Autonomous Driving Simulator Design and Analysis

A Technical Report submitted to the Department of Mechanical and Aerospace Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Marlena Reinhard

Spring, 2024

Capstone Project Team Members

Julia Blackin

Emma Dalkin

Brian Luong

Marlena Reinhard

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature : *Marlena Reinhard* Date: 5/8/2024

1. Introduction

1.1 Importance of Topic

Autonomous driving is one of the world's most recent technological advancements, holding the potential to revolutionize the future of transportation. The concept has been developing since the early 1960s but has made the most significant progress within the last two decades. To further the advancement of integrating autonomous vehicles (AVs) in the real world, it is essential to develop driving simulators and other testing technologies to ensure their safety and reliability. According to the June 2023 Report to Congress regarding autonomous vehicles, the process of passing even just a small feature of the system involves several stages. These stages include test scenarios, metrics, simulation, test track, on-road, framework testing, and preventative maintenance (Report to Congress, 2023). The report indicates that most of the primary and critical testing is done in the simulation and test track phases, therefore highlighting the urgent need of high-level adaptive AV simulators to further the progression of the new technology. In addition to being a crucial aspect in testing processes, AV simulators are exponentially safer than on-road tests due to the lack of physical interaction, allowing for no harm to be done to the operators, the vehicle, and other vehicles on the road. Legislation in many states also prevents the testing of AVs on public roads for this exact reason (Favaró, Eurich, & Rizvi, 2019). In the VICTOR laboratory, this year's capstone team is attempting to improve the existing manually operated driving simulator by incorporating an open-source autonomous driving simulator known as CARLA. The autonomous driving simulator consists of the front section of a Subaru Forester cockpit mounted and controlled by the MOOG actuation platform for the purpose of eventually being able to successfully run tests that ensure the safe implementation and operation of autonomous vehicles on road systems. This initiative is driven by the pursuit of gaining deeper insight into unforeseen situations that might arise as AVs are integrated into real-world road scenarios, ultimately bringing us a step closer to the full implementation of autonomous vehicles.

Figure 1: The University of Iowa NADS-1.

1.2 Review of Existing Technologies

The most robust driving simulator available in recent years is the University of Iowa's National Advanced Driving Simulator (NADS-1), designed by the College of Engineering. The design features a 360° projection onto a 24 foot dome surrounding a real and interchangeable vehicle (*Fig. 1*). The image shows the series of tracks that allow the user to be able to experience the feelings of acceleration and rotation in several directions. The design is also atop a motion platform seen beneath the dome to simulate pitch, yaw, and roll. The design sits on a motion platform atop a series of tracks able to reach speeds of 20 ft/sec. The NADS-1 is equipped with a variety of tools to test driver response in a variety of scenarios including distracted and impaired driving, such as eye and head tracking as part of a driver monitoring system (DMS). The DMS has also been used to test the reaction time of drivers being prompted to return to manual mode from autonomous mode in order to generate models to be used for future autonomous algorithms.

In a study done by the Driving Simulation Association, it was noted that immersion is of particular importance because a lack of immersion can cause drivers to behave differently than they would in a real-world situation (2019). In their study, they exposed some participants to a virtual reality experience of walking to a car before entering the simulator, while other participants went without a virtual experience and were able to see the simulator itself. This was done in order to test a greater suspension of reality with a virtual experience versus subjects that were more exposed to the reality of the simulator. Findings indicated that the immersed participants were more likely to make better risk assessments and were more present. This was measured using a phone call scenario during the driving simulation, physiological monitoring such as stress levels and heart rate, and behavioral observations. The study was able to conclude

that immersion does indeed positively promote driving more realistically in the simulator, making immersion a high priority for driving simulators to be used most effectively.

In another study done by the Polytechnic University of Valencia, testing a design of a system with virtual driving scenarios was performed in which models of real roads were used (2016). This was done using 3D modeling software which lowers the cost and allows for familiar software but requires the size of the environment to be below 100 square kilometers. Participants who had driven on the real-world road tested the simulator and were asked how high the similarity of the environment should be to the real environment. They agreed that it should be 'middle' or 'high' similarity and 95.8% of participants believed that the simulator accomplished these levels of similarity. This was based on factors such as naturalness of driving, familiarity with the road, and the perceived realism of the environment. This study shows the importance of the realism of the virtual environment as a means of replication of real-world scenarios for simulation. Since many real-world roads and scenarios can be complex and require adaptability, the replication of realism is important to make sure that autonomous algorithms are prepared for such scenarios.

