19 jun. 2023.

MATHWORKS. **Design driving scenarios, configure sensors, and generate synthetic data**. [S. l.], 2023. Disponível em: <<https://www.mathworks.com/help/driving/ref/drivingscenariodesigner-app.html>>. Acesso em: 16 fev. 2023.

MAZIDI, M. A.; MCKINLAY, R. D.; CAUSEY, D. **PIC microcontroller and embedded systems: using Assembly and C for PIC18**. [s. l.], p. 816, 2008. Disponível em: <<http://www.staroceans.org/kernel-and-driver/PIC%20Microcontroller%20and%20Embedded%20Systems%20Using%20ASM%20%26%20C%20for%20PIC18.pdf>>. Acesso em: 19 jun. 2023.

MICHAEL. **Vehicle Dynamics: The Kinematic Bicycle Model**. [S. l.: s. n.], 2020. Disponível em: <<https://thef1clan.com/2020/09/21/vehicle-dynamics-the-kinematic-bicycle-model/>>. Acesso em: 19 jun. 2023.

MORENCY, C.; TRÉPANIER, M. Characterizing parking spaces using travel survey data. **Canada: Cirrelet**, [s. l.], 2008.

NISE, N. S. **Engenharia de Sistemas de Controle**. [S. l.: s. n.], 2017. *E-book*. Disponível em: <https://www.amazon.com.br/Engenharia-Sistemas-Controle-Norman-Nise/dp/8521634358/ref=asc_df_8521634358/?tag=googleshopp00-20&linkCode=df0&hvadid=379685646399&hvpos=&hvnetw=g&hvrand=12653030860403483300&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9102098&hvtargid=pla-811998329716&psc=1>. Acesso em: 14 fev. 2023.

NOAL, L. **Retrômobilismo#89: O mais tecnológico dos Miura, trio X8, Top Sport e X11 acabou esquecido com os importados!**. [S. l.], 2015. Disponível em:

<<https://www.conexaoautomotivabr.com/2015/06/retromobilismo89-o-mais-tecnologico-dos.html>>. Acesso em: 16 fev. 2023.

OGATA, K. **Modern Control Engineering Fifth Edition**. [S. l.: s. n.], 2009-. ISSN 0018-9286.v. 17 Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1100013>>. Acesso em: 14 fev. 2023.

OGUNTOSIN, V.; AKINDELE, A. Design of a joint angle measurement system for the rotary joint of a robotic arm using an Incremental Rotary Encoder. **Journal of Physics: Conference Series**, [s. l.], v. 1299, n. 1, 2019. Disponível em: <https://www.researchgate.net/publication/336309956_Design_of_a_joint_angle_measurement_system_for_the_rotary_joint_of_a_robotic_arm_using_an_Incremental_Rotary_Encoder>. Acesso em: 19 jun. 2023.

OLSSON, C. **Model Complexity and Coupling of Longitudinal and Lateral Control in Autonomous Vehicles Using Model Predictive Control**. 2015. [s. l.], 2015. Disponível em: <<https://www.semanticscholar.org/paper/Model-Complexity-and-Coupling-of-Longitudinal-and-Olsson/bd092b7847c6634f572b598c13abce7320274659>>. Acesso em: 19 jun. 2023.

PALLAS-ARENY, R.; WEBSTER, J. G. Sensors and signal conditioning. **John Wiley & Sons**, [s. l.], 2012.

PAROMTCHIK, I. E.; LAUGIER, C. Motion generation and control for parking an autonomous vehicle. **Proceedings - IEEE International Conference on Robotics and Automation**, [s. l.], v. 4, p. 3117–3122, 1996. Disponível em: Acesso em: 16 fev. 2023.

PIBORG. **HC-SR04 Ultrasonic Distance Sensor**. [S. l.], [s. d.]. Disponível em: <<https://www.piborg.org/sensors-1136/hc-sr04>>. Acesso em: 19 jun. 2023.

PREDKO, M. **PROGRAMMING AND CUSTOMIZING THE PIC® MICROCONTROLLER**. [S. l.: s. n.], 2008. *E-book*. Disponível em: Acesso em: 19 jun. 2023.

PURDY, K. W.; FOSTER, C. G. **Automobile - Other European developments**. *Em: BRITANNICA*. [S. l.: s. n.], 2023. Disponível em: <<https://www.britannica.com/technology/automobile/Other-European-developments>>. Acesso em: 16 fev. 2023.

RAJAMANI, R. **Vehicle Dynamics and Control**. [S. l.]: Springer Science & Business Media, 2011. *E-book*. Disponível em: <<https://books.google.com.br/books?id=eoy19aWAjBgC>>. Acesso em: 16 fev. 2023.

