FEDERAL UNIVERSITY OF SANTA CATARINA
JOINVILLE TECHNOLOGICAL CENTER
AUTOMOTIVE ENGINEERING COURSE

MATHEUS DE MORAES MACAGNAN

CALIBRATION OF A NUMERICAL MODEL FOR LONGITUDINAL DYNAMICS OF A TRANSIT BUS USING CARLA SIMULATOR

Joinville
2024

MATHEUS DE MORAES MACAGNAN

CALIBRATION OF A NUMERICAL MODEL FOR LONGITUDINAL DYNAMICS OF A TRANSIT BUS USING CARLA SIMULATOR

Work presented as a requirement for obtaining a Bachelor's degree in Automotive Engineering at the Joinville Technological Center of the Federal University of Santa Catarina.

Advisor: Dr. Sérgio J. Idehara

Co-Advisor: Dr. Joshua A. Bittle

Joinville
2024

MATHEUS DE MORAES MACAGNAN

CALIBRATION OF A NUMERICAL MODEL FOR LONGITUDINAL DYNAMICS OF A TRANSIT BUS USING CARLA SIMULATOR

This Graduation Thesis was deemed suitable for obtaining the Bachelor's degree in Automotive Engineering at the Joinville Technological Center of the Federal University of Santa Catarina.

Joinville (SC), November 26, 2024.

Examining Committee:

____

Dr. Sérgio J. Idehara
Advisor/President

____

Dr. Lucas Weihmann
Member
Federal University of Santa Catarina

____

Dr. Marcos A. Rabelo
Member
Federal University of Santa Catarina

This work is dedicated to my father, Edison, my mother, Maristela, and my brother, Arthur.

ACKNOWLEDGMENT

I want to thank my parents, Edison and Maristela, for all the love and care they have given me throughout my life, for their teachings, for being my role models, and for supporting and taking care of me throughout my graduation, allowing me to dedicate myself fully to my studies and projects. Not everyone has the supportive family that I have, and I am grateful to all of you. I also want to thank my brother, Arthur, for giving me the privilege of being the older brother and for being such a great companion in life. I am grateful to my advisor, Sérgio, not only for guiding me in this project but also for all the knowledge imparted during my graduation. I would also like to thank the Federal University of Santa Catarina for providing me with the opportunity to work towards becoming an engineer. I would like to thank Professor Bittle for all the support in this work and for the chance to be part of a great environment with many opportunities to learn. I also thank the University of Alabama for providing the structure and resources for this project. Lastly, I want to thank all my friends for their help during my graduation and for making the journey easier. I’ll always remember the good moments and laughs we shared along the way.

ABSTRACT

With the rise and increasing use of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving Systems (ADS) on public roads—where, in 2020 alone, 3.5 million passenger cars worldwide were sold with Level 2 ADAS capabilities—the safety of those involved has become a critical issue, particularly in relation to testing and validating systems still under development, such as Level Four and Five autonomous vehicles, which do not require a driver. In these cases, testing in a virtual environment offers the advantage of ensuring safety, along with time and cost savings, while enabling comprehensive tests that may not be feasible in the real world. Given this, it is essential that the virtual environment accurately reflects real-world conditions. This correspondence ensures that the behavior of the vehicle and its interactions with the surroundings in the simulation match what would occur. By accurately replicating road conditions, obstacles, and sensor data (such as LIDAR and RADAR), it becomes possible to test the performance and safety of these systems in scenarios that are too risky or complex to perform in real life. The University of Alabama operates a fleet of 35 diesel buses to serve over 40,000 students, staff, and more than 100,000 residents. The university's transit system has an annual ridership exceeding 1 million passengers. The university plans to integrate both diesel and electric buses equipped with ADAS capabilities, utilizing near-market ADAS technologies developed by a third-party company. A simulation environment of bus routes, vehicle dynamics, and ADAS sensors and actuators was developed in the lab, serving as the foundation for technology validation. The present work focuses on analyzing the longitudinal dynamic behavior of a real bus and configuring the physics parameters of its virtual model using Unreal Engine 4 for simulations in the CARLA Simulator. The study presents the necessary physics parameters for calibrating the virtual model based on real test data.

Keywords: CARLA Simulator; Real-Word Test; Vehicle Dynamics

RESUMO

Com o aumento e a crescente adoção de Sistemas Avançados de Assistência ao Condutor (ADAS) e Sistemas de Condução Autônoma (ADS) em vias públicas, a segurança dos envolvidos tornou-se uma prioridade, especialmente no que diz respeito aos testes e à validação de sistemas que ainda estão em desenvolvimento, como veículos autônomos de níveis quatro e cinco, que não exigem a presença de um condutor. Nesses casos, a utilização de ambientes virtuais de teste oferece a vantagem de garantir a segurança, além de proporcionar economia de tempo e custos, permitindo a realização de testes detalhados que seriam inviáveis no mundo real. Diante desse contexto, é fundamental que o ambiente virtual reproduza de maneira precisa as condições do mundo real. Essa correspondência assegura que o comportamento do veículo e as interações com o ambiente na simulação sejam equivalentes às situações reais. Ao replicar com precisão as condições das vias, obstáculos e dados de sensores, como LIDAR e RADAR, é possível testar o desempenho e a segurança dos sistemas em cenários muito arriscados ou complexos para serem realizados fisicamente. A Universidade do Alabama possui uma frota de 35 ônibus a diesel para atender a mais de 40.000 alunos, funcionários e mais de 100.000 residentes. O sistema de transporte da universidade tem uma demanda anual superior a 1 milhão de passageiros. A universidade planeja incorporar ônibus a diesel e elétricos equipados com capacidades de ADAS, utilizando tecnologias de ADAS próximas do mercado, desenvolvidas por uma empresa terceirizada. Um ambiente de simulação de rotas de ônibus, dinâmica veicular e sensores e atuadores de ADAS foi desenvolvido em laboratório, servindo como base para a validação dessas tecnologias. O presente trabalho se concentra na análise do comportamento dinâmico longitudinal de um ônibus real e na configuração dos parâmetros físicos de seu modelo virtual usando o Unreal Engine 4 para simulações no CARLA Simulator. O estudo apresenta os parâmetros físicos necessários para calibrar o modelo virtual com base em dados de testes reais.

Palavras-chave: CARLA Simulator; Dinâmica Veicular; Teste em Mundo Real.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

4WD	Four-Wheel Drive
ADAS	Advanced Driver Assistance Systems
API	Application Programming Interface
AV	Autonomous Vehicle
CD	Drag Coefficient
CSV	Comma-Separated Values
ECU	Electronic Control Unit
EPA	Environmental Protection Agency
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
OBD II	On-Board Diagnostics II
RADAR	Radio Detection and Ranging
RMSE	Root Mean Square Error
ROS	Robot Operating System
RPM	Revolutions Per Minute
SIL	Software-in-the-Loop
UE4	Unreal Engine 4
VGT	Variable Geometry Turbocharger

SUMMARY

1. INTRODUCTION

Vehicles were once considered the realm of mechanical engineers. However, the unprecedented advancements in automobiles and information technology have transformed the traditional vehicle from an old-fashioned source of commute into a full-scale, smart, and infotainment-rich computing and commuting machine on the move (HUSSAIN; ZEADALLY, 2019).

For Li and Shi (2022), the development of Advanced Driver Assistance Systems (ADAS) technology is a direct consequence of advancements in the automotive industry. These autonomous driving features have the potential to significantly transform transportation by reducing risky driver behavior, alleviating traffic congestion, lowering carbon emissions and transportation costs, while also improving road safety, increasing independence for seniors and individuals with disabilities, and enhancing human productivity.

However, the development of autonomous driving technology presents one of the most complex engineering challenges of our time. Autonomous Vehicles (AV) are expected to navigate arbitrarily complex scenarios with greater reliability than human drivers, relying on both offline and online information processed by the vehicle (GÓMEZ-HUÉLAMO et al., 2021).

According to Nenseth, Ciccone, and Kristensen (2019), the situation for transit buses is even more delicate, as most of the relevant research has focused on autonomous cars, neglecting autonomous buses, which appear to be a more environmentally friendly version of autonomous vehicles

Developing these technologies for higher levels of driving automation introduces unforeseen challenges, with many situations that cannot be predicted. According to Stević et al. (2019), the complexity of modern vehicles continues to grow in terms of algorithmic sophistication and the hardware required. Testing these solutions in the real world is risky, expensive, and time-consuming.

To address this, driving simulations allow us to verify various scenarios iteratively so that we can travel 'billions of miles,' enabling difficult tests in the real world and making it possible to diagnose and verify real-world problems before the actual implementation of autonomous functions (WON; KIM, 2022). Driving simulators provide a controlled environment for reproducing specific situations, such as risky or complex driving conditions, and enable higher levels of controllability and reproducibility in experiments. As noted by Olstam et al. (2011), simulators are often chosen for driving behavior studies because they allow strict control over the behavior of surrounding road users.

However, the limitations of driving simulators are clear. The disadvantages are limited realism regarding the behavior of surrounding vehicles and limitations in the complexity of the scenario situations, due to both the complexity of scenario programming and the programming effort required (OLSTAM et al., 2011). This difficulty in transferring simulated experience to real-world scenarios is known as the "Reality Gap." To make simulation environments effective for autonomous vehicle research, bridging this gap is essential (PATEL, 2020).

The ego vehicle also plays a crucial role in achieving realism within simulations. According to Song and Shin (2019), the ego vehicle's behavior is critical for modern driver assistance systems, as sensor outputs—such as those from cameras, Light Detection and Ranging (LIDAR), and Radio Detection and Ranging (RADAR)—are often impacted by disturbances from vehicle dynamics, resulting in degraded sensor accuracy.

To estimate this motion, dynamic analysis can be performed. The subject of 'vehicle dynamics' is concerned with the movements of the vehicle, focusing on acceleration, braking, turning, and ride (MOLA, GENERAL MOTORS INSTITUTE, 1969). This dynamic behavior is particularly important in autonomous driving simulators, as it helps reduce the Reality Gap, making simulations more reflective of real-world scenarios. From this perspective, this study proposes analyzing the dynamic behavior of a real bus to implement a realistic model in the CARLA Simulator, serving as a basis for autonomous driving simulations.

1.1. OBJECTIVES

To establish the correct dynamic behavior of a real vehicle in an autonomous driving simulator, the following objectives are proposed in this thesis.

1.1.1. Main Objective

To develop a longitudinal dynamic model of a bus using Unreal Engine 4 (UE4) for simulations in the CARLA Simulator, and to adjust and validate the results by comparing the simulated vehicle's behavior with the real one.

1.1.2. Specific Objectives

- Perform data acquisition from real-world tests with a bus on a specific route to analyze the vehicle’s dynamic behavior;
- Analyze the dynamic setup data in UE4 correlated with the conditions of the test;
- Define the input values for UE4 based on those conditions;
- Conduct tests in the CARLA Simulator to analyze the bus's longitudinal dynamic behavior;
- Compare the simulation results with the real-world data to update and validate the virtual model.

2. THEORETICAL FOUNDATION

This chapter presents the key concepts necessary for understanding the scope of the project. It starts by discussing the justification and relevance of the proposed work within vehicle modelling. The focus then shifts to the key vehicle components involved in this project, offering an overview of each vehicle system that is important for understanding this work. Finally, an overview of the simulation approach is provided, along with a brief introduction to the tools and methodologies used for validation and testing.

2.1. TOOLS AND SOFTWARES

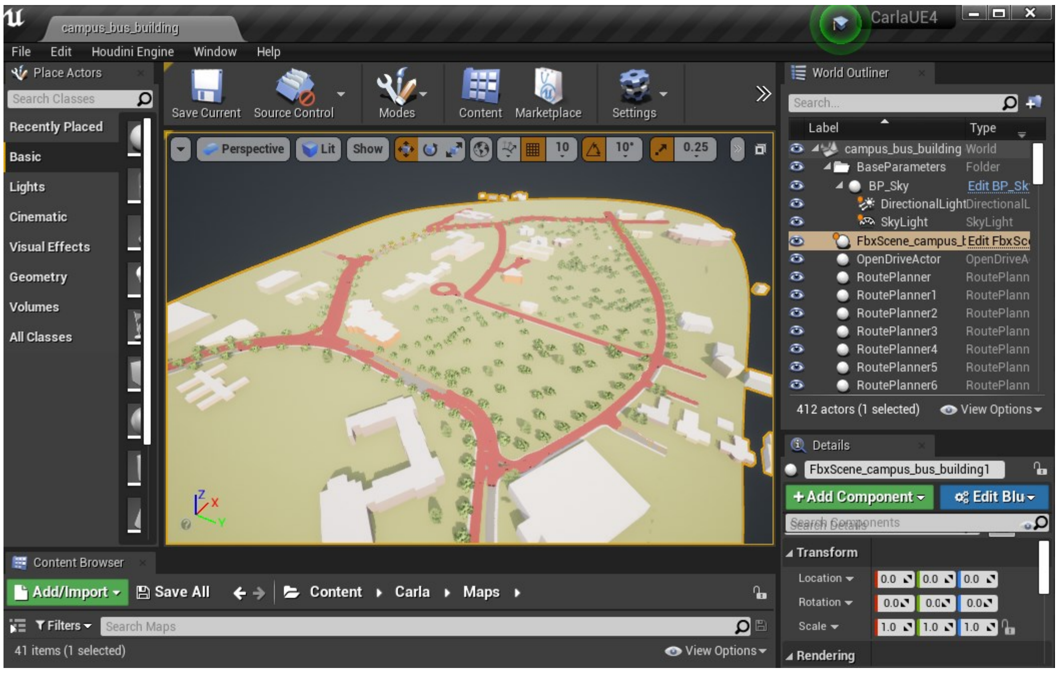
In this work, the vehicle dynamics are modeled using an open-source software called CARLA, which is based on the Python programming language. The pre- and post-processing are performed using UE4, a state-of-the-art, real-time, open-source game engine with photorealistic graphics (UE4, 2021). Using CARLA through UE4 provides the ability to integrate new actors and environments, as well as modify the physics assets of the simulation world—features that are used for the development of this work.

2.1.1 Unreal Engine

Canale Romeiro et al. (2023) explain that UE4, developed by Epic Games, is a game engine software released in 2014. It is used for game development, architectural visualization, simulations, virtual reality, augmented reality, video editing, and sound design. Epic Games continues to support and enhance the engine, providing resources and updates to its community of developers.

In this work, UE4 serves as a tool to interface between CARLA's simulation environment and user visualization. Specifically, UE4 is utilized to create an interactive interface that allows users to visualize and interact with the data generated from the CARLA simulation. Through this interface, the dynamic response of the bus, including parameters of bus components such as the engine and transmission, is animated in real-time within a 3D virtual environment. Figure 1 illustrates the UE4 environment.

Figure 1 – UE4 editor environment



Source: Author (2024).

The use of UE4 enables the simulation to include custom environments and actors, allowing the user to analyze the dynamic movements of the vehicle model within a specific environment. This helps in understanding complex mechanical responses. This approach allows developers and researchers to visually validate the simulation's accuracy, aiding in debugging and refining vehicle simulation algorithms. Furthermore, Unreal Engine's rendering capabilities make it an excellent platform for creating realistic environments and detailed vehicle models.

2.1.2 CARLA Simulator

The increasing integration of autonomous driving technologies and ADAS systems into modern vehicles calls for robust methods to test, validate, and develop such systems under realistic yet controlled conditions. To address these needs, simulators, particularly those like CARLA, have emerged as tools, simulating environments to conduct complex experiments that may otherwise be risky, costly, or ethically challenging to perform in real life (LEE et al., 2024). Stević et al. (2020) emphasized the importance of virtual simulation tools in the development and testing of AV functionalities, especially at higher levels of autonomy.

These tools enable vehicle manufacturers to test functionalities in a controlled virtual environment, which is more cost-effective and less risky compared to real-world testing. CARLA, an open-source simulator, has proven to be practical in various autonomous vehicle research contexts, allowing the simulation of urban environments with diverse conditions, such as traffic intersections, pedestrians, and varying weather patterns. The simulator also provides functionalities for reinforcement learning applications, including dynamic obstacle avoidance using deep reinforcement learning algorithms, such as double deep Q-networks, which have demonstrated improved efficiency and adaptability in complex environments (SIVYAZI, MANNAYEE, 2024).

The use of Software-In-the-Loop (SIL) techniques, in combination with simulators like CARLA, allows for more rigorous validation of perception and decision-making systems before physical prototypes are produced. The research demonstrated the integration of ADAS perception applications in Robot Operating System (ROS) environment, utilizing CARLA for input data generation and simulating sensor outputs. Such tools enable engineers to create realistic and repeatable testing scenarios, reducing the likelihood of design errors and minimizing costs during the vehicle production phase (STEVIĆ et al., 2020).