1.3 Objectives

The final design for the autonomous driving simulator aims to provide an accurate method of testing autonomy to ensure the safety of autonomous and human-driven vehicles once fully implemented into society. Furthermore, the final design plans to have a new enclosure that will provide a more realistic driving experience. The implementation of software such as CARLA (*Fig. 2*) along with the application of simulation sensors will allow the simulator to be tested in different driving environments and encounter a variety of driving conditions. Additionally, the use of CARLA's simulated sensors provide a wealth of possible future applications. The CARLA software is used in parallel with the Python code to allow for realistic and plausible motion of the simulator. Some minor mechanical fixes to complete the final design are to more securely fasten the simulator to the ground, reconnect the pedals to the

Figure 2: CARLA virtual simulation

floor, and potentially install a different steering wheel that is more representative of a realistic steering system.

Additionally, the simulator has a new immersive enclosure that provides a more realistic driving feel. The simulator is moved to a new location that is less confined and will offer more possibilities for future improvements and additions. For example, in the new space the three wall projector system has a new curved screen that offers a more cohesive field of view. Three projectors are still being used; however, the three projectors are projected onto a single screen rather than multiple. Additionally, a support system made out of wooden trusses is built to attach to the back of the curved screen to make the structure more stable and create a more permanent solution for the projector system. Furthermore, a potential new sound system will be installed to enhance the audio. These two systems together, the projector system and sound system, ultimately allow the driver and any passengers in the simulator to truly feel as if they are driving a car in a real road environment rather than in a lab room.

1.4 Outline

In order to accomplish these objectives, the process can be broken down into three stages:

1) Design and Research
    a) Evaluation of existing progress
    b) Customer needs
    c) Target specifications
    d) Concept generation and selection
    e) Functional analysis of design
    f) Ordering and purchasing parts
2) Development
    a) Repairing and organizing the existing design
    b) Familiarization with software
    c) Evaluation of the durability of the MOOG platform
    d) CARLA implementation
3) Testing
    a) Testing code for the MOOG platform
    b) Troubleshooting

2. Essential Knowledge

2.1 Essential Knowledge of System Components

There are three main components of the ADS system: the MOOG motion system, the cave automatic virtual environment (CAVE), and Assetto Corsa/CARLA. A basic overview of each component will be given in this section.

The MOOG motion system consists of a platform, six electromechanical actuators, and a central processing computer. A diagram is shown below in figure 3.

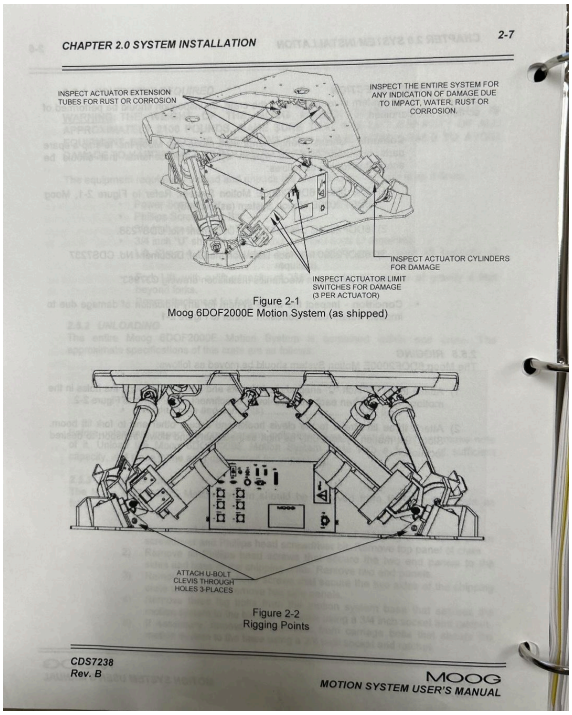


Figure 3: MOOG diagram

The system is capable of accelerating a payload of 2000 kg at 6 m/s^2 in translation and 300 degrees/second in rotation. It receives power from a 240 VAC outlet and has a backup battery in the event of failure of the main power supply. To achieve 6 degrees of freedom, the computer controls the positioning of six electromechanical actuators; by rotating and translating the actuators, the platform can simulate roll, pitch, yaw, heave (forwards/backwards), surge (upwards/downwards), and lateral (left/right). Each of these motions can be combined with others to simulate 3D motion. Motion is controlled by sending the desired motion as a singular value of roll, pitch, yaw, heave, surge, or lateral, or a combination of these, ranging from 0-32767, with each incremental bit representing 0.00049 inches of linear movement. In its initial