RODRIGUE, J.-P. World Automobile Production and Fleet, 1965-2021. **The Geography of Transport Systems**, [s. l.], 2020.

SNIDER, J. M. **Automatic Steering Methods for Autonomous Automobile Path Tracking**. [s. l.], 2009. Disponível em: <https://www.ri.cmu.edu/pub_files/2009/2/Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf>. Acesso em: 14 fev. 2023.

SOUDBAKHS, D.; ESKANDARIAN, A. **Vehicle lateral and steering control**. [S. l.]: Springer London, 2012. v. 1–2 *E-book*. Disponível em: <https://link.springer.com/referenceworkentry/10.1007/978-0-85729-085-4_10>. Acesso em: 14 fev. 2023.

STACK OVERFLOW. **Tkinter Python GUI frame positioning**. Disponível em: <<https://stackoverflow.com/questions/68294777/tkinter-python-gui-frame-positioning>>. Acesso em: 25 set. 2023

SUH, J.; OH, S. A cost-aware path planning algorithm for mobile robots. **IEEE International Conference on Intelligent Robots and Systems**, [s. l.], 2012. Disponível em: <<https://ieeexplore.ieee.org/document/6386237>>. Acesso em: 15 fev. 2023.

SYNOPSYS. **WHAT IS ADAS?** [S. l.], 2023. Disponível em: <<https://www.synopsys.com/automotive/what-is-adas.html>>. Acesso em: 15 fev. 2023.

SYNOPSYS. **What is LiDAR and How Does it Work?**. [S. l.], [s. d.]. Disponível em: <<https://www.synopsys.com/glossary/what-is-lidar.html>>. Acesso em: 16 fev. 2023.

TEXA DO BRASIL. **Adas – soluções para sistemas de assistência a condução**. [S. l.: s. n.], [s. d.]. Disponível em: <<https://www.texabrasil.com.br/produtos/radar-camera-calibration-kit>>. Acesso em: 16 fev. 2023. Acesso em: 16 fev. 2023

THEERS, M.; SINGH, M. **Kinematic Bicycle Model — Algorithms for Automated Driving**. [S. l.: s. n.], 2023. Disponível em: <<https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/BicycleModel.html>>. Acesso em: 15 fev. 2023.

ULRICH, K. **Há 125 anos Carl Benz solicitava patente do primeiro automóvel.** [S. l.: s. n.], 2011. Disponível em: <<https://www.dw.com/pt-br/há-125-anos-carl-benz-solicitava-patente-do-primeiro-automóvel/a-14799147>>. Acesso em: 15 fev. 2023.

VAZQUEZ, M. M. **Radar For Automotive: Why Do We Need Radar?**. [S. l.], 2022. Disponível em: <<https://semiengineering.com/radar-for-automotive-why-do-we-need-radar/>>. Acesso em: 16 fev. 2023.

WASLANDER, S. **Lesson 1: Proportional-Integral-Derivative (PID) Control.** [S. l.: s. n.], 2018a. Disponível em: <<https://www.coursera.org/learn/intro-self-driving-cars/lecture/QMOMH/lesson-1-proportional-integral-derivative-pid-control>>. Acesso em: 16 fev. 2023.

WASLANDER, S. **Lesson 2: Geometric Lateral Control - Pure Pursuit.** [S. l.: s. n.], 2018b. Disponível em: <<https://www.coursera.org/learn/intro-self-driving-cars/lecture/44N7x/lesson-2-geometric-lateral-control-pure-pursuit>>. Acesso em: 16 fev. 2023.

WASLANDER, S. **Lesson 2: Longitudinal Speed Control with PID.** [S. l.: s. n.], 2018c. Disponível em: <<https://www.coursera.org/lecture/intro-self-driving-cars/lesson-2-the-kinematic-bicycle-model-Bi8yE>>. Acesso em: 16 fev. 2023.

XINYU, W.; *et al.* Bidirectional Potential Guided RRT* for Motion Planning. **IEEE Access**, [s. l.], v. 7, 2019. Disponível em: <<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8763966>>. Acesso em: 15 fev. 2023.

YOUNG, H. D.; FREEDMAN, R. A.; FORD, A. L. **University Physics with Modern Physics.** [S. l.]: Pearson Education, 2012. *E-book*. Disponível em: <<https://books.google.com.br/books?id=5fgsAAAAQBAJ>>. Acesso em: 15 fev. 2023.

ZIEBINSKI, A.; *et al.* Review of advanced driver assistance systems (ADAS). 2017. **Anais [...]**. [S. l.: s. n.], 2017. p. 120002.