Moreover, CARLA has been employed for the testing of federated learning systems, which facilitate privacy-conscious model training using distributed data. Federated learning is particularly beneficial for AV applications due to the sensitive nature of driving data. Through simulation tools like CARLA, researchers can monitor federated learning progress, optimize model parameters, and simulate critical AV functions such as lane recognition and obstacle detection, all without risking real-life accidents (LEE et al., 2024).

Beg et al. (2024) investigated the application of the CARLA simulator for vehicle detection and collision estimation, focusing on high-risk scenarios such as intersections and roundabouts. The study highlighted the advantages of simulation in assessing safety features, reducing research costs, and minimizing risks. Their research demonstrated how simulated environments can effectively replicate challenging driving conditions, including limited visibility and unpredictable vehicle behavior, providing a safe and cost-effective method for developing AV safety systems.

He et al. (2022) proposed an end-to-end autonomous driving approach using CARLA, employing Conditional Imitation Learning and Deep Deterministic Policy

Gradient. This framework allows advanced urban driving simulations in complex environments, enhancing the learning and performance of AVs beyond human capabilities. The use of image segmentation and reinforcement learning demonstrates the potential of improving generalization in unknown environments, a crucial aspect for autonomous urban navigation.

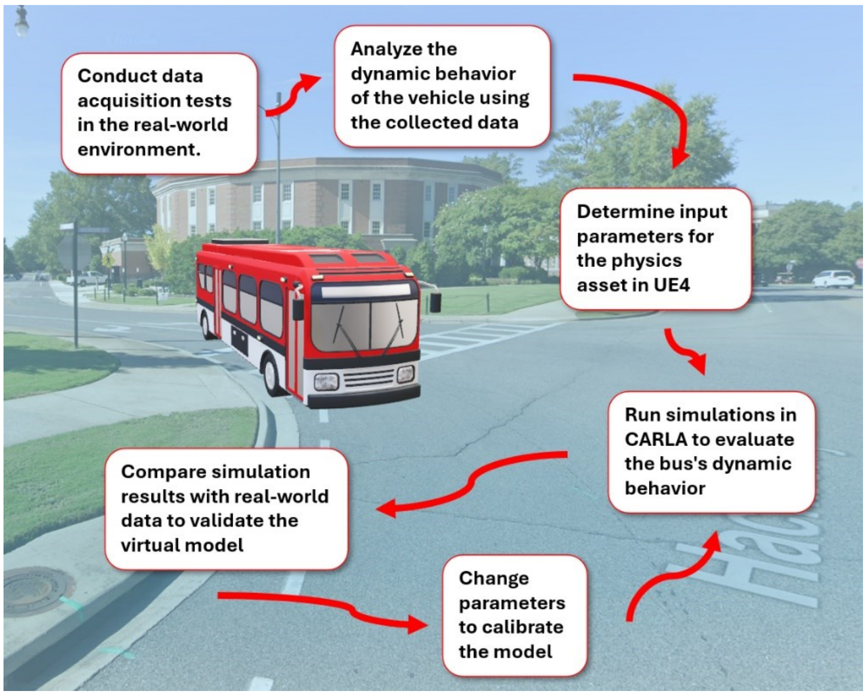
The role of simulation in autonomous vehicle research has been studied by recent advancements in Hardware-in-the-Loop (HIL) simulations and adaptive sensor technologies. Brogle et al. (2019) introduced a HIL simulation system that integrates CARLA with realistic compute hardware identical to that used on a Formula SAE autonomous vehicle, without enforcing real-time constraints. This approach helps to validate sensor outputs and vehicle dynamics by comparing simulated results to those obtained from physical testing. By using HIL, researchers can increase the level of autonomy without the risks and limitations associated with real-world tests, allowing for rapid iteration in controlled virtual environments. The implementation of ROS and the publish/subscribe software architecture enables flexibility in sensor and system integration, making it easier to adapt components during the simulation process (BROGLE et al., 2019).

The use of simulators like CARLA, integrated with Autoware, enables detailed testing of functionalities such as lane detection, forward collision warning, and lane change assistance. These capabilities are applied for validating AV perception and action modules under a variety of virtual driving conditions, including adverse weather and complex traffic layouts. The flexibility provided by these simulation environments helps to replicate unpredictable real-world situations, thereby accelerating the testing process for AV technologies while ensuring safety and reliability Stević et al. (2020)

3. METHODOLOGY

In this section, the methods used to obtain the results presented in this thesis are discussed. For this work, the dynamic model was implemented using UE4 in conjunction with CARLA. The workflow is illustrated in Figure 2.

Figure 2 – Workflow


Conduct data acquisition tests in the real-world environment.
Analyze the dynamic behavior of the vehicle using the collected data
Determine input parameters for the physics asset in UE4
Run simulations in CARLA to evaluate the bus's dynamic behavior
Change parameters to calibrate the model
Compare simulation results with real-world data to validate the virtual model

Source: Author (2024).

The bus used in this study is a specific model provided by the University of Alabama for testing purposes. The following stages involve the development of the dynamic behavior model, starting with real-world tests and leading to the validation of the results obtained from the simulations.

3.1 THE REAL-WORD DATA ACQUISITION

The first step in constructing the virtual model of the bus is obtaining data from the real vehicle. To achieve this, a dynamic test was conducted on a specific route, during which data on position, acceleration, and engine performance were collected.

These data are essential for calibrating the virtual model in the simulator, ensuring it accurately reflects the real vehicle's behavior.

For the data collection, a university bus driver participated in the process. During the test, the driver followed the same routine as in normal operations, ensuring that the driving conditions mirrored those experienced in daily use. The bus used in this study is part of the university's fleet, and the driver was already familiar with operating this vehicle. The specific aspects of the test are detailed in the following sections.

3.1.1 Bus Used for Testing

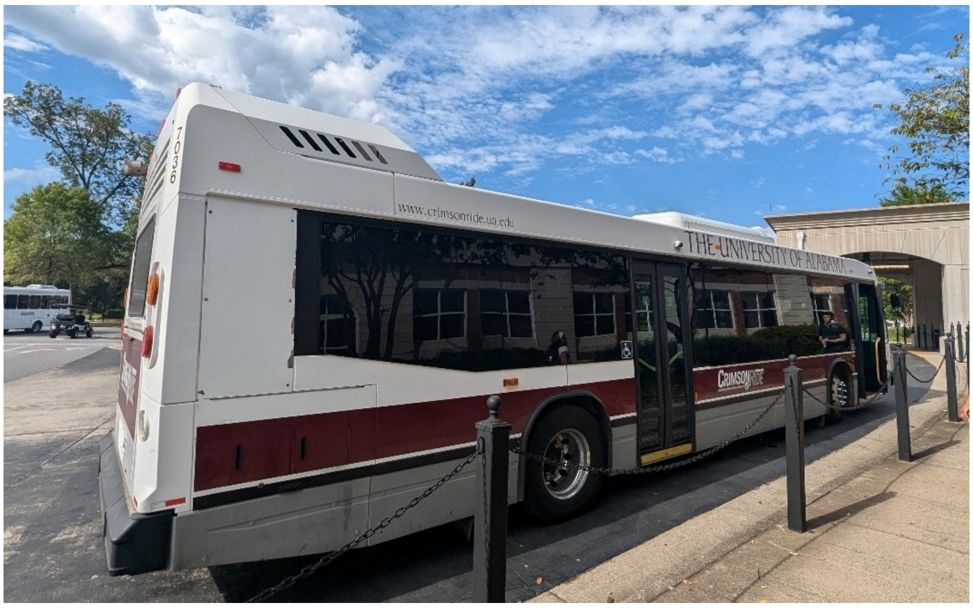
The vehicle used for the test is a Nova Bus LFS Diesel, an internal combustion bus designed for passenger transportation. The bus used for the test is the same brand and model as the buses that will receive the ADAS kit installation. Therefore, the dynamic behavior of the bus used in the tests corresponds to that of the autonomous buses that will be deployed later. Table 1 provides the specifications for this model and Figure 3 shows the bus used during the tests.

Table 1 – Bus specifications

Specification	Details
Length	12.19 m
Width	12.59 m
Height	3.20 m
Wheelbase	6.19 m
Seating capacity	Up to 41 passengers
Loading capacity	Up to 80 passengers
Structure	Stainless steel
Engine	Cummins ISL 8.9
Electric system	Volvo multiplex system
Transmission	ZF EcoLife 6 speed 6AP1400B
Front axle	ZF RL-82
Rear axle	ZF AV-132
Brake system	ABS disc brakes with traction control
Fuel tank capacity	125 gallons
Turning radius	12.45 m

Source: Adapted from Novabus (2024).

Figure 3 – University bus



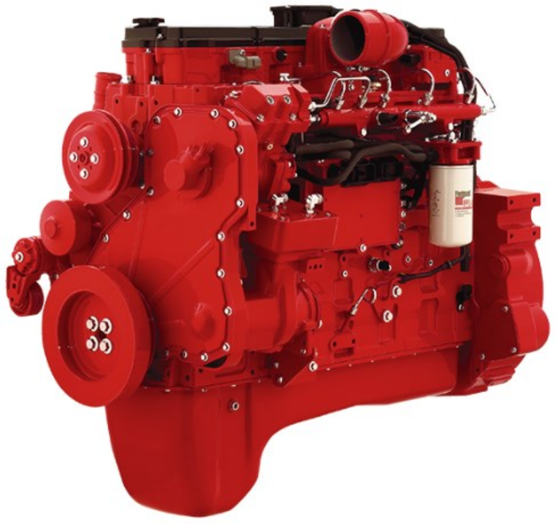
Source: Author (2024).

In addition to data sourced from the Novabus website, the university's transportation services provided details about the bus's weight and production year. The bus has a curb weight of approximately 18,000 kg and was manufactured in 2016. The subsequent sections provide an overview of its key components.

3.1.1.1 Engine

The bus is powered by a Cummins ISL 8.9 engine, weighing 800 kg. It is equipped with a Variable Geometry Turbocharger (VGT) (CUMMINS, 2024). Figure 4 illustrates the engine.

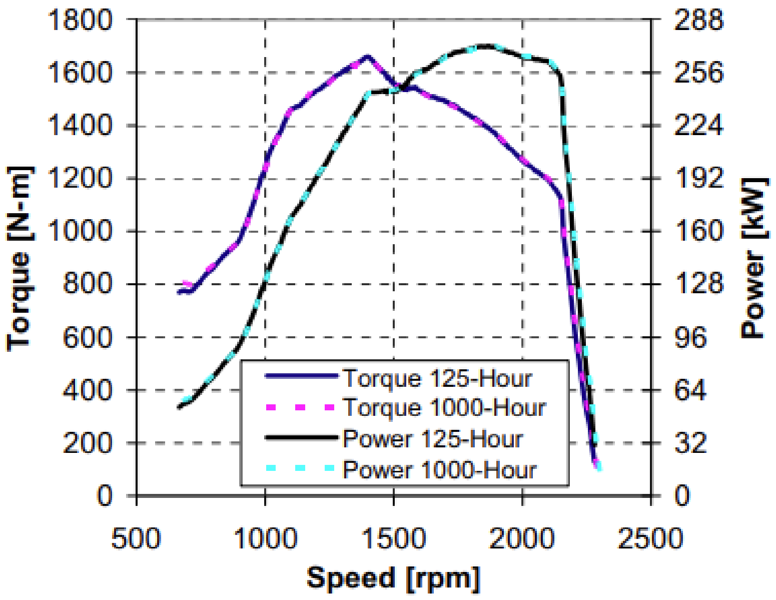
Figure 4 – Cummins ISL 8.9



Source: (CUMMINS, 2024)

For this work, the information provided by Bartoli et al. (2009) will be used. The 8.9-liter heavy-duty, on-highway diesel engine produced 268 kW using B20 fuel. This inline 6-cylinder engine has a rated speed of 2100 revolutions per minute (RPM) and a peak torque of 1708 N·m at 1400 RPM. The torque and power curves are shown in Figure 5 and will serve as a reference for building the virtual bus model in the following sections.

Figure 5 – Engine torque and power curves



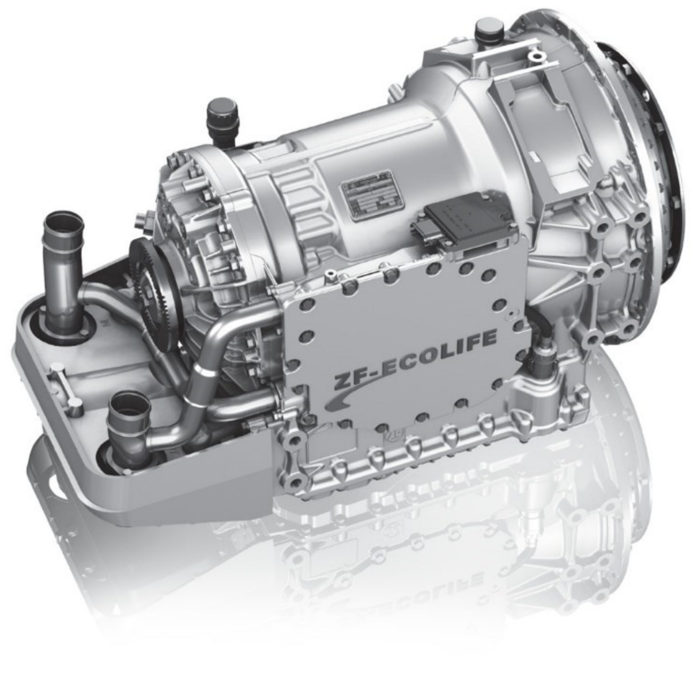
Source: (BARTOLI et al., 2009).

The figure displays the engine’s torque and power traces for the 125-hour and 1000-hour tests, where the engine performance remained essentially the same in both cases (BARTOLI et al., 2009). However, the two different time test curves are not the focus of this work.

3.1.1.2 Transmission

The bus is equipped with the first-generation ZF Ecolife, a six-speed automatic transmission introduced in 2007, specifically designed for city bus applications. It complies with the Euro 6 and the Environmental Protection Agency (EPA) emission standards in Europe and North America, respectively (ZF, 2024). Figure 6 shows the transmission.

Figure 6 – ZF Ecolife automatic transmission



Source: (ZF, 2024).

The transmission in this bus is the 6AP1400B, weighing approximately 351 kg and with an oil capacity of 38 liters. It supports vehicle weights up to 28,000 kg. The gear ratios of the transmission are relevant to this project and are presented in Table 2.

Table 2 – Transmission gear ratios

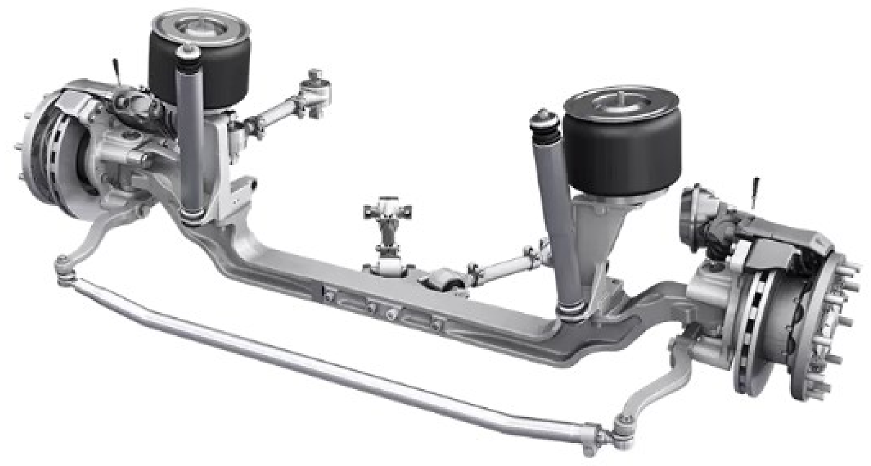
1st	2st	3st	4st	5st	6st	R
3.36	1.91	1.42	1.00	0.72	0.62	9.84

Source: (ZF, 2024).

3.1.1.3 Axles

ZF supplies complete front and rear axle systems, including wheel suspensions, shock absorbers, and brakes, to automakers worldwide (ZF, 2024). This bus is equipped with ZF axles on both the front and rear. The front axle is the ZF RL 82 A, a steering axle with pneumatic ventilated disc brakes and a wheel travel of +90 / -100 mm (Version RL 82 ET). It is a low-floor axle with a floor height of 350 mm and steering angles of up to 55°. Figure 7 shows the system.