position, each actuator rests at 16383 as neutral, with 0 acting as the most negative command and 32767 as the most positive command. The MOOG computer receives these commands in a data frame consisting of 17 bytes, with bytes 3-14 controlling the six actuators. Commands are sent directly to the MOOG computer using USB serial transfer. A Python script was created that takes telemetry data from a virtual simulation environment, such as Assetto Corsa or CARLA, calculates the corresponding values for each degree of freedom and sends the data frame to the MOOG computer at a rate of 60 Hz. A detailed communications flowchart from the MOOG Owner's Manual can be found in the Appendix. A block diagram depicting the circuitry and logic flow used by the MOOG computer is also provided in the Appendix. This effectively allows automatic control of the platform based on motion data from a simulated virtual environment. The motion can be damped using the Python script, to reduce the acceleration due to bumps in the road and upshifting/downshifting the car's transmission. Additionally, an emergency stop button allows for immediate termination of the simulation in the event of a malfunction or unsafe situation.

The CAVE system is used to display the virtual environment in an immersive enclosure. It uses three Optoma short throw projectors with display resolutions of 1080x1920. The projectors are configured to extend the main display, allowing simulation programs to make use of a triple screen display configuration. The virtual environment is projected onto a curved screen, composed of three individual screens connected and positioned around the MOOG motion system. The screens on the left and right of the motion system are supported by wooden walls, allowing easy repositioning of the screens. The center screen is supported by the back wall of the building and curves at the left and right ends to connect to the two side display. Audio is provided with the use of a Yamaha home theater system, consisting of two speakers and one subwoofer with 5:1 channel surround sound capability, and two Radioshack subwoofers connected into the same Yamaha AV receiver. Both the projectors and sound system receive input from the ADS computer running the simulation program.

To implement autonomous driving, the open-source software Car Learning To Act (CARLA) is utilized. CARLA is a very powerful tool used to develop and test autonomous driving software and its subsequent components. The software runs on a server-client architecture, using a Python API to allow multiple Python scripts running concurrently to control various aspects of the simulation server, such as weather, traffic generation, and sensors.

Multiple Python clients controlling different autonomous or manual actors in the virtual environment can be created as well. The autonomous driving software uses a suite of sensors, like LIDAR and ultrasonic, to control the car, effectively simulating real autonomous driving. Developers have also provided multiple maps to simulate different environments, such as small towns or large cities, and a multitude of vehicles, capturing almost every vehicle type (i.e. compact sedan, full-size SUV, motorcycles, etc.). CARLA also supports the ability to create maps, vehicles, and traffic scenarios as well. ROS integration is provided with the use of a ROS bridge, which can be used to allow the software to control the simulator (http://carla.org).

2.2 Essential Knowledge of Past Team Efforts

The existing University of Virginia design was constructed by the 2022 team. Their design was based on researched customer needs including a realistic user environment, the ability to change between automatic and manual mode similar to real autonomous vehicles, a realistic virtual environment that the user can interact with including other moving entities, a bounded and encompassing display, and the ability to mitigate latency between user input and visual and haptic response in the simulator. When choosing the design, the team prioritized a cockpit that resembles the interior of a car to the greatest extent possible and a series of projectors interfacing with the virtual environment creating a CAVE system for the software visual response. This was believed to be the most effective for creating an immersive user experience and creating an advantage over simpler designs using computer monitors.

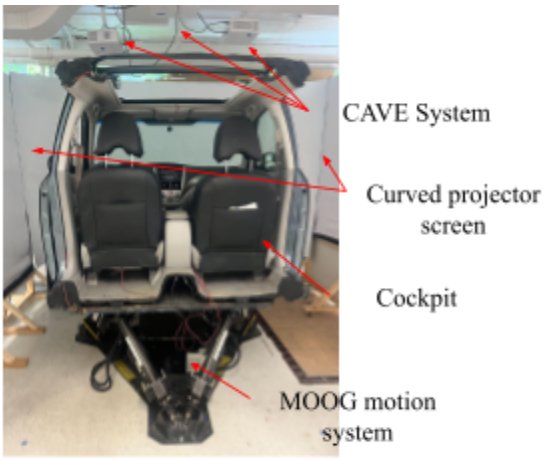


Figure 4: Overview design of system

The 2022 design featured a 2009 Subaru Forester cockpit mounted on a MOOG motion actuation platform with six degrees of freedom. This design allowed for a simple and realistic dashboard display as well as the ability to install a steering wheel, pedals, and gear shift, which the team did using Logitech hardware. The cockpit was situated in the center of the CAVE system, consisting of three short throw projector screens allowing for a 270° field of view (*Fig.*

4). The intended display running off of the computer was connected to the projectors using an HDMI cord and was developed into a CAVE system which allows the projectors to create an immersive three dimensional view of the simulated surroundings. The 2022 team also utilized OpenDS, which is a software used to simulate a virtual driving environment, as well as Gazebo, which is a physics simulation software that communicates with the MOOG platform to simulate the acceleration and motion of the car. Additionally, the team used Assetto Corsa (*Fig. 5*), which is another driving simulator software, using Python code to communicate with the MOOG platform and simulate motion. The MOOG platform was secured to the floor using several wooden beams over a grate.

The 2022 group also ran several tests to determine necessary improvements as well as test the safety of the design. A Python code was written in order to move the MOOG platform to several different extreme positions in quick succession. The positions would not normally be used during a simulation but were used to determine the movement of the platform itself which were found to be insignificant. An area in need of improvement was the frame rate which was tested to be ranging from 8 to 15 frames per second (fps) while the human eye is able to see up to 120 fps, which hinders the immersivity of the display. A newer and stronger graphics card was recommended for future teams. Another recommendation was to use CARLA as a virtual driving environment software as it is able to lower the latency between user input and visual response as well as test more robust autonomous driving algorithms by simulating sensors that would be attached to a real autonomous vehicle.

Figure 5: Assetto Corsa virtual simulation

3. Design Process

3.1 Design Constraints

In order for the previous teams to accomplish their objectives, the project had to be engineered around several significant design constraints. While not ideal, the solutions implemented by the previous teams ensured a relatively safe operating environment for the simulator. Previous teams chose to build the simulator in the corner of the lab space, most

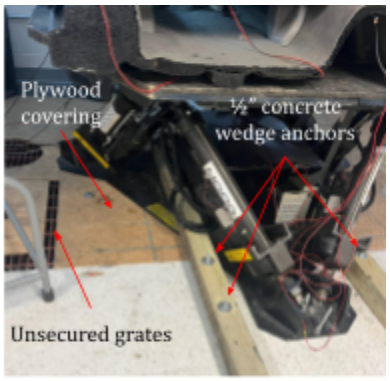


Figure 6: MOOG motion system on metal grate

likely due to the presence of other teams in the area and the large footprint of the motion base. The first major constraint is the placement of the MOOG motion system on a metal grate that was not rigidly connected to the rest of the building (fig. 6). There is a large room underneath the grate, which could have been used for purposes relating to the VICTOR lab's previous designation as a nuclear reactor research facility. The room below the grates terminates 2.5 feet from the wall of the lab space and does not extend past what can be seen in fig. 6. Plywood sheets were used to make the surface level with the rest of the concrete floor. Since the motion base was not placed entirely over the grate, this allowed it to be compressed between the plywood and two 4x4" wooden beams running the length of the motion base. The wooden beams could be fastened to the concrete floor on either end with ½" concrete wedge anchors and nuts. However, while this prevented the simulator from tipping over or from excessive translation, there was noticeable vibration and movement of the motion base during moments of large accelerations, decelerations, or movements of the actuators. The forces and moments from the motion base were so great that it caused the base to momentarily leave the ground in some areas, creating a rocking motion as the simulator settled. This caused the beams to deform and bow upwards. Additionally, the metal housing for the actuators at the front of the base were beginning to damage the wooden beams (fig. 7).

Figure 7: Damage to wooden beams

This presented a safety concern and an area of improvement for this Capstone project.
An additional design constraint was identified for the installation of the CAVE system. The ceiling of the lab space is made of reinforced concrete, with the rebar running the length of the concrete slab approximately 3" from the edge. Two large HVAC ducts and a water pipe also limit

the available space to install projectors. This can result in misaligned and disconnected projector displays. A possible workaround for this problem could be installing the projectors behind the cockpit of the simulator, but this would require purchasing new projectors that are not short throw.

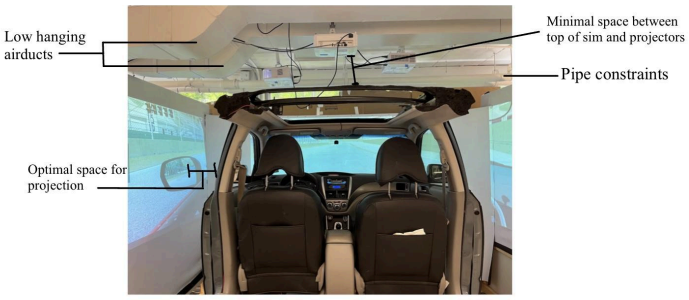


Figure 8: New simulator location space constraints

The concrete floor that was used to bolt the simulator down after its relocation is also reinforced with rebar. In addition, it is 4 inches in depth. As there is no way to know if an area is free from rebar before drilling the bolt holes, the length and type of fastener will need to be taken into consideration to ensure it will have enough holding strength at a reduced depth if a hole is blocked due to rebar.