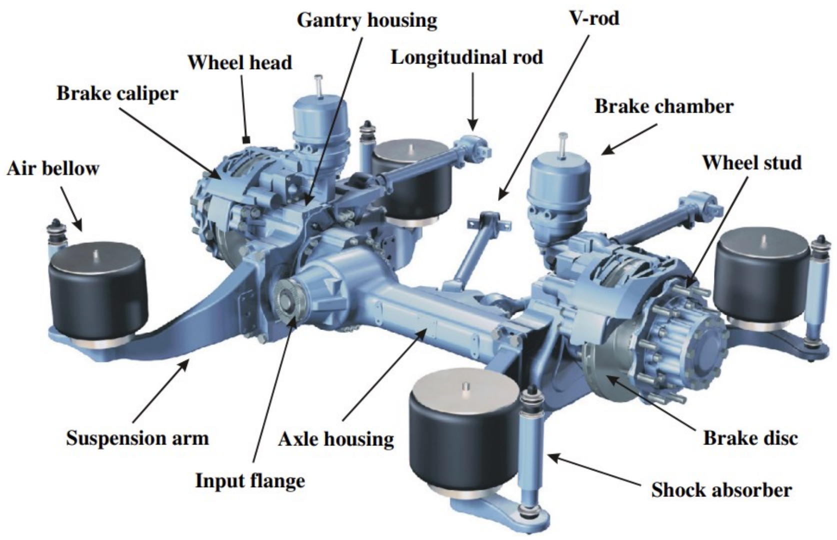
Figure 7 – Front axle RL 82 A



Source: (ZF, 2024).

The rear axle of the vehicle is a ZF AV 132 portal axle, which contributes to the bus's low-floor design. It features an open differential. Poojitganont et al. (2020) specify that the ZF AV-132 has a gear ratio of 7.38. Figure 8 illustrates the rear axle.

Figure 8 – Rear axle RL AV 132

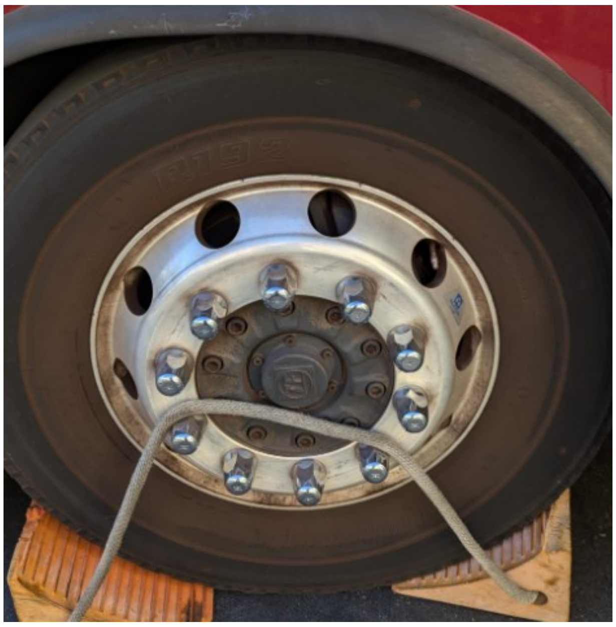


Source: (ZF, 2024).

3.1.1.4 Wheels and tires

The bus is equipped with 305/70R22.5 tires mounted on 22.5-inch Alcoa® wheels. For this project, the dimensions and weight of the tire-wheel assembly are important characteristics. The tire has a section width of 305 mm, with a height equal to 70% of its width, and a wheel diameter of 22.5 inches. Figure 9 shows the tire-wheel assembly on the bus.

Figure 9 – Bus wheel tire assemble



Source: Author (2024).

For the weight calculation, the combined weight of the tire and wheel will be considered. The tire weight is 51 kg (MICHELIN, 2024), while the wheel weight is 18 kg (ALCOA WHEELS, 2024). The Table 3 provides more detailed information on the key parameters relevant to the project.

Table 3 – Wheel-tire dimensions

Parameter	**Single wheel**	**Dual wheel**
Width (mm)	305	610
Radius (mm)	499.25	499.25
Weight (kg)	69	138

Source: Adapted from Michelin and Alcoa (2024).

3.1.2 Devices Employed in Testing

To conduct the dynamic tests on the vehicle presented, the following devices were applied.

3.1.1.1 AIM ECULog

The data logger used to capture information from the vehicle's Electronic Control Unit (ECU) is connected to the bus's On-Board Diagnostics 2 (OBD II) port and linked to a data hub, enabling the integration of a Global Positioning System (GPS) during testing. Table 4 provides the specifications.

Table 4 – Data logger specifications

Parameter	**Specifications**
External Power Supply	9 to 15 V
Internal Memory	4 GB
Removable USB Key	16 GB USB Type-C
Dimensions	61.4 x 44.7 x 27.2 mm
Weight	60 g

Source: (AIM SPORTLINE, 2024).

3.1.2.2 AIM GPS09 Pro

The GPS09 Pro is a high-precision GPS module used to capture real-time data on position, speed, and trajectory. It integrates directly with the data hub, operating at a frequency of 25 Hz with an average tolerance of less than 0.5 meters. Figure 10 shows the device.

Figure 10 – AIM GpS09 Pro

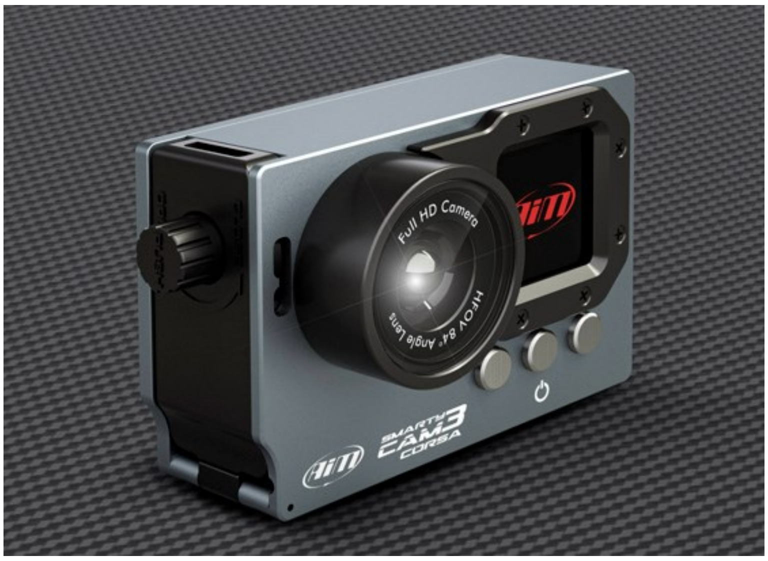
Source: (AIM SPORTLINE, 2024).

In addition to GPS functionality, the device includes a 100 Hz internal 3-axis ±5G accelerometer, 3-axis gyro, and 3-axis magnetometer. This feature was used to measure the accelerations of the bus.

3.1.2.3 AIM SmartyCam 3 Corsa

The SmartyCam 3 Corsa is a telemetry-integrated camera that records video synchronized with vehicle performance data. Developed for racing environments, it captures high-quality footage while logging key parameters such as speed, acceleration, and ECU data. This data are overlaid on the video, offering a detailed visual representation of events during tests, the Figure 11 shows the device.

Figure 11 – AIM SmartyCam 3 Corsa



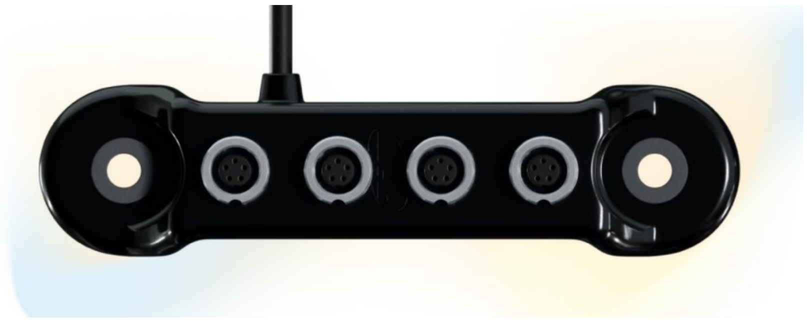
Source: (AIM SPORTLINE, 2024).

Connected via the data hub, the SmartyCam 3 Corsa integrates with the ECULog, GPS, and other sensors. The recorded video can be analyzed through RaceStudio 3, enabling a thorough review of vehicle performance.

3.1.2.4 AIM Four-Way Data Hub

The CAN connection multiplier, shown in Figure 12, is designed to streamline the integration of multiple peripherals with the datalogger. Its primary function is to act as an interface between the AIM datalogger and up to four peripheral devices, such as the GPS09 module. This setup enables the collection of essential vehicle performance data while reducing cable complexity.

Figure 12 – AIM Four-Way Data Hub



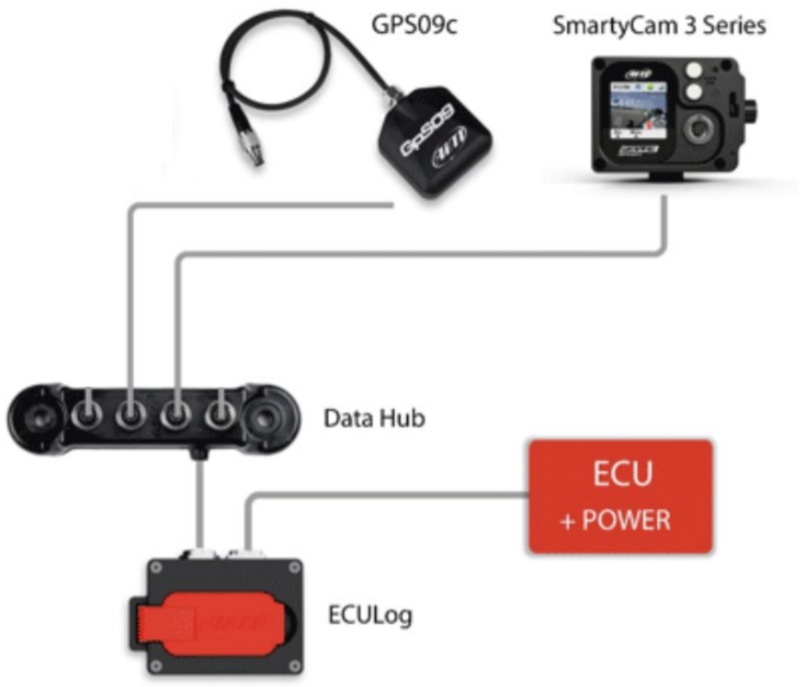
Source: (AIM SPORTLINE, 2024).

The data hub reduces cable requirements by transmitting all data through a single connection to the logger. With four additional CAN channels, it enables the creation of a networked system, ensuring efficient and organized data acquisition during testing.

3.1.2 The Setup Layout

The setup was established by connecting the GPS, IMU, and camera peripherals to data hub, which was then linked to the data logger. The connection to the ECU was made through the OBD II port of the bus's ECU, directly to the data logger. Figure 13 illustrates this layout.

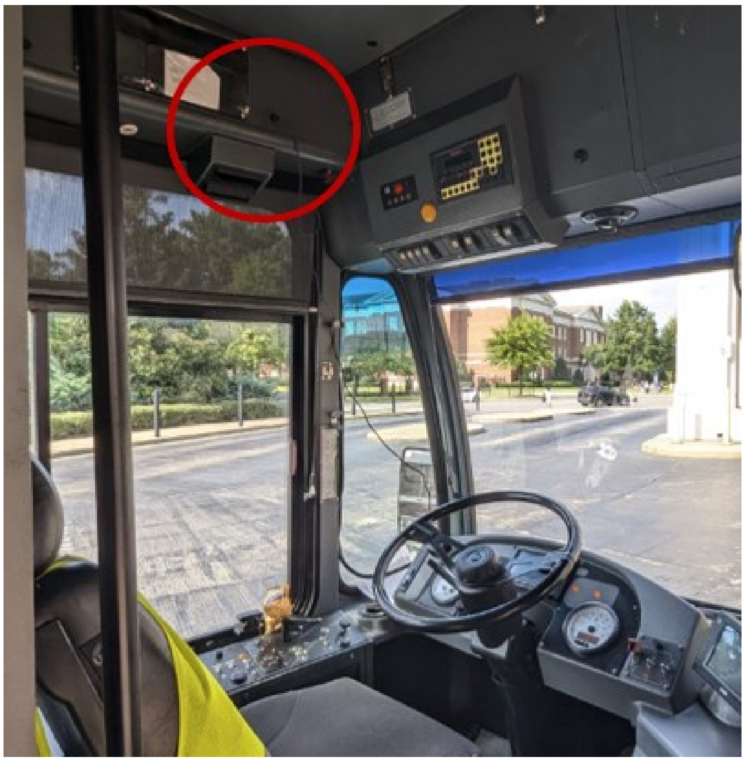
Figure 13 – Connection layout



Source: Adapted from AIM Sportsline (2024).

All devices were mounted on the vehicle's dashboard. In this specific bus, the OBD II port is located above the driver, on the left-hand side, as shown in Figure 14. The ECU connection was secured, and the other components were positioned on the dashboard accordingly.

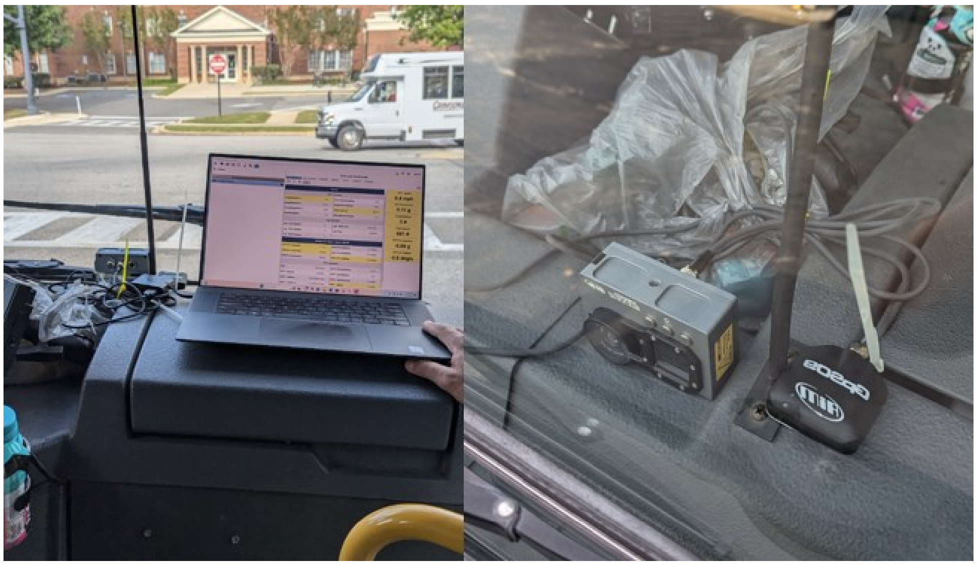
Figure 14 – OBD II connection



Source: Author (2024).

To analyze the data collected during the test, a computer was used as a dashboard, running Real Studio 3, as shown in Figure 15. The camera and GPS were mounted in the center of the dashboard, with the GPS fixed in place using a zip tie for stability.

Figure 15 – Data acquisition devices on the vehicle



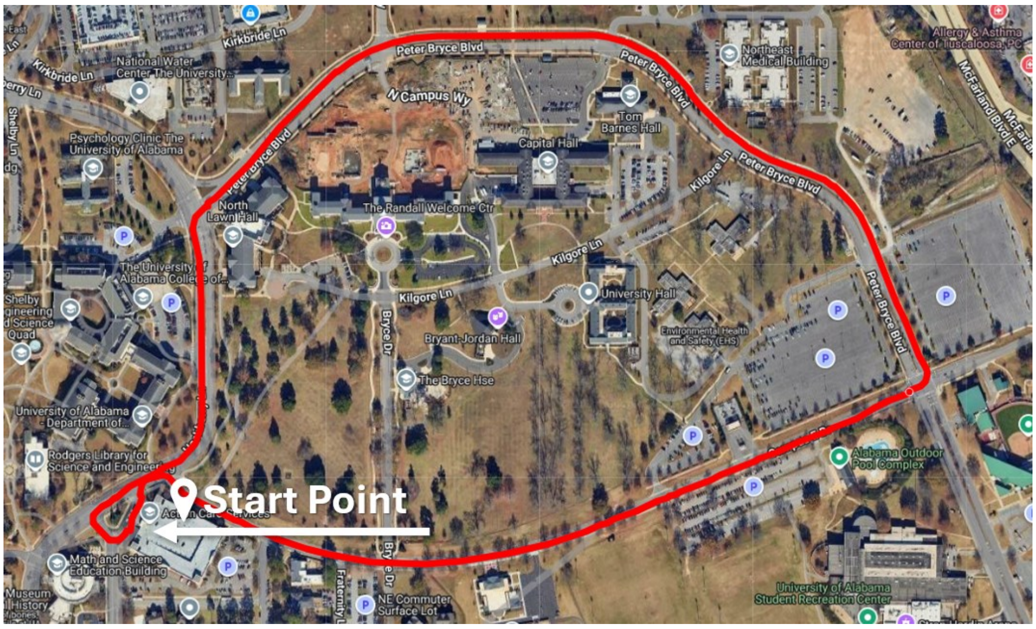
Source: Author (2024).