Another design constraint that the team encountered was a broken hard drive and outdated PC components that were left by the previous team. The broken hard drive contained many of the files and software used by the previous team which were not available this year. The team this year found it immediately necessary to rebuild the PC and replicate the code and software that were used by previous teams.

3.2 Design Process

In order for this project to be beneficial for a large audience,the customer needs were determined. The customer needs include the capability to test the driving simulator in a safe environment, the integration of the CARLA programming framework to simulate various driving environments and conditions, and the ability to evaluate the proficiency of the autonomous algorithms across diverse scenarios for human and machine training purposes. Additionally, the most paramount need is to be able to provide a realistic experience through the use of high quality sensors and graphics working in conjunction with the mechanics of the system. Furthermore, this year's team determined the target specifications necessary to meet these needs and the technical importance of each specification *(Table I)*. The most prioritized target

Table I: Technical Importance of Target Specifications

SPECIFICATIONS:	GPU Processing Speed	Hard Drive Space	Display Resolution	Sound System	Steering Console	CPU Processing Speed	3-D Graphics API	Latency b/w input and simu	CARLA Integration
	1	2	3	4	6	7	8	9	10
	↑	◯	↑	◯	↑	↑	↑	↓	↑
									5
	4					4	4		5
						4		4	5
	2		3	1	4		2	3	3
	2		2			2	2		5
	4		5	2	2		4		
						4		2	5
	4		2		4	5	3	5	
	5		5	2	4	4	3	3	3
	32	0	15	3	12	46	32	29	94
	12.2%	0.0%	5.7%	1.1%	4.6%	17.5%	12.2%	11.0%	35.7%
	3	9	6	8	7	2	3	5	1

specification is the implementation of CARLA programming. Without this, the simulators would not be able to be tested in different driving environments. CARLA is necessary for the examination of how the simulator performs within different driving conditions, environments, terrains, and other varying factors (Dosovitskiy, A. et al, 2017). Another important target specification includes having high CPU (central processing unit) speed. This allows for fast data processing and communication between the computer and the simulator and further enables the simulator to lower the latency between user input and visual feedback. Additionally, high graphics processing unit (GPU) speed and realistic 3D graphics were ranked high among the target specifications, as seen in Table I. A high GPU speed ensures that the projector system is providing clear images along with high frames-per-second as the virtual car moves through an environment. In considering the difficulties the previous team encountered using an outdated GPU, it was decided that a top of the line GPU should be used to surpass the technical specifications and ensure that the hardware can handle future updates to CARLA and other GPU-intensive autonomous driving softwares without becoming obsolete. Lastly, an essential target specification involves low latency between the computer and the simulator. This too ensures that the autonomy of the simulator provides quick driving decisions and reactions based on the driving conditions or a specific computer input. Optimization of the Python and ROS

programs is critical in lowering latency. After analyzing the motion platform's Python code from last year's team, it was decided that all further programming efforts need to be developed using ROS 2 as it is better suited for integration with CARLA, as there is a well-documented procedure for creating a ROS 2 bridge for CARLA on the main website. Additional subscribers will also be developed for the sensors required by the autonomous driving software. The motion control program was replicated in Python for use with ROS 2. After the motion base was successfully controlled manually, CARLA was implemented. Re-testing of CARLA in manual driving mode commenced in order to debug possible errors within the program. Subsequent iterations were carried out interfaced with autonomous driving software and sensors.

Later in the year, a practical visual feel was also determined to be a necessary customer need. Using new hardware, software, and CARLA simulation will ensure that the technology and systems all work together to make decisions; however, a driving simulator loses some of its significance and credibility if it does not provide a realistic feel. Moreover, the collected data and evaluation of the functioning of the simulator is not as reliable if the simulator experience does not feel real or if it gives a false impression of a driving environment. Therefore, creating a new simulator enclosure that provides a more authentic driving experience was established as a vital customer need.

For the concept generation, a morphological analysis chart (*Table II*) highlights the primary functions that will be performed when the simulator is running and shows the components and solutions that will contribute to each of these functions. Subfunctions include technical processes such as software solutions. For example, the simulator receives user input from the steering wheels and brakes, the CARLA software, the computer keyboard, and the video game console. In contrast, the data sensors work in parallel with the Python code, ROS software, CARLA software, and the source code. Moreover, there is a variety of software, hardware, and mechanical components working simultaneously in order to run the simulator. Each additional component of technology eliminates a need for human action, making the simulator more autonomous and increasing its level of automation (Center for Sustainable Systems University of Michigan, 2023). A key difference between this year's concept generation and past teams is that this year's team hopes to incorporate more components that make the driving experience as realistic as possible by potentially installing new tools such as haptic technology, a vibrating seat and steering wheel, and an emergency alert sound system.

Table II: Morphological Analysis of Concept Generation

Sub-functions	Solutions				
Receive User/AV Input	Steering wheel + pedals	CARLA/AV software	Keyboard	Video game controller	
Receive Sensor Data	Python Program	ROS Sub/Msg	CARLA	Source Code	
Physics Calculations and Motion	5 DOF motion base	Motion Tracks	Haptic technology		
Render Virtual Environment	Assetto Corsa	CARLA	OpenDS	Gazebo	Source Code
Visualization of Virtual Environment	Computer Monitor	CAVE system	Virtual Reality		
Feedback to User	Alert Sounds	Vibrating Wheel/Seat	Haptic Vest	Virtual warning systems	
Receive Power	Electrical outlet	Battery			

Lastly, for the concept selection, concept scoring and concept screening is computed to quantitatively and qualitatively prioritize the necessary criteria. For concept scoring, the most critical criteria for the driving simulator from most important to least important includes realistic graphics, motion and haptics, testing capability, cost and efficiency, variety of use, and standards compliance *(Table III)*. Furthermore, these criterias were ranked for different types of simulators such as a desktop simulator, cockpit simulator, autonomous simulator, and a few others *(Table IV)*. Through these ratings, it is found that an autonomous simulator ranked highest. For the

Table III: Ranking of Customer Needs

#	CUSTOMER NEEDS:	
1	The simulator needs to be able to test autonomous vehicles in a safe environment.	4
2	The simulator should use CARLA programming.	5
3	The simulator should be able to be used to evaluate the criteria of autonomous vehicle programming.	5
4	The simulator needs to be able to be used for human and machine training purposes.	3
5	The simulator needs to include a variety of driving scenarios to test autonomous driving algorithms.	3
6	The simulator should be adjustable with clear settings and interface.	2
7	The simulator should simulate sensors of autonomous vehicles realistically.	3
8	The simulator should have minimal delay between user inputs and simulation	3
9	The simulator should be highly realistic to the real world, including graphics and physics.	3

Table IV: Concept Scoring for Concept Selection

	Driving Simulators→		Desktop sim		Cockpit sim		Manual Sim		Autonomous Sin		VR sim		Specialized sim	
#	Selection Criteria ↓	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight	Rating	Weight
1	Realistic Graphics	25%	4	1	4	1	5	1.25	5	1.25	5	1.25	4	1
2	Motion and haptics	25%	0	0	3	0.75	5	1.25	5	1.25	1	0.25	4	1
3	Variety of uses	10%	2	0.2	3	0.3	2	0.2	4	0.4	4	0.4	0	0
4	Cost and efficiency	15%	5	0.75	4	0.6	3	0.45	3	0.45	5	0.75	1	0.15
5	Testing capability	20%	1	0.2	3	0.6	2	0.4	4	0.8	2	0.4	4	0.8
6	Standards compliance	5%	1	0.05	2	0.1	3	0.15	3	0.15	1	0.05	4	0.2
		100%		2.2		3.35		3.7		4.3		3.1		3.15
			RANK	1		4		5		6		2		3

quantitative concept screening this year's team determined which of the criteria for each type of simulator currently meets, does not meet, or surpasses the customer needs. The autonomous simulator is the only option where all criteria are met or surpassed. Although the autonomous simulator meets all of the customer needs, there are still areas with room for improvement or change in order to provide the most realistic driving experience and provide practical testing.

The use of a CAVE system for projection of the virtual environment was also chosen from the morphological analysis because of its advantages in immersiveness and realism. This also aligns with the customer needs that identified that the simulator should be highly realistic in order to replicate real world conditions as closely as possible. Many other driving simulators do not incorporate such a system for projection, so this design choice will be unique and provide a significant advantage over other simulators.

4. Final Design

4.1 System Diagrams

Before providing a comprehensive overview of the system, the following two figures *(Fig. 9 and Fig. 10)* illustrate its key operational components. Figure 9 depicts the internal cockpit of the simulator. When operated manually, the simulator is controlled by the logitech steering wheel, gearshift and

Figure 9: Cockpit of driving simulator components

three pedal system with the virtual environment displayed by the CAVE system projectors allowing for a 270 degree field of view. On the center console, there is a digital display of the gear shift that the car is currently in and a working stereo with bluetooth connectivity along with a hidden keyboard and mouse to operate the software from the inside. Underneath all this, figure 8 displays the wiring of the MOOG platform. The main purpose of the MOOG platform is to operate the six actuators labeled A-F on the diagram. As long as the main power source is plugged in, once the silver battery switch is flipped up and communication is sent to the MOOG base through the arduino connection (*Fig. 11*), the actuators are able to move the simulator in six different directions.