3.1.3 Real Test Procedure

For the test, a specific route was chosen, which is a path already used for internal transportation within the university. The route covers approximately 2.4 km. The test took place at 3:21 PM under clear, sunny conditions, with a temperature of 30°C. It was conducted in an open area with ongoing local traffic, including other vehicles and pedestrians. During the test, all traffic laws were followed, with the vehicle stopping at traffic lights, intersections, and pedestrian crossings. The driver was instructed to drive normally, as in everyday operations.

The bus used for the test was reserved specifically for this purpose, and there were no passengers on board. Therefore, the bus did not stop at the designated bus stops along the route. The route consisted of a closed loop, with the starting and ending points at the same location. Figure 16 shows the route with the starting point.

Figure 16 – Test route map


Start Point
Peter Bryce Blvd
Kirkbride Ln
National Water Center The University
N Campus Wy
Northeast Medical Building
Tom Barnes Hall
Capital Hall
Psychology Clinic The University of Alabama
North Lawn Hall
The Randall Welcome Ctr
Kilgore Ln
University Hall
Bryant-Jordan Hall
The Bryce Hse
Environmental Health and Safety (EHS)
Bryce Dr
University of Alabama Department of
Rodgers Library for Science and Engineering
Math and Science Education Building
NE Commuter Surface Lot
Alabama Outdoor Pool Complex
University of Alabama Student Recreation Center
Allergy & Asthma Center of Tuscaloosa, PC
McFarland Blvd E
Shelby Ln

Source: Author (2024).

The route consists of four intersections with traffic lights and one intersection with a stop sign. Additionally, there are vehicle entries and exits along the path due to the presence of parking areas. The route covers three streets, as detailed in Table 5.

Table 5 – Streets of the route

Street	Length (m)
Campus W Dr.	970
Peter Bryce Blvd	1135
Hackberry Ln	300

Source: Author (2024).

3.2 REAL WORLD DATA ANALYSIS SETUP

For the analysis of the data collected, AIM Race Studio software was employed. This software is widely used in motor sports and technical environments for processing and analyzing telemetry data. It allows for a detailed examination of vehicle performance metrics, such as speed, acceleration, and engine data, which are used for dynamic studies and simulations.

Considering that the data acquisition equipment, including the GPS, data logger, data hub, and camera, was provided by AIM Motorsport, the use of Race Studio 3 ensured effective integration. This setup allowed for efficient data synchronization across multiple devices. Using AIM equipment with Race Studio simplified the process by providing an interface that made it easier to correlate information from various sources, such as the ECU, GPS, and IMU.

The acquired data was processed using Race Studio, where ECU data, GPS readings, and the IMU integrated into the GPS were analyzed. Table 6 presents the parameters collected during the test.

Table 6 – Collected channels

ECU channels	GPS channels
Actual engine per T (%)	Inline Acceleration (g)
Drivers demand (%)	Lateral Acceleration (g)
Engine load (%)	Vertical Acceleration (g)
Engine RPM	PitchRate (deg/s)
EngineSpeed	RollRate (deg/s)
Throttle (%)	YawRate (deg/s)
	Speed (km/)

Source: Author (2024).

3.3 DINAMIC SETUP

This stage focuses on analyzing the parameters within UE4 for constructing the dynamic model of the bus, to gain an understanding of each parameter that the software allows us to use. The description of each parameter provided by the software is shown in the following tables. No numerical equations from the software model were used in this work. Those parameters were set with the objective of performing simulations in CARLA. In the following sections, we will examine the relevant parameters for building this model.

3.3.1 Engine setup

To configure the engine parameters in UE4, the data collected from the bus engine will be used. For the virtual model's calibration to match the real vehicle, two parameters will be set: the torque curve and the maximum RPM. Table 7 provides a detailed description of each parameter.

Table 7 – Virtual model engine setup

Parameter	Description
Torque curve	Amount of torque at a given RPM (Figure 5)
Max RPM	Maximum revolutions per minute of the engine

Source: Author (2024).

For the torque curve, the one provided by Bartoli et al. (2009), shown in Figure 5 from the Cummins ISL 8.9 engine, will be used. Key points were extracted from this curve to be input into UE4. Table 8 lists the selected points, and Figure 17 shows the torque curve as implemented in UE4.

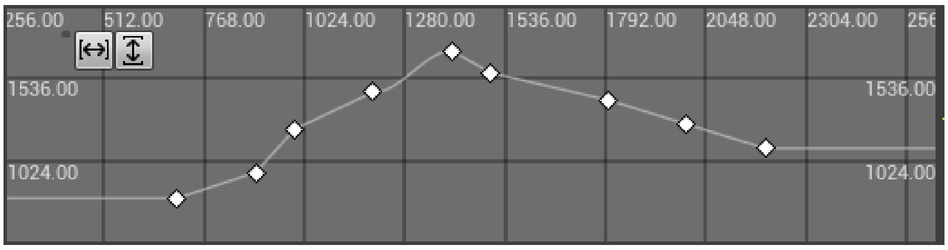
Table 8 – Torque curve data

Engine RPM	Torque (Nm)
700	790
900	950
1000	1220
1200	1450
1400	1708

Engine RPM	Torque (Nm)
1500	1570
1800	1400
2000	1250
2200	1100

Source: Adapted from Bartoli ae al. (2009).

Figure 17 – Torque curve in UE4



Source: Author (2024).

For the maximum RPM parameter, the value of 2100 RPM will be used, as specified by Cummins for this engine model (Cummins, 2024).

3.3.2 Differential setup

For the differential setup, Table 9 shows parameters available to be configured.

Table 9 – Virtual model differential setup

Parameter	Description
Differential type	Type of differential used in the vehicle.
Rear left-right split	Torque distribution between rear wheels

Source: Author (2024).

To set it up, the first step was to define the differential type. The available options include limited slip front drive, limited slip rear drive, limited slip Four-Wheel Drive (4WD), open rear drive, open front drive, and open 4WD. Since the vehicle in question is a rear-wheel drive with an open differential, the open rear drive option was selected. The rear left and right split parameter was set to 0.5, ensuring that 50% of the torque is distributed to each wheel.

3.3.3 Transmission setup

For the transmission setup, the collected information about the transmission that equips the bus will be used. Table 10 presents the parameters that will be applied to configure the bus model.

Table 10 – Virtual model transmission setup

Parameter	Description
Gear ratio	Gear ratio from first to last gear.
Reverse gear ratio	Gear ratio for the reverse gear.
Final ratio	Final Ratio of the transmission.
Up ratio	Value of engine RPM is high enough to increment gear.
Down ratio	Value of engine RPM is low enough to trigger a downshift.

Source: Author (2024).

For the gear ratio setup, the software allows the configuration of the gear ratio for each individual gear. For this, the gear ratios described in Table 2 were used in the setup. For the upshift and downshift parameters, which control the engine speed at which gear changes occur, the software configures the shifts based solely on engine RPM, disregarding other parameters typically used in electronic transmission control, such as throttle position, vehicle speed, and engine load.

To set the gear change speed ratios, the data collected during testing was used, based on the RPM values at which the gear changes occurred. All gear change events that occurred during the test are shown in Table 11.

Table 11 – Gear change events

Gear change event	Speed (km/h)	Start Gear	Final Gear	Gear change RPM
1st	8	1st	2nd	1063
2nd	14	2nd	3rd	1176
3rd	11	3rd	2nd	707
4th	6	2nd	1st	688
5th	12	1st	2nd	1274
6th	17	2nd	3rd	1169
7th	24	3rd	4th	1329
8th	24	4th	3rd	936

Gear change event	**Speed (km/h)**	**Start Gear**	**Final Gear**	**Gear change RPM**
9th	11	3rd	2nd	715
10th	15	2nd	3rd	1324
11th	12	3rd	2nd	725
12th	6	2nd	1st	688
13th	8	1st	2nd	1185
14th	11	2nd	3rd	968
15th	24	3rd	4th	1302
16th	23	4th	3rd	879
17th	11	3rd	2nd	678
18th	12	2nd	1st	672
19th	5	1st	2nd	1159
20th	17	2nd	3rd	1233
21st	25	3rd	4th	1323
22nd	23	4th	3rd	924
23rd	12	3rd	2nd	696
24th	6	2nd	1st	682
25th	8	1st	2nd	1185
26th	18	2nd	3rd	1237
27th	24	3rd	4th	1322
28th	23	4th	3rd	932
29th	12	3rd	2nd	684
30th	6	2nd	1st	681

Source: Author (2024).

The vehicle speed values during gear shifts were also analyzed in the process, with the aim of verifying discrepancies between different gear changes. The speed values can also be found in Table 11 . With the RPM values for gear shifts recorded, the average shift value for each gear, both upshift and downshift, was calculated. Table 12 and Table 13 show the results.

Table 12 – Mean upshift RPM thresholds

Gear	RPM	Standard Deviation (RPM)
1st	1173	75
2nd	1185	120
3rd	1319	12

Source: Author (2024).

Table 13 – Mean downshift RPM thresholds

Gear	RPM	Standard Deviation (RPM)
2nd	682	7
3rd	701	18
4th	918	26

Source: Author (2024).

Furthermore, the final ratio of the transmission can also be set. The final drive ratio corresponds to the differential system ratio. The bus uses a portal-type rear axle mounted with a differential system. The final gear ratio used in this project is provided by Poojitganont et al. (2020), who specify the gear ratio for the ZF AV-132 as 7.38.

3.3.5 Mass and drag setup

The mass and drag values are specified in the vehicle setup section of UE4. The Table 14 provides an overview of each parameter used in the setup process.

Table 14 – Virtual model mass and drag setup

Parameter	Description
Mass	Mass to set the vehicle chassis to
Center of mass offset	Offset for the center of mass of a vehicle.
Drag coefficient	Drag Coefficient of the vehicle.

Source: Author (2024).

For the vehicle's mass, the data was provided by the University transportation department, with a weight of 18,000 kg being considered. As a simplification in the model, the center of mass will be assumed to be at the center of the vehicle.

For the Drag Coefficient (CD), Bhave and Taherian (2014) conducted a study on various bus body designs with different drag coefficients. For the purposes of this work, Model 1, with a CD of 0.56, was selected, as it most closely resembles the vehicle used in this project.

3.3.6 Wheel setup

To configure the wheels' parameters in the software, the information presented in section 3.1.1.4 will be used. Table 15 outlines the parameters that can be adjusted within the software.

Table 15 – Virtual model wheel setup

Parameter	**Description**
Shape radius	Radius of the wheel.
Shape width	Width of the wheel.
Mass	Mass of the wheel.

Source: Author (2024).

The wheel setup for the simulation used a single-wheel configuration on the front axle and a dual-wheel configuration on the rear axle. The single-wheel setup featured a width of 305 mm, a radius of 499.25 mm, and a weight of 69 kg. In contrast, the dual-wheel configuration doubled the width to 610 mm and the weight to 138 kg, while maintaining the same radius of 499.25 mm.

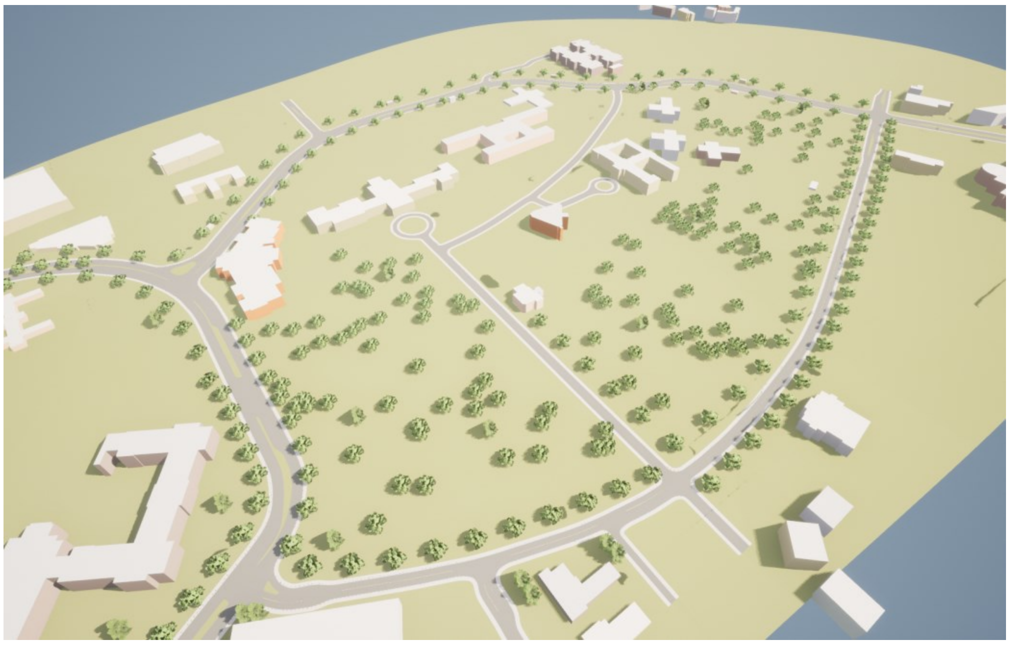
3.4 SIMULATION CONFIGURATION

This section discusses the process from setting up the environment to interfacing with the Python Application Programming Interface (API) and running the simulation for the specific test case presented in this work. The following steps outline the procedure carried out to obtain the simulation results for the virtual bus.

3.4.1 Environment

The first step was to obtain the map of the area of interest, which corresponds to the same area of the university where the real tests were conducted. The map being used had already been created, and the files related to the map were provided for the purpose of this work. With the Filmbox (.fbx) and OpenDrive (.xodr) files in hand, the map was imported into CARLA, built from source. Figure 18 shows the imported map.

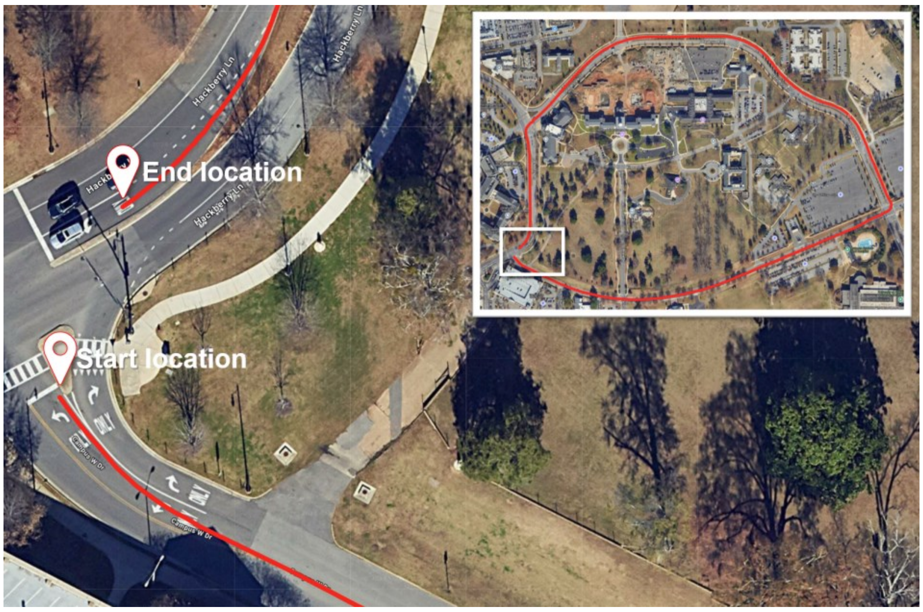
Figure 18 – University map



Source: Author (2024).

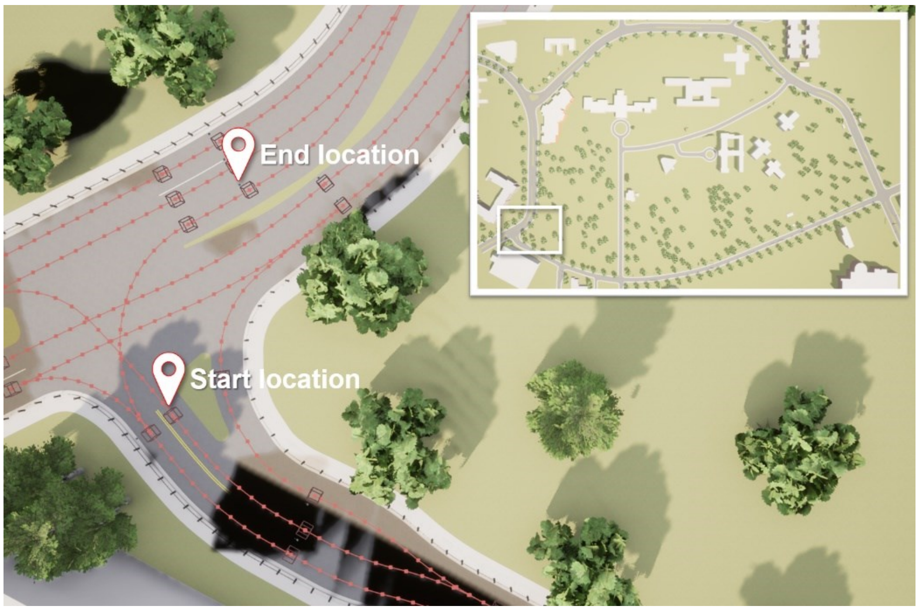
To validate the virtual environment, a test was conducted in the simulation to compare the difference in distance between the virtual world and the real world, using the same start and end points. The test involved retracing the path taken during the real-world test within the simulation and comparing the difference in the distance traveled. For this, the first step was to set the same starting and ending points in the simulation as in the real test. Figure 19 and Figure 20 shows the comparison between the two scenarios.