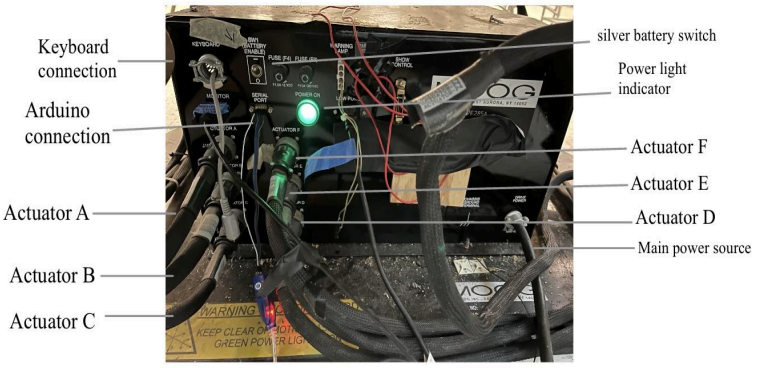


Figure 10: MOOG Base components

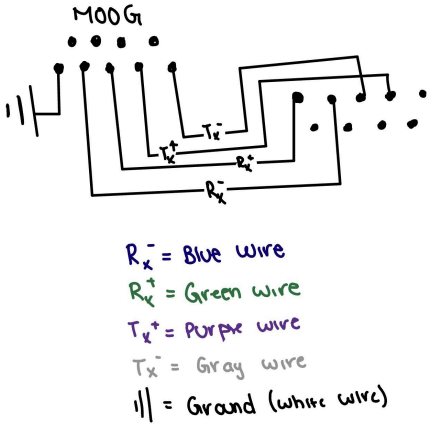


Figure 11: Arduino connection wiring

To fully understand the system and possible subsystems of the simulator, a function decomposition was created to identify the process flow of data through the system (*Fig. 12*). The morphological analysis table created during the concept generation phase identifies the needed sub-functions required during certain stages of the process. Based on the current setup of the simulator, two subsystems were also identified: one that processes and translates data to send (simulator computer), and one that executes the necessary tasks (simulator). The two subsystems are denoted by the dashed lines surrounding their respective subfunctions. Electricity from a wall outlet is received by the entire system to power it. Data inputs that control movement are transmitted to the simulator computer from 1) the user via the steering wheel, pedals, and gear shifter or 2) an autonomous driving software running in parallel on the driving simulator computer using ROS messengers and subscribers. Additional simulated environmental sensor data, such as those communicated by LIDAR which are ultrasonic sensors commonly used on AVs, are also received as inputs. The input data is then used to calculate and translate the physical movement into angles of pitch, yaw,

and roll to be replicated by the motion base using Python code in a ROS package. This program will also allow us to make additional fine tuning adjustments, such as damping. Based on how much movement is required, the virtual environment is then rendered. This environment contains the position of the car in virtual space, map data, and sound information. The motion data and virtual environment data are then sent to the simulator. Motion data is processed by the MOOG

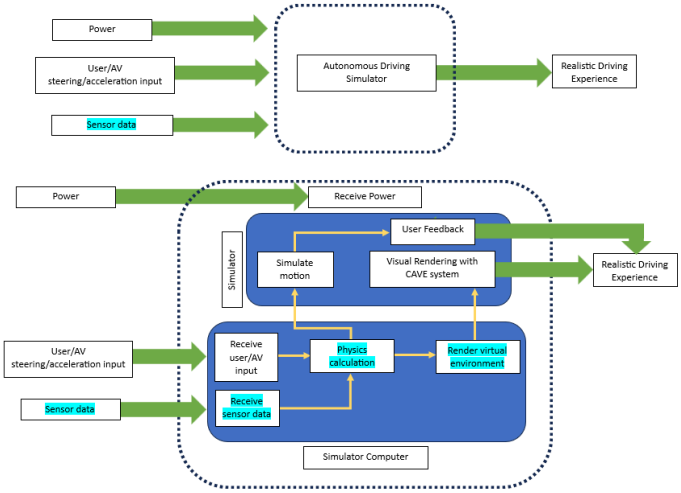


Figure 12: Functional decomposition of system

computer to actuate the motion base and provide user feedback that the action has been completed. The virtual environment data is sent to the CAVE system to be projected in reality. The combined output from motion, projection, and user feedback create a realistic driving experience.

With the subfunctions identified, the system diagram provided in Fig. 13 shows the basic components and connections. As stated before, two subsystems are required: the driving sim PC and the MOOG simulator. The driving sim PC is responsible for running all of the required programs, calculations, and visual rendering for the overall system. In order to receive valid inputs from the steering wheel, pedals, gear shifters, autonomous driving software, or sensors, the simulation programs like CARLA and Assetto Corsa are initialized and run using the PC. All of the program's data, including its inputs, is then sent back to the PC for processing. A separate Python program will translate the velocity, acceleration, position, and orientation data into rotational angles before transmitting it to the simulator. The simulator subsystem is composed of the MOOG computer and the motion base as well as the CAVE projection system. The motion

platform is controlled by the MOOG computer based on the calculated physics data from the driving sim PC. Data is transmitted between the computers using a USB-A connection. The CAVE system projects the virtual environment rendered by CARLA or other simulation programs using the driving simulator PC's GPU. The CAVE's three main projectors are connected to the GPU

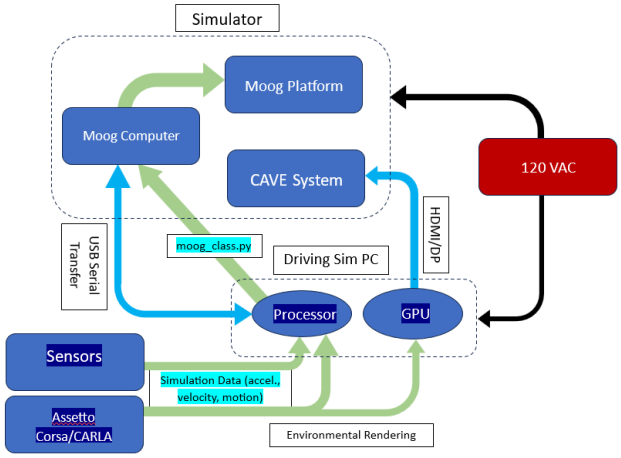


Figure 13: ADS System Diagram

using Display Port cables. Audio information is also sent from the PC to the Yamaha AV receiver and subwoofers surrounding the simulator.

4.2 Designed System: ADS Computer and CARLA

With AC running smoothly and solutions for known faults created, full implementation of CARLA began by adapting the code to ROS 2. This is not a trivial process, as the Python code is not directly transferable to ROS 2, but it will result in a more robust and adaptable program. Integrating the AV sensors also requires the use of ROS messengers and subscribers. Frame rate analysis conducted using AC's built-in feature also suggests that frame rates hover at 60 frames per second with high graphics quality. This is under the desired target specifications.

The new ADS computer was rebuilt with an MSI B760 Motherboard to maximize processing speeds, 128GB of DDR5 RAM to maximize memory speed, a 1200W power supply, and a NVIDIA RTX 4090 GPU to successfully run the CARLA virtual environment. After downloading the correct NVIDIA drivers for the graphics card, performance assessments of the CAVE system and simulator are conducted in Assetto Corsa. Both the ADS computer and the CAVE system are able to render the AC virtual environment at a high graphics settings with the frame rate locked at 60 fps. The MOOG motion system functioned without any serial

communication faults, consistent with performance with the previous computer. CARLA v.0.9.15 was installed following instructions on the official CARLA website for a Linux build using Ubuntu 20.04. Unreal Engine 4.26 containing CARLA-specific patches required for rendering the virtual environment is also installed. It should be noted that downloading the Debian package for CARLA from the command line was not possible as the CARLA servers handling said operations have been offline since 2022. The package was instead retrieved from the GitHub repository. Additional assets, such as vehicles and maps were also downloaded. Following this, the CARLA client library was installed in a Python 3.8 virtual environment. After the assets were downloaded, CARLA was built and a local server was successfully run. Using the example Python scripts included in the package, multiple clients could be established concurrently (fig. 14). Among the most useful are the scripts to generate traffic within the environment and autonomous and manual driving clients.

Figure 14: Running multiple clients on CARLA server

4.3 Designed System: CAVE System and Relocation of Simulator

The simulator has also been given a new location for its setup. To fix the vibration issues, this year's team, with help from UVA Facilities and Maintenance, was able to move the simulator to a new permanent location, allowing for proper installation of securing bolts and opening up the opportunity to redesign the interior of the enclosure.The new location offers a larger space that provides the ability to incorporate features to make the simulator and overall driving experience more realistic. The larger space along with a newly built truss to support the 154" by 86" projector screen allows for three screens to be connected in a curved formation to provide a continuous 270 degree display. A three projector system is used to display the virtual environment on the curved projector screen. There is a projector that displays on