Figure 19 – Real start location



Source: Author (2024).

Figure 20 – Simulated start location



Source: Author (2024).

To determine the distance in the simulation, the Euclidean distance between each point along the path taken by the vehicle was calculated using the (x, y, z) coordinates. These distances were then iteratively summed to determine the total distance traveled by the bus at the end of the simulation. Table 16 presents the total distance for both worlds and the associated mapping error.

Table 16 – Map error

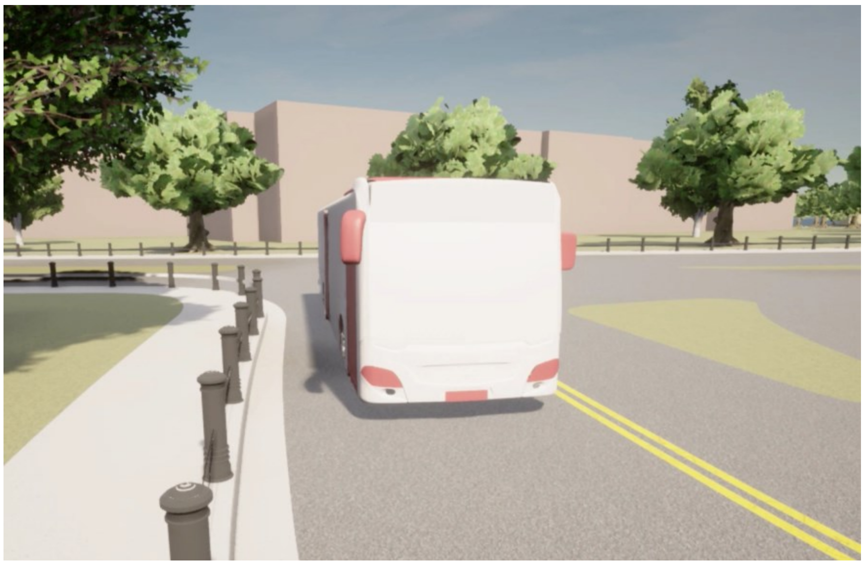
Parameter	Value
Real distance	2510 m
Simulated distance	2459 m
Error	2.03%

Source: Author (2024).

3.4.2 Ego vehicle

For the bus used as the ego vehicle in the simulation, the corresponding model was also imported into CARLA. Specifically, a 2011 Mercedes-Benz Citaro was selected as the ego vehicle for this simulation due to also being a transit bus with a 40-ft length, the same characteristics as the real bus used in the tests. Figure 21 shows the bus model.

Figure 21 – Virtual bus



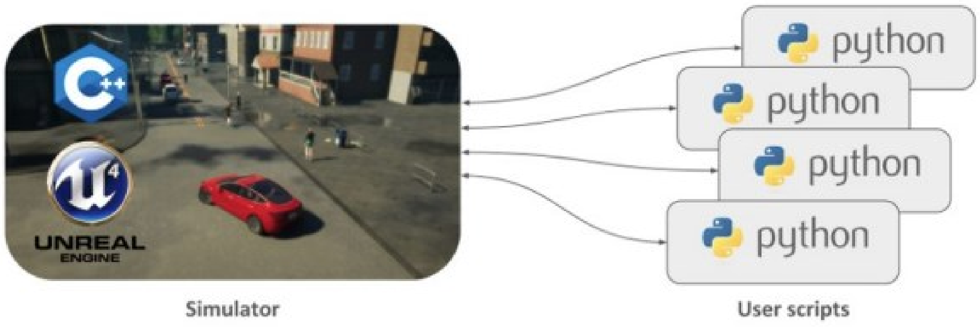
Source: Author (2024).

To import the model into the software, a Filmbox (.fbx) file was used, and the vehicle blueprint was set up. The parameters collected from the model, described in the previous sections, were then input into the software to calibrate the model.

3.4.3 Environment Configuration

CARLA is controlled via a Python API, which provides a link between the simulation environment and the user. The server runs the simulation environment, while the Python script acts as a client, sending commands to control vehicles, spawn actors, and gather data. Figure 22 illustrates the basic structure.

Figure 22 – CARLA basic structure



Source: (FERNANDEZ NARVAEZ, 2021).

Python is a widely used programming language for autonomous simulation due to its extensive library support in this field. In this work, the libraries used include CARLA, which provides the necessary classes and methods to control and interact with the CARLA simulator; NumPy, used for numerical operations such as reshaping images and performing mathematical computations; and OpenCV, which handled image processing and visualization, including displaying RGB camera feeds and overlaying data.

3.4.4 Vehicle Control

To control the vehicle during the simulation, the inputs were directly assigned to the vehicle's torque, brake, and steering variables. The available capabilities of the CARLA library were used to apply the controls to the vehicle. The next section describes the process used to determine the value for each vehicle control parameter.

3.4.4.1 Torque Control

To control the vehicle's acceleration, direct torque control was applied using torque data collected from real-world tests. This data maps torque values specific distances traveled by the vehicle. A Comma-Separated Values (CSV) file containing torque and distance data is used for this purpose. The data points are interpolated, as shown in Appendix A, allowing the system to dynamically determine the appropriate torque value based on the current distance traveled during the simulation. As the simulation progresses, the distance traveled by the virtual bus is calculated in real time, and the corresponding torque value is applied to the vehicle's model.

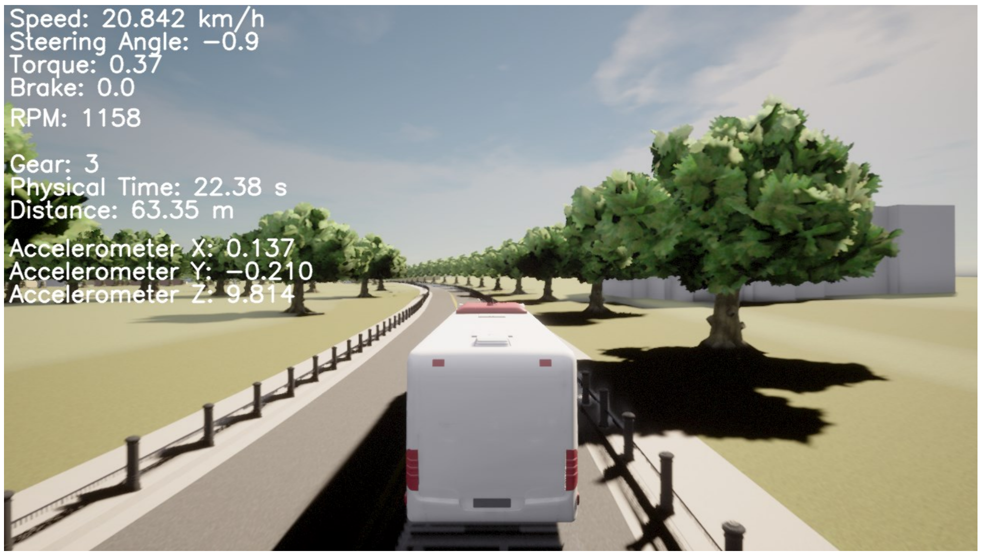
3.4.4.2 Steering

For steering control, a predetermined path is used as input in the simulation. Path generation is managed by the CARLA library, which defines a route based on specified start and end points. The generated route consists of a series of waypoints, with each waypoint specifying a 3D position and orientation. The vehicle's current position is continuously compared to the next waypoint, and the steering angle is calculated at each step to align the vehicle with the predefined path. Appendix B shows how the vehicle is controlled.

3.4.5 Sensors

Two sensors provided by CARLA were used in simulation. One of these was an RGB camera, which was used to capture images of the vehicle during the simulation to analyze its body motion. The camera was positioned 4 meters above and 13 meters behind the vehicle's center, providing a view of the vehicle and its surroundings. This setup also allowed real-time vehicle information to be overlaid on the camera feed, enabling visualization of the vehicle's state and its interaction with the simulated environment, as shown in Figure 23.

Figure 23 – RGB Camera



Source: Author (2024).

Another sensor used from the CARLA library was the IMU sensor, aimed at collecting data in the form of acceleration along the x, y, and z components. These outputs were also displayed on the screen through the RGB camera feed, as shown in Figure 23.

3.4.7 Data collection

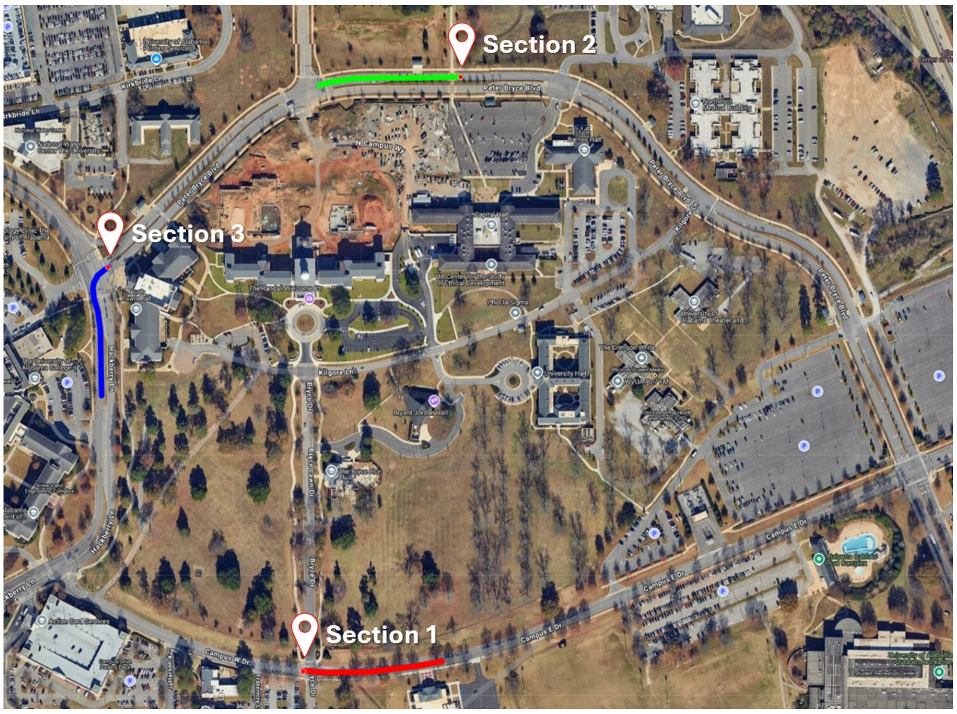
The simulation environment is configured to collect a broad range of data, systematically stored in a CSV file for later analysis. This data includes parameters such as speed, steering angle, torque, brake input, accelerometer readings, RPM, and distance traveled by the vehicle. During each step of the simulation, these values are logged in real time. The data is organized into a CSV format with appropriate headers, allowing for straightforward analysis and processing.

After collecting, the data undergoes post-processing to validate the simulation results. This process involves comparing the logged data with real-world observations or using it to refine the virtual model. The collected data is used for evaluating the effectiveness of the simulation and provides insights for improving the vehicle’s behavior and performance in different scenarios. This structured approach documents all relevant aspects of the vehicle's operation, supporting ongoing development and optimization of the simulation model.

3.5 SIMULATION PROCEDURE

To evaluate the dynamic model and calibrate the vehicle's longitudinal motion during acceleration, data samples collected from real-world tests were used, and simulations of these same scenarios were conducted for comparison. For calibration, three different simulations were performed, each consisting of acceleration tests starting from a stationary position and covering a distance of 150 meters. Acceleration-only scenarios were chosen for the analysis. Each of these acceleration scenarios was conducted in a different location on the map to assess the model's performance under varying conditions. Figure 24 shows the starting positions for each analysis section.

Figure 24 – Map sections



Source: Author (2024).

Each acceleration scenario was conducted at a different location on the map to evaluate the model's performance under varying conditions. The vehicle followed the route in an anticlockwise direction. Section One represents the first instance of acceleration from a stationary position, which occurred 360 meters after the start point of the real-world test, at an intersection with a stop sign. Section Two occurred at 1,738 meters from the start, also after a stop, and Section Three took place at 2,185 meters from the beginning of the test, following a stop at a traffic light.

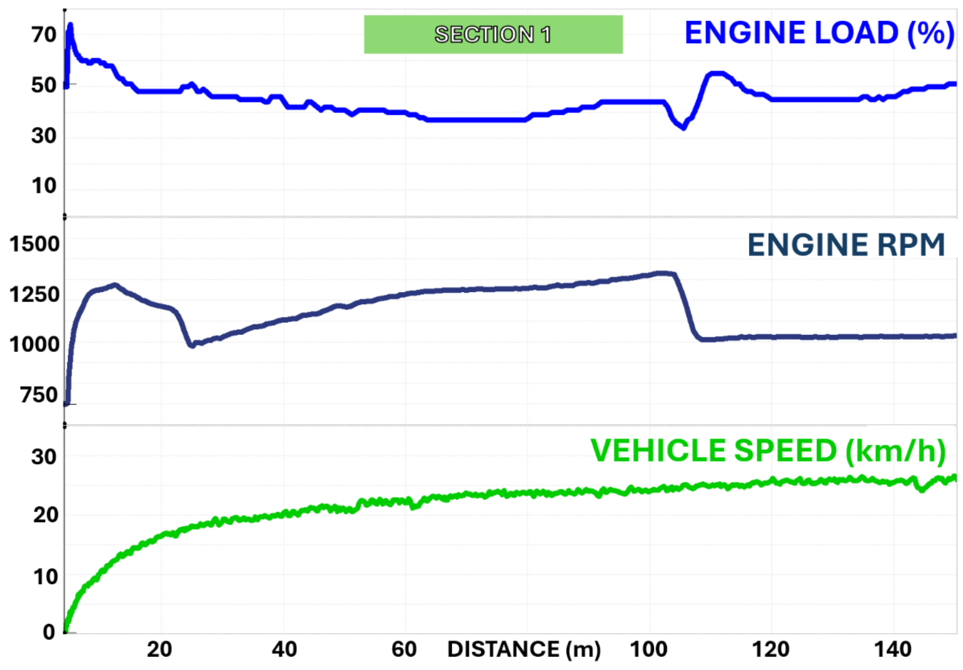
4. RESULTS

In this chapter, all results from both the real tests and the CARLA model are presented and analyzed. Additionally, the data from both sources will be compared and the vehicle parameters are going to be tweaked to calibrate and validate the virtual model, which is the primary objective of this work. The similarities and differences between the two sets of data will be discussed in detail.

4.1 DATA COLLECTION RESULTS

For the dynamic analysis of the real-world test, AIM Race Studio software was used. The collected data was analyzed through the software interface, focusing on engine data to evaluate the vehicle's acceleration response in the selected sectors of the study. In all three sectors, the engine output showed a similar pattern, characterized by a peak in torque during the initial launch from a stationary position and torque variations during gear shifts, which are evident from the sudden RPM drops in the graphs. Another consistent trend across all three scenarios is the speed profile, which reaches a maximum of approximately 25 km/h in each case. Figure 25 illustrates the engine output for Section One.

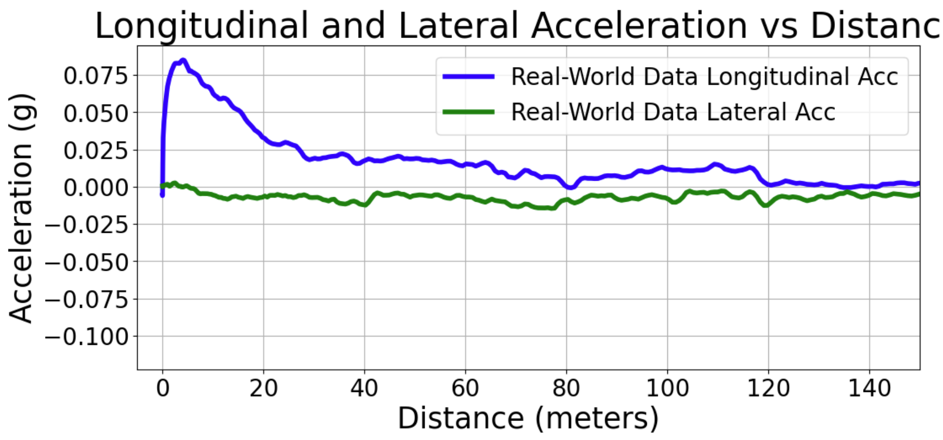
Figure 25 – Real-world test data Section One



Source: Author (2024).

In Section One, the torque peak occurs at a speed of 3 km/h, reaching a value of 71% at 839 RPM, which corresponds to a torque of 901.2 Nm at that point. In addition to engine data, the longitudinal acceleration data was also analyzed, with the maximum value reaching approximately 0.80 m/s², as shown in Figure 26. This is a lower value compared to other urban bus measurements, such as 2.18 m/s² on a dry asphalt surface reported by Frej et al. (2023).

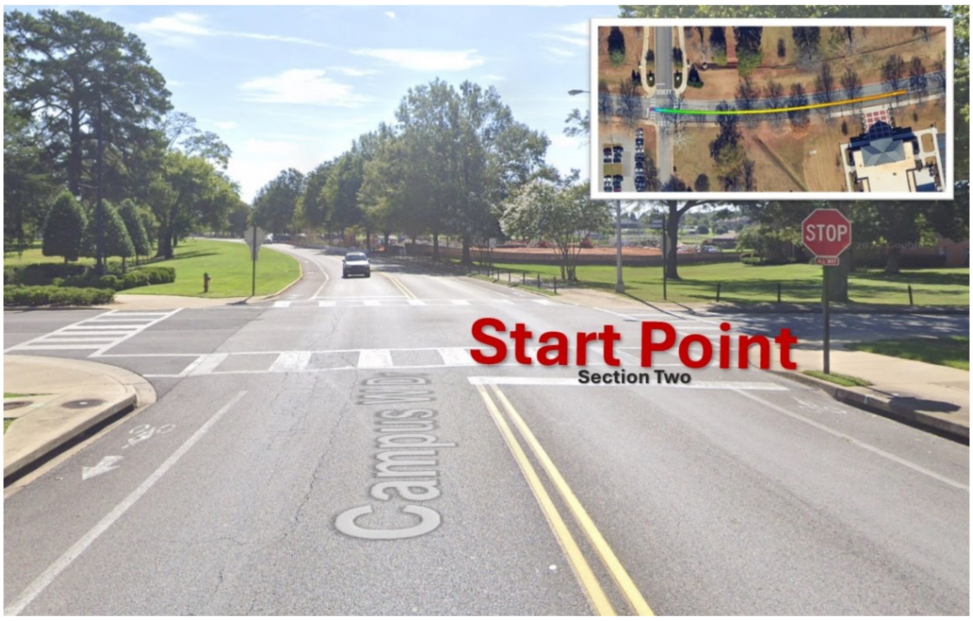
Figure 26 – Longitudinal and lateral acceleration at Section One



Source: Author (2024).

For lateral acceleration, although the values remained relatively low throughout the path, a tendency towards negative values can be observed, which is due to the left-hand turn with a long radius, as shown in Figure 26. The terrain elevation also varies along the route; this section is characterized by an uphill slope with an elevation gain of 1.5 meters from the starting point, as seen in Figure 27. The starting point is characterized by being next to a stop sign. It is important to note that the IMU sensor was mounted at the front of the vehicle during the test.

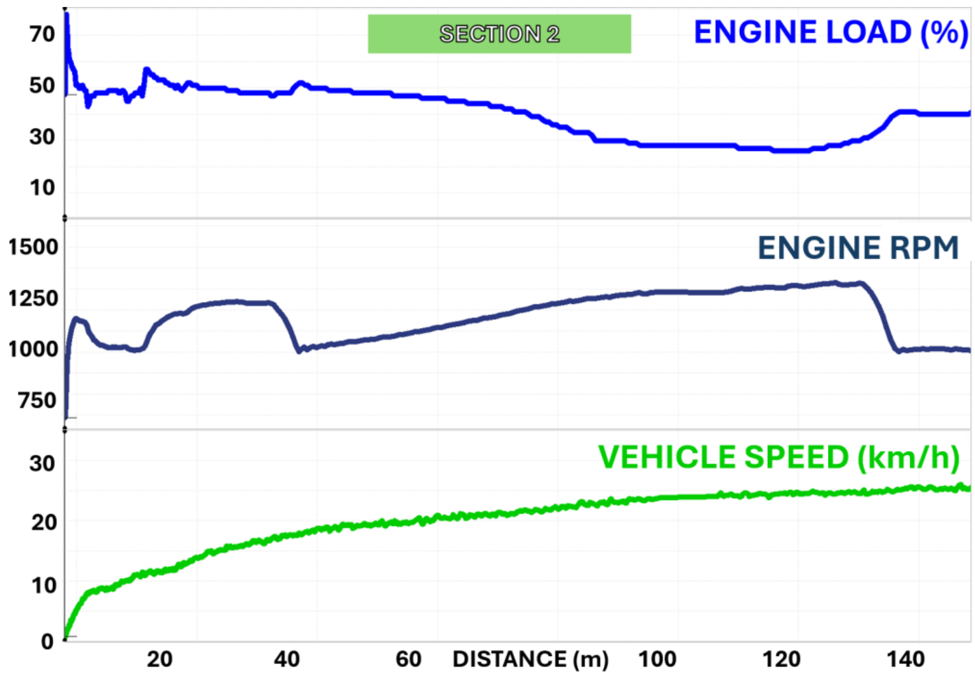
Figure 27 – Street view Section One



Source: Author (2024).

In Section Two, the torque peak occurs at a speed of 1 km/h, with a value of 77% at 858 RPM, resulting in a torque peak of 916.4 Nm at that point. Another point to analyze in the engine data is the behavior of the engine immediately after the vehicle's launch. As seen in Figure 28, during the first 30 meters, the engine behavior does not follow the expected gradual RPM increase alongside speed, with sudden drops caused by gear changes. This can be explained by the automatic transmission with a torque converter used in the bus. This phenomenon is described by Diachuk and Easa (2022), who explain that the torque converter experiences slippage between the engine and transmission, especially at low speeds. This results in engine RPM that does not immediately correspond to wheel speed, causing fluctuations until the system reaches a more stable state. This relationship between the engine coupling and the gearbox torque converter is also illustrated in Figure 34.

Figure 28 – Real-world test data Section Two

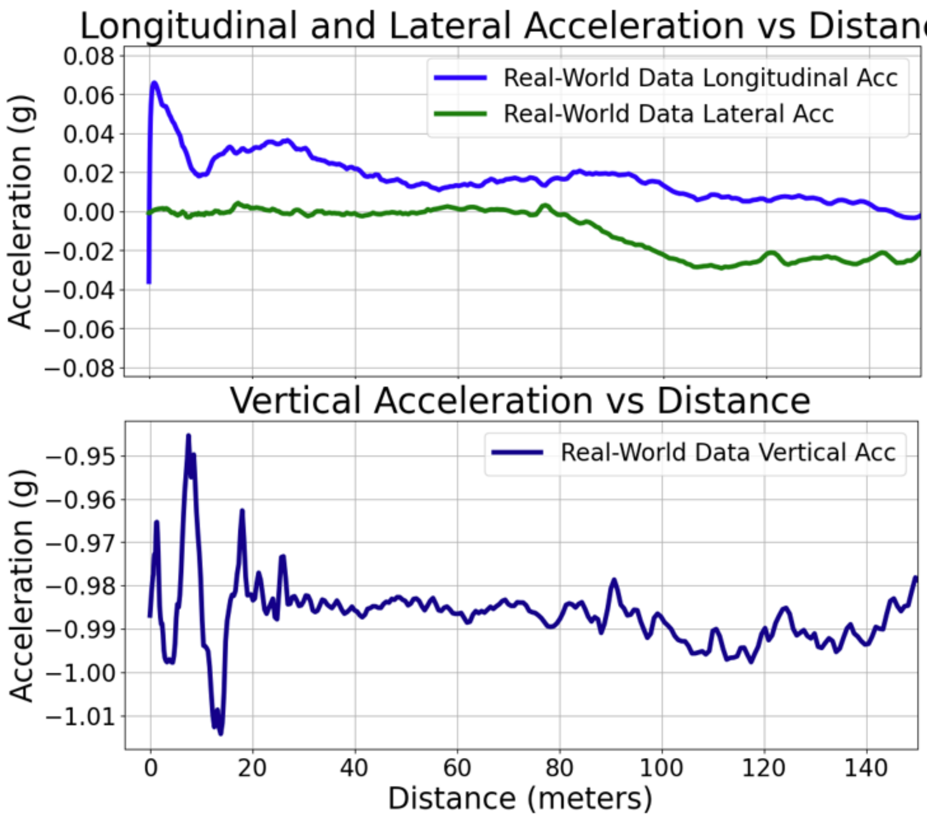


Source: Author (2024).

For the acceleration analysis in this section, the longitudinal acceleration shows a peak occurring during the vehicle's launch at low speed, as indicated in Figure 29. This pattern is consistent across all three sections analyzed. The peak occurs because, during launch, the transmitted torque increases from zero to a high value, which instantly induces higher acceleration. High acceleration values can cause discomfort for passengers; Szadkowski and Morford (1992) have shown that limiting acceleration to 0.15 g ensures a comfortable launch, which is greater than the peak longitudinal accelerations recorded in all three sections.

Regarding the path, it is characterized by an initial straight section of 80 meters, followed by a long radius left curve over the next 70 meters. Figure 29 shows the lateral acceleration profile, which peaks at approximately 0.3 m/s². Additionally, there is an elevation gain of 2 meters between the start and end points of this section, as shown in Figure 30, which illustrates the actual path.

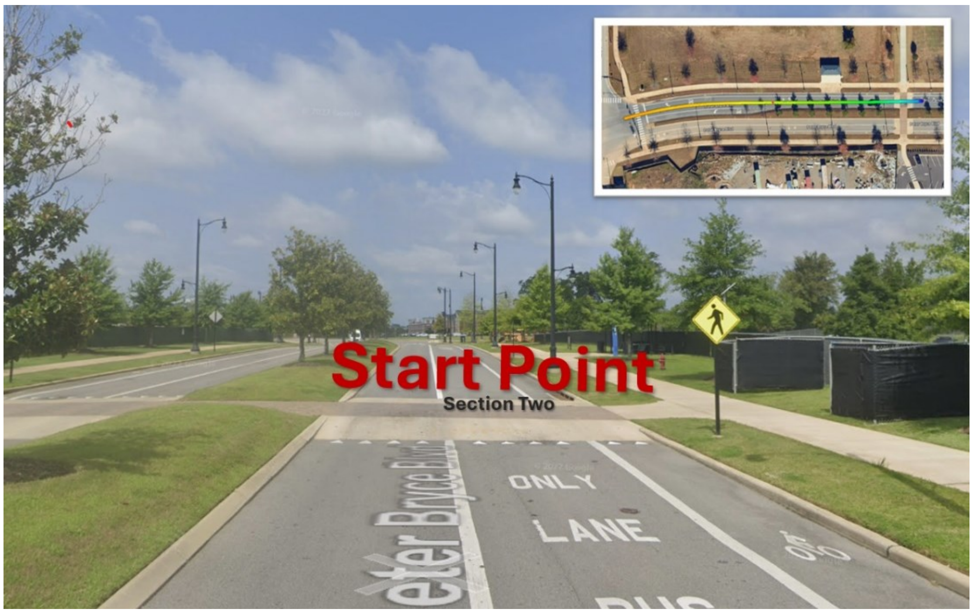
Figure 29 – Longitudinal, lateral and vertical acceleration at Section Two



Source: Author (2024).

Additionally, in the second section, a speed bump at the beginning of the sector caused a larger amplitude variation in vertical acceleration, as seen in Figure 29. The sensor output registered -0.99 g when the vehicle was at rest, which can be attributed to the inertial accelerometer capturing gravitational acceleration. The reason the output is not exactly -1 g is that the IMU sensor's z-axis was not perfectly vertical during the test. The vertical acceleration data collected during the test is used to identify speed bumps and road irregularities. Figure 30 shows a view of the actual street

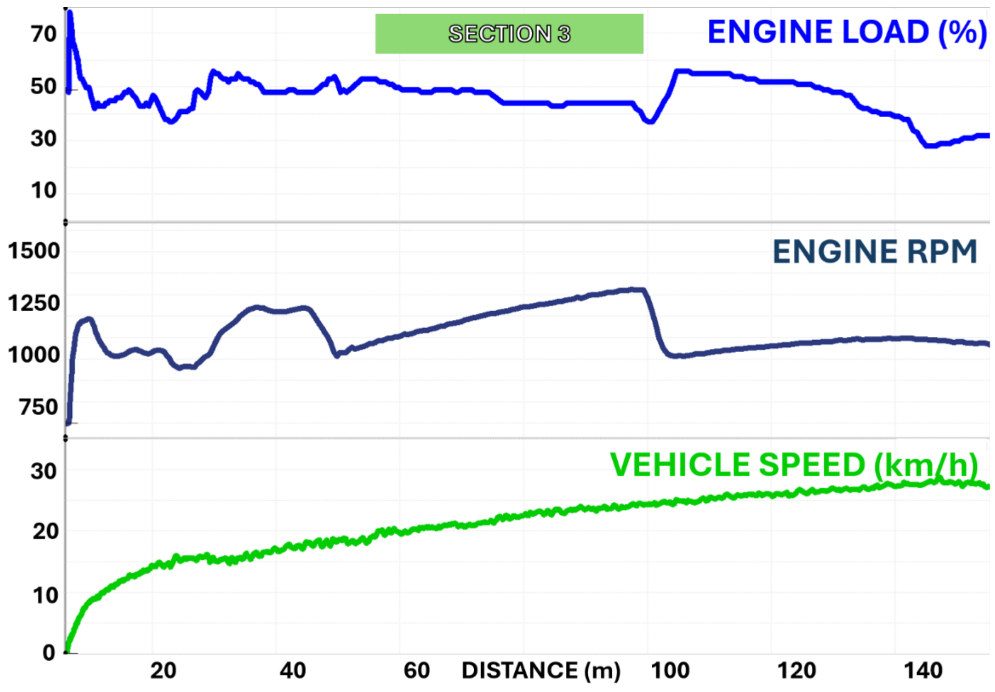
Figure 30 – Street view Section Two



Source: Author (2024).

In Section Three, the torque peak occurs at a speed of 2 km/h, reaching a value of 77% at 826 RPM, resulting in a torque of 890.8 Nm at that point. Figure 31 shows the engine data. This section is characterized by an intersection at the start, as seen in Figure 33. Due to this intersection, the bus crossed it at a lower speed, making Section Three the sector that required the longest distance to reach 20 km/h, taking 58 meters compared to 53 meters in Section Two, which also included a speed bump. This highlights the need for a greater distance to reach higher speeds compared to Section One, where the vehicle reached 20 km/h in 32 meters.

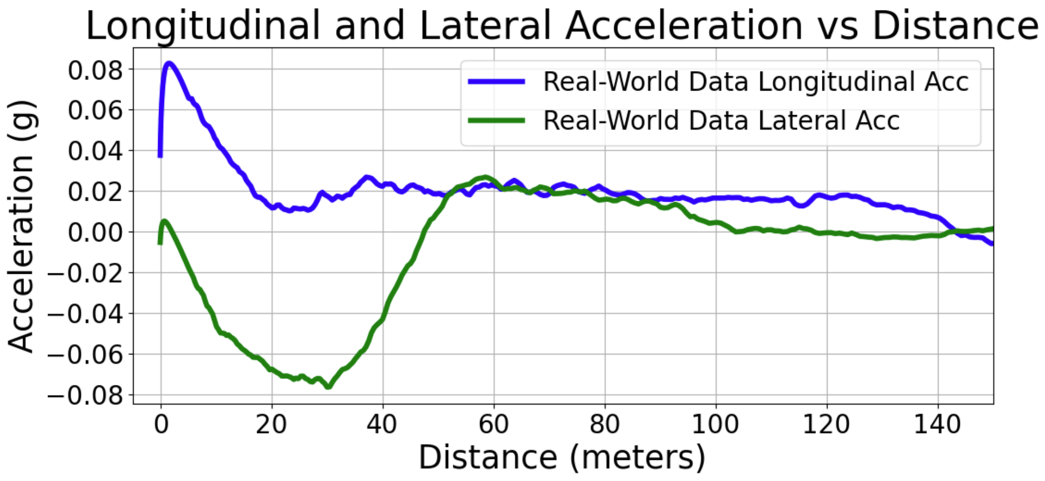
Figure 31 – Real-world test data Section Three



Source: Author (2024).

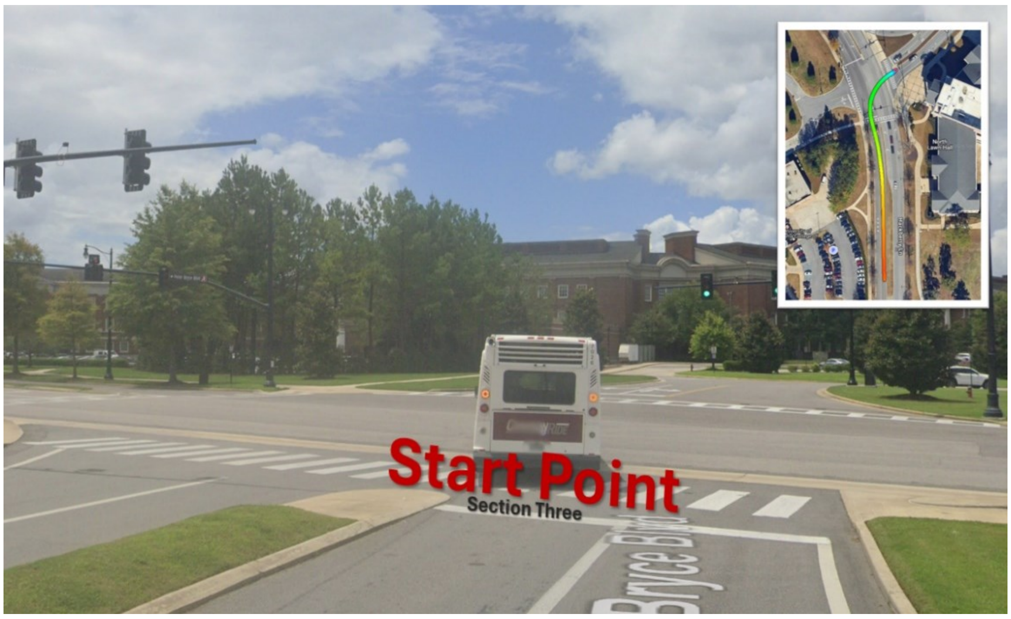
Due to the left-hand turn at the beginning of the section, a higher variation in lateral acceleration is observed in this sector, as shown in Figure 32. The peak value was 0.74 m/s² at a speed of 15 km/h during a turn with a radius of 22.12 meters, which is below the limit proposed by Esveld (2001), who suggests that maximum lateral acceleration should be 1 m/s² for comfort reasons. Just after the first left-hand turn, the path features a right-hand curve, resulting in a positive value for lateral acceleration on the graph. The top right corner of Figure 33 shows the trajectory with a color gradient indicating speed variation. Regarding the z-coordinate, this section is characterized by constant elevation, with no significant changes.

Figure 32 – Longitudinal and lateral acceleration at Section Three



Source: Author (2024).

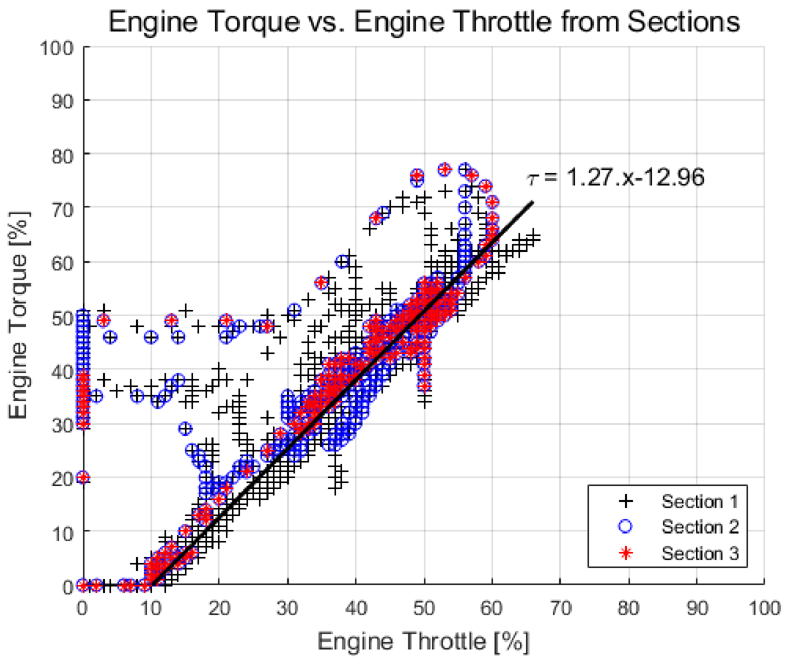
Figure 33 – Street view Section Three



Source: Author (2024).

The correlation between engine torque and throttle position, as shown in Figure 34, is approximately linear. There is, however, a portion of nonlinearity caused by the coupling of the gearbox torque converter. An analysis of the three measured sections indicates similar behavior across all of them. In the linear portion of the relationship between torque and throttle position, a linear regression produced a slope of 1.27 torque per unit of throttle. Consequently, a linear relationship between engine torque and accelerator pedal position (throttle) was adopted in the model adjustment.

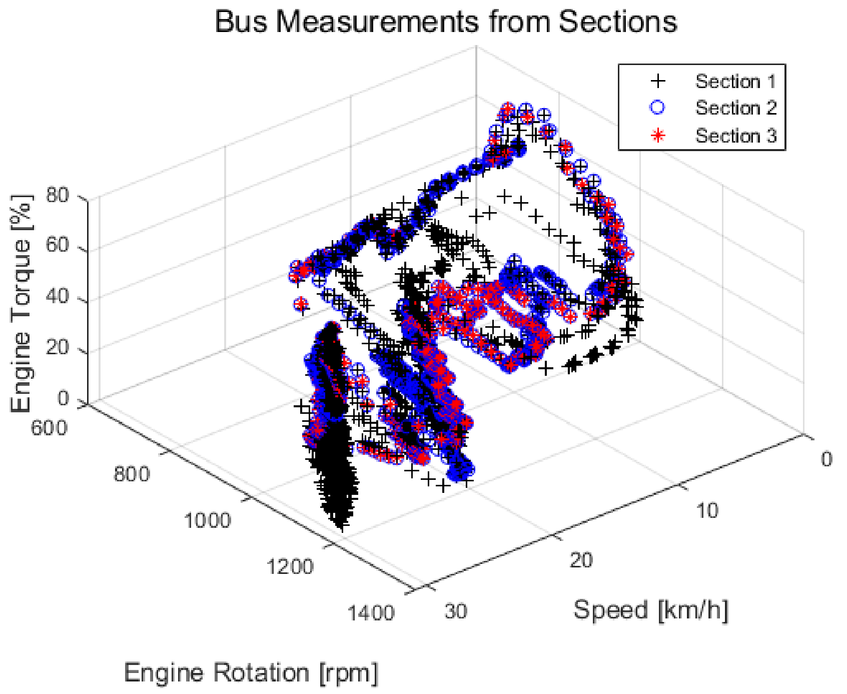
Figure 34 – Engine torque vs. throttle data for the three sections



Source: Author (2024).

Furthermore, comparing the three sections of the bus and engine dynamics, as illustrated in Figure 35 using a 3D plot of engine torque as a function of engine rotation and vehicle speed, the three curves from the section measurements exhibited similar behavior. This indicates that the tested conditions were very similar across the sections, making them suitable for comparison during the model update and validation process.

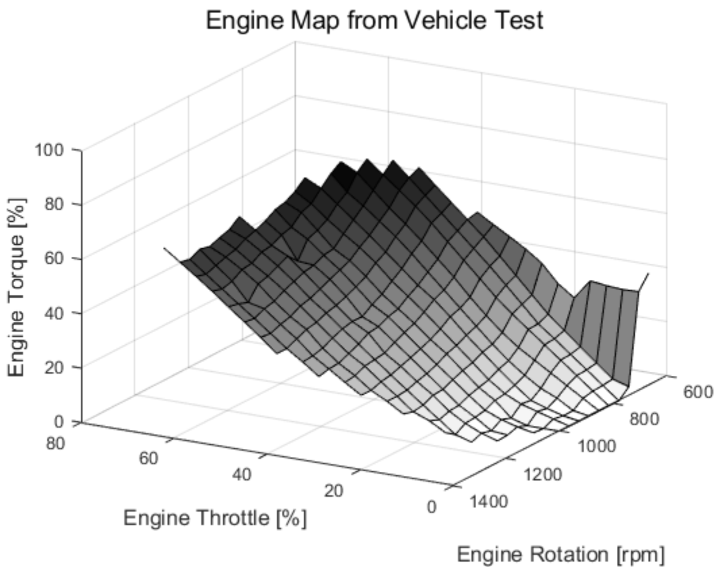
Figure 35 – Torque vs. RPM vs. speed for the three section



Source: Author (2024).

Figure 36 illustrates the relationship between main engine parameters - torque, engine rotation, and throttle position - based on the measurements of engine performance. The engine exhibits a general linear relationship between these parameters, except at lower engine speeds, around 800 RPM, where the torque converter's influence alters the curve, resulting in higher torque at low engine rotation. This occurs during take-off situations when the vehicle is stationary and begins accelerating, necessitating synchronization between the torque converter's pump and turbine before the lock-up clutch is engaged. Consequently, under these powertrain characteristics, the CARLA software model is likely more representative during steady-state movement (after the clutch is engaged) compared to the take-off phase.

Figure 36 – 3D engine map

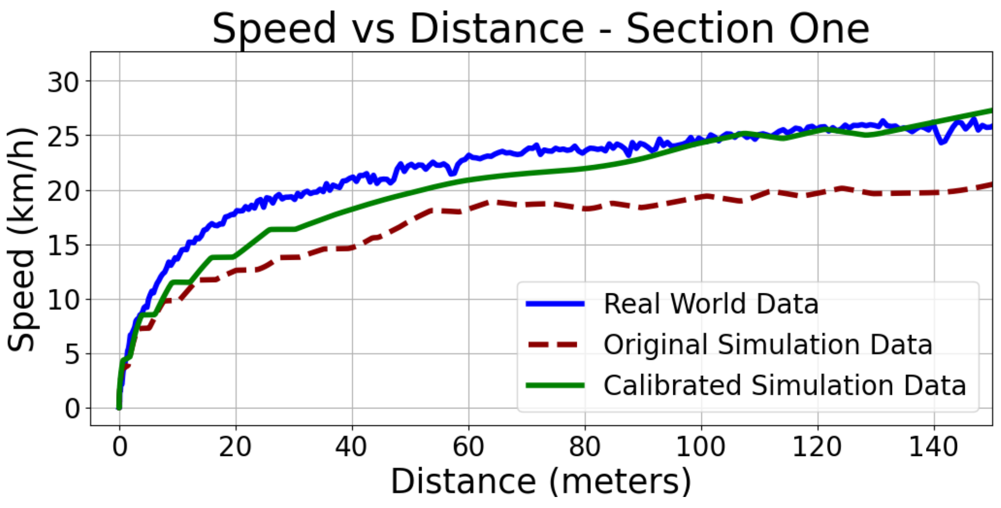


Source: Author (2024).

4.2 SIMULATION RESULTS AND CALIBRATION

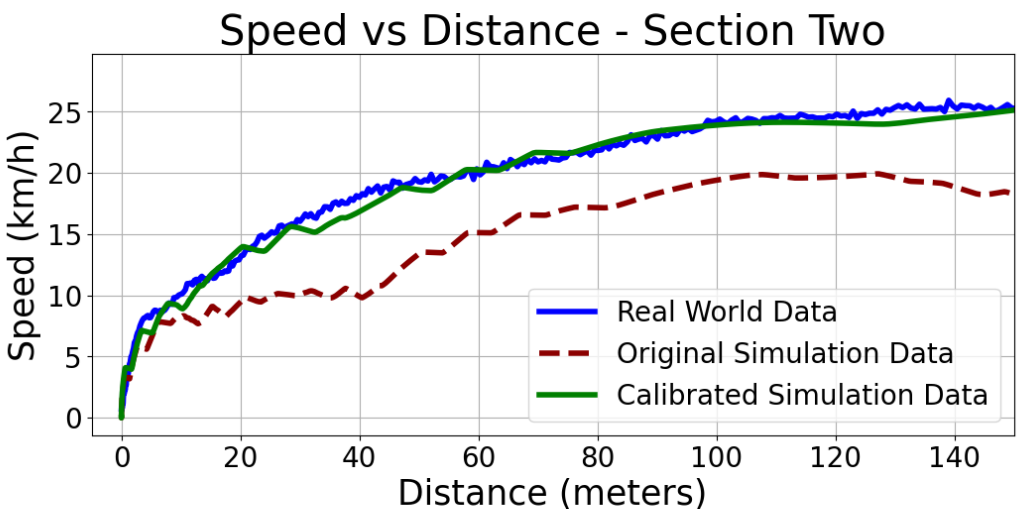
The simulation process was divided into two stages, consisting of six simulations in total: three using the original parameters and three additional simulations with the calibrated parameters. To calibrate the vehicle model in CARLA, the speed versus distance data for each sector was used as a reference, aiming to align the simulated curves with the real-world test data. Figure 37, Figure 38 and Figure 39 show the results.

Figure 37 – Speed vs. distance simulation results for the Section One



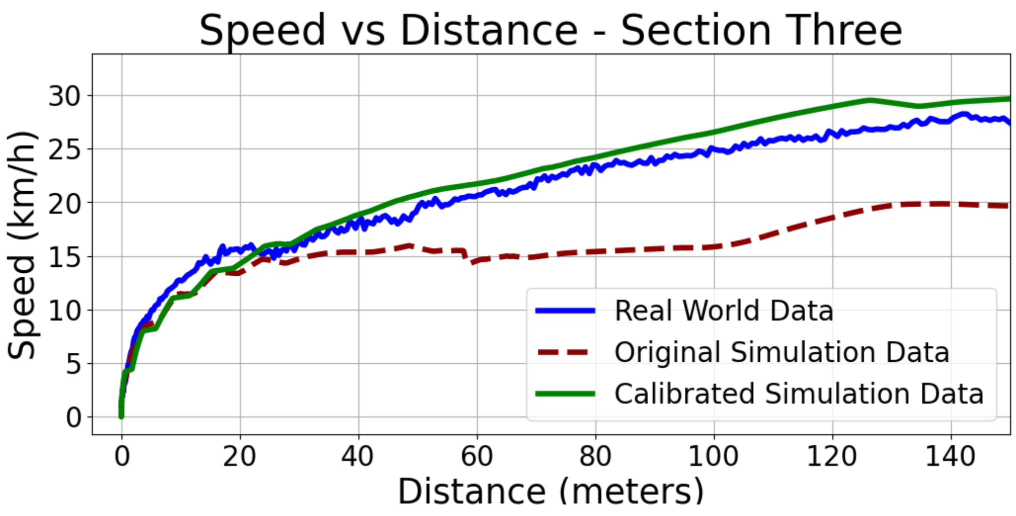
Source: Author (2024).

Figure 38 – Speed vs. distance simulation results for the Section Two



Source: Author (2024).

Figure 39 – Speed vs. distance simulation results for the Section Three



Source: Author (2024).

The figures above compare the speed data obtained from real-world tests, the original simulation, and the simulation with the calibrated model. The original simulation results, represented by the dark red curve, used the parameters detailed in Section 3.3 of this work. Table 17 summarizes the parameters applied in the original setup.

The difference between the speed profiles of the original and calibrated simulations can be seen in the graphs. The green curve, representing the calibrated simulation, closely aligns with the blue curve, which corresponds to the real-world test data. In contrast, the dark red curve from the original simulation shows a greater deviation, highlighting the performance difference between the uncalibrated model and the actual vehicle behavior.

The results analysis of the original vehicle simulation provides the information on the adjustments needed to align the simulated and real-world curves. The original test revealed a lack of acceleration performance in the virtual scenario compared to the real-world data for the same torque input, resulting in the inability to maintain speed as in the real scenario. To address this, a calibration process was carried out. Since the focus was on calibrating the longitudinal dynamics of the vehicle during acceleration, adjustments were primarily made to the vehicle's mass and torque parameters.

Table 17 – Bus parameters for original simulation

Parameter	**Value**
Max engine speed	2100 RPM
Differential type	Open Rear Drive
Rear left right split	0.5
Gear ratio first gear	3.36
Gear ratio second gear	1.91
Gear ratio third gear	1.42
Gear ratio fourth gear	1
Gear ratio fifth gear	0.72
Gear ratio sixth gear	0.62
Reverse gear ratio	9.84
Final ratio	7.38
Vehicle mass	18000 kg
Drag coefficient	0.56
Wheel radius	499.25 mm
Front wheel width	305 mm
Rear wheel width	610 mm
Front wheel mass	69 kg
Rear wheel mass	138 kg

Source: Author (2024).

. To improve the alignment of the simulated curves with the real-world data, a successive approximation method was employed. Through consecutive progressive changes to the parameters of mass and torque, simulations for the three sectors were performed for each successive approximation until the simulated speed curve closely matched the real-world data. The goal was to ensure that the simulated curves aligned with the real-world data across all three sectors.

The calibrated model considered in this work was the one that provided the best match for the speed versus distance curves in all three sectors. The objective was to achieve a calibration that accounted for the three different scenarios. The calibration was evaluated during each successive approximation by analyzing the proximity of the curves for all three cases.

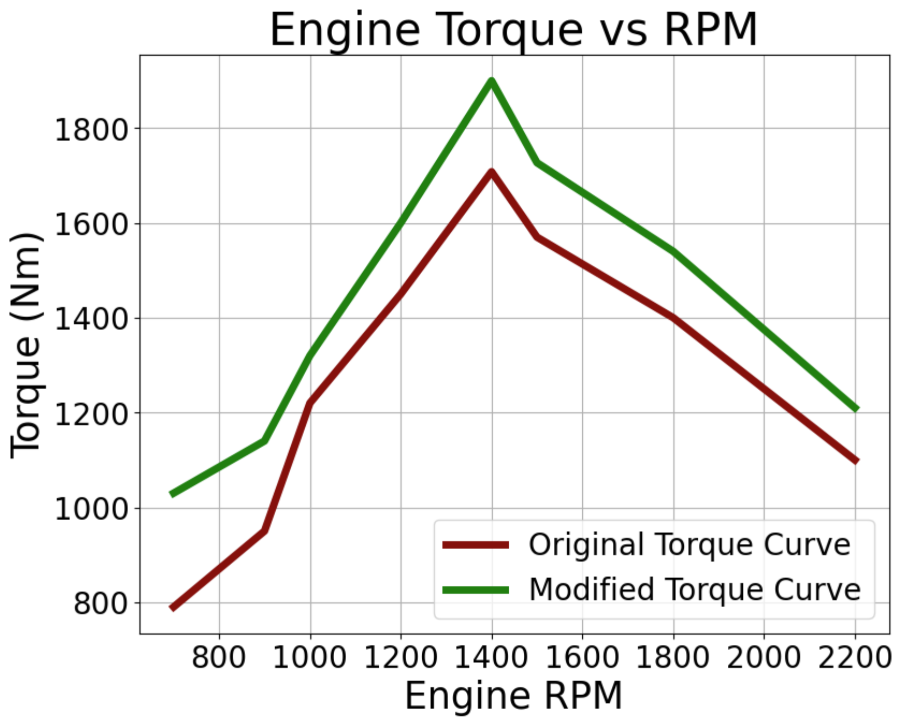
With the calibration completed, the mass of the calibrated vehicle was set to 14,000 kg, which is 4,000 kg less than the original parameter of 18,000 kg, as shown in Table 17. Regarding the torque, the curve set in the software was modified, where the torque values at certain RPM points, as presented in Table 8, were multiplied by a scaling factor. Table 18 shows the adjusted torque values, and Figure 40 illustrates the original and modified torque curves.

Table 18 – Original and modified torque curve parameters

Engine RPM	**Original Torque (Nm)**	**Scaling Factor**	**Adjusted Torque (Nm)**
700	790	1.3	1030
900	950	1.2	1140
1000	1220	1.1	1320
1200	1450	1.1	1600
1400	1708	1.1	1900
1500	1570	1.1	1730
1800	1400	1.1	1540
2000	1250	1.1	1375
2200	1100	1.1	1210

Source: Author (2024).

Figure 40 – Original and modified torque curve



Source: Author (2024).

To evaluate the accuracy of the calibration, an analysis of the error between the original and calibrated results was performed. For this, the Root Mean Square Error (RMSE) metric was used, as it is useful for measuring the difference between model-predicted values and actual observed values (HODSON, 2022). The calculation is performed using the equation below.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2} \tag{1}$$

Where n is the number of measurements, y_i is the model value, and p_i is the measured value. The RMSE values for both the original and calibrated parameters of the vehicle model are shown in Table 19 and Table 20.

Table 19 – RMSE for the three sectors using the original calibration

Section	Original RMSE
One	5.224
Two	5.296
Three	6.339

Source: Author (2024).

Table 20 – RMSE for the three sectors using the modified calibration

Section	Calibrated RMSE
One	1.878
Two	0.719
Three	1.608

Source: Author (2024).

Analyzing the errors across the sections, the reduction for Sector One was 2.78 times, for Sector Two it was 7.37 times, and for Sector Three it was 3.94 times. A more precise calibration can be observed for Sector Two, which is also reflected in the convergence of the curves shown in Figure 38. The reductions in the other sectors were also significant, although Sector One showed the smallest improvement. This highlights an area for further study in future calibration efforts.

6. CONCLUSION

The alignment between simulators and real-world conditions is essential for developing and validating Advanced Driver Assistance Systems (ADAS) and autonomous driving technologies. CARLA provides a safe environment for replicating complex urban scenarios, reducing the costs and risks of physical tests, ensuring accurate and repeatable results, and supporting the validation of perception systems and decision-making processes. Building on this foundation, this study focused on calibrating the longitudinal dynamics of a bus within the CARLA Simulator.

Regarding the proposed objectives, the goals set for this research were achieved. Real-world data acquisition was successfully carried out, providing the necessary information for calibration development. Additionally, the test offered an initial understanding of the actual vehicle where ADAS capabilities will be implemented. The results included engine data, essential for virtual model calibration, as well as acceleration data under normal operating conditions. Peak accelerations of 0.8 m/s^2 were observed, which is considered within the expected range for passenger comfort.

The parameters of the real bus were also collected to replicate its original condition, allowing for a detailed study of its components. The calibration process successfully improved the longitudinal dynamics within the analyzed distance range, reducing errors by 2.78 times in the least improved sector and by 7.37 times in the most adjusted sector. This study also marked the initial use of the CARLA Simulator in the research project to which this work belongs, a tool that will continue to be used throughout the project.

Finally, this work represents a first step in the research and development of an autonomous bus. Future efforts could focus on further improving the dynamic model, such as measuring the vehicle's weight with more precise methods, implementing a PID controller based on the speed profile to refine calibration using data like torque, engine RPM, and lateral and longitudinal acceleration, and extending the calibration to include the lateral dynamics of the model.

REFERENCES

AHMED, S. R.; WOLF-HEINRICH HUCHO; AL, E. **Aerodynamics of road vehicles: from fluid mechanics to vehicle engineering**. Warrendale: Society Of Automotive Engineers, 1998.

AIM Sportline. **Homepage**. Available at: <https://www.aim-sportline.com/\>. Accessed on: October 16, 2024.

ALCOA WHEELS. ***ULA18x***. Available at: <https://www.alcoawheels.com/north-america/en/products/ula18x/>. Accessed: October 21, 2024.

BARTOLI, Y. et al. 1000-hour durability evaluation of a prototype 2007 diesel engine with aftertreatment using B20 biodiesel fuel. **SAE International Journal of Fuels and Lubricants**, v. 2, n. 2, p. 290-304, 2009.

BEG, M. S.; YUSRI, I. M. Investigation of collision estimation with vehicle and pedestrian using CARLA simulation software**. Journal of Mechanical Engineering and Sciences**, v. 18, n. 1, p. 9949-9958, 2024.

BHAVE, A.; TAHERIAN, H. **Aerodynamics of intercity bus and its impact on co2 reductions**. Proceedings of the Fourteenth Annual Early Career Technical Conference, [*s. l.*], 1 nov. 2014.

BROGLE, C.; ZHANG, C.; LIM, K. L.; BRÄUNL, T. **Integration of CARLA simulator with hardware-in-the-loop for autonomous vehicle testing.** IEEE International Conference on Robotics and Automation, 4 Mar. 2019, Montreal, Canada. IEEE, 2019. p. 1234-1240.

ESVELD, C. **Modern railway track**. Dior Zwarthoed-van Nieuwenhuizen ed. Zaltbommel: Mrt Productions, 2001. v. 2

CANALE ROMEIRO, N.; SALOMÃO, A.; FRAPORTI ZANINI, L. M.; HORN VIEIRA, M. L. Research of photogrammetry transformation into virtual reality environments suitable for use in virtual reality. **Anais do Congresso Internacional de Conhecimento e Inovação – ciki**, *[S. l.]*, v. 1, n. 1, 2023. Avaiable at: https://proceeding.ciki.ufsc.br/index.php/ciki/article/view/1433. Accessed on: Nov. 20, 2024.

CUMMINS. **ISL Engine**. Available at: <https://www.cummins.com/engines/isl>. Accessed: October 21, 2x024.’

DIACHUK, M.; EASA, S. M. **Modeling combined operation of engine and torque converter for improved vehicle powertrain’s complex control**. Vehicles, v. 4, n. 2, p. 501-528, 2022. Available at: https://doi.org/10.3390/vehicles4020030. Accessed on: Nov. 19, 2024.

FERNANDEZ NARVAEZ, P. J. **CARLA-based Simulation Environment for Testing and Developing Autonomous Vehicles in the Linden Residential Area.** 2021. Master's thesis – Ohio State University, Ohio. Available at: http://rave.ohiolink.edu/etdc/view?acc_num=osu1619122432157249. Accessed on: November 18, 2024.

FREJ, D.; GRABSKI, P.; JURECKI, R. S.; SZUMSKA, E. M. **Experimental study on longitudinal acceleration of urban buses and coaches in different road maneuvers**. Sensors, v. 23, n. 3125, 2023.

GENTA, G.; MORELLO, L. **The Automotive Chassis**. [s.l.] Springer Nature, 2019.

GÓMEZ-HUÉLAMO, C. et al. Train here, drive there: ROS based end-to-end autonomous-driving pipeline validation in CARLA simulator using the NHTSA typology. **Multimedia Tools and Applications**, v. 81, n. 3, p. 4213–4240, 3 dez. 2021.

HE, J.; WANG, Z.; LIU, M. **End-to-end autonomous driving using conditional imitation learning and DDPG**. Applied Sciences, v. 12, n. 22, p. 12345-12360, 2022.

HODSON, T. O**.** Root-mean-square error (rmse) or mean absolute error (mae):when to use them or not**. Geoscientific Model Development**, v. 15, n. 14, p. 5481–5487, 2022

HUSSAIN, R.; ZEADALLY, S. Autonomous Cars: Research Results, Issues, and Future Challenges. **IEEE Communications Surveys & Tutorials**, v. 21, n. 2, p. 1275–1313, 2019.

LEE, A.; SMITH, J.; KIM, H. Dynamic Autonomous Driving Strategies Using Reinforcement Learning. In: **Proceedings of the 37th AAAI Conference on Artificial Intelligence**, 13 Feb. 2024, New York, USA. AAAI Press, 2024. p. 2315-2323.

LI, Y.; SHI, H. **Advanced Driver Assistance Systems and Autonomous Vehicles**. [s.l.] Springer Nature, 2022.

MICHELIN. **305/70 R22.5 X METRO HD TL**. Available at: <https://metro.michelin.com/en/tires/305-70-r22-5-x-metro-hd-tl/>. Accessed: October 21, 2024.

MOLA, S.; GENERAL MOTORS INSTITUTE. **Fundamentals of vehicle dynamics.** Flint, Mich.: Product Engineering Dept., General Motors Institute, 1969.

NENSETH V., CICCONE A., KRISTENSEN N.. **Societal consequences of automated vehicles**: Norwegian scenarios. TØI report 1700, [*s. l.*], 2019.

NOVABUS. **Nova LFS Diesel**. Available at: <https://novabus.com/blog/bus/lfs_diesel/>. Accessed on: October 16, 2024.

OLSTAM, J. et al. An algorithm for combining autonomous vehicles and controlled events in driving simulator experiments. **Transportation Research Part C: Emerging Technologies**, v. 19, n. 6, p. 1185–1201, dez. 2011.

PATEL K. **A simulation environment with reduced reality gap for testing autonomous vehicles**. 2020. Thesis (Master of Science in Computer Science) – University of Windsor, Windsor, Ontario, 2020. Available at: <https://scholar.uwindsor.ca/etd/8305>. Accessed on: October 17, 2024.

POOJITGANONT, T. et al. Efficiency and Emission Simulations of Hydrogen-Fuel City Buses. **IOP Conference Series: Materials Science and Engineering**, v. 886, p. 012025, 28 jul. 2020.

SIVAYAZI, K.; MANNAYEE, G. **Autonomous navigation for safe open environments using double DQN algorithm.** e-Prime - Advances in Electrical Engineering, Electronics and Energy, v. 8, p. 100581, 2024.

SONG, M.; SHIN, D. A study on ego-motion estimation based on stereo camera sensor and 2G1Y inertial sensor with considering vehicle dynamics. **Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering**, v. 233, n. 8, p. 2174-2186, 2018.

STEVIĆ, S.; KRUNIĆ, M.; DRAGOJEVIĆ, M.; KAPROCKI, N**. Development of ADAS perception applications in ROS and Software-In-the-Loop validation with CARLA simulator.** Telfor Journal, v. 12, n. 1, p. 40-45, 2020.

SZADKOWSKI, A.; MORFORD, R. B. Clutch Engagement Simulation: Engagement Without Throttle. **SAE technical papers on CD-ROM/SAE technical paper series**, 1 fev. 1992.

UE4. **Unreal Engine**. Unreal Engine. Available at: https://www.unrealengine.com/en-US/. Accessed on: March 2, 2021.

WON, M.; KIM, S. Simulation Driven Development Process Utilizing Carla Simulator for Autonomous Vehicles. In: **12th INTERNATIONAL CONFERENCE ON SIMULATION AND MODELING METHODOLOGIES, TECHNOLOGIES AND APPLICATIONS (SIMULTECH)**, 2022, Belgrade. Anais [...]. Belgrade: SCITEPRESS – Science and Technology Publications, 2022. p. 202-209. DOI: 10.5220/0011139300003274

ZF. **Product Finder – Buses**. Available at: <https://www.zf.com/products/en/cv/productfinder/buses.html>. Accessed: October 21, 2024.

APPENDIX A – PYTHON LOGIC FOR TORQUE CONTROL

```
…
# Load Torque and Distance values from the .csv file
def load_torque_data(filename):
    torque_data = []
    with open(filename, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            # Convert distance from feet to meters and store the torque value
            distance_meters = float(row['DistanceCorrect']) * 0.3048
            torque = float(row['Torque']) / 100
            torque_data.append((distance_meters, torque))
    return torque_data

# Load the torque CSV based on distance
torque_data = load_torque_data('C:\\Users\\path_to_the_folder\\torque_input.csv')

# Create the interpolation function for throttle based on distance
distances, torques = zip(*torque_data)
torque_interpolator = interp1d(distances, torques, bounds_error=False, fill_value=(torques[0],
torques[-1]))

# Function to calculate the distance traveled in the simulation
def calculate_distance(location1, location2):
    dx = location2.x - location1.x
    dy = location2.y - location1.y
    dz = location2.z - location1.z
    return math.sqrt(dx**2 + dy**2 + dz**2)

# Main Loop
counter = 0
while counter < len(route):
    counter += 1
    torque_value = float(torque_interpolator(distance_travelled))
…
```

APPENDIX B – PYTHON LOGIC FOR STEERING CONTROL

```
…
# Function to calculate angle between car and target waypoint
def get_angle(car, wp):
    vehicle_pos = car.get_transform()
    car_x = vehicle_pos.location.x
    car_y = vehicle_pos.location.y
    wp_x = wp.transform.location.x
    wp_y = wp.transform.location.y
    # Vector to waypoint
    x = (wp_x - car_x) / ((wp_y - car_y) ** 2 + (wp_x - car_x) ** 2) ** 0.5
    y = (wp_y - car_y) / ((wp_y - car_y) ** 2 + (wp_x - car_x) ** 2) ** 0.5
    # Car vector
    car_vector = vehicle_pos.get_forward_vector()
    degrees = math.degrees(np.arctan2(y, x) - np.arctan2(car_vector.y, car_vector.x))
    # Adjust angles if they are out of range
    return degrees + 360 if degrees < -300 else degrees - 360 if degrees > 300 else degrees

# Function to get a proper angle for steering
def get_proper_angle(car, wp_idx, rte):
    next_angle_list = []
    for i in range(10):
        if wp_idx + i * 3 < len(rte) - 1:
            next_angle_list.append(get_angle(car, rte[wp_idx + i * 3][0]))
    idx = 0
    while idx < len(next_angle_list) - 2 and abs(next_angle_list[idx]) > 40:
        idx += 1
    return wp_idx + idx * 3, next_angle_list[idx]
# Main Loop
counter = 0
while counter < len(route):
    counter += 1
    # Calculate steering angle
    curr_wp, predicted_angle = get_proper_angle(vehicle, curr_wp, route)
    # Apply control to the vehicle
    steer_input = predicted_angle
…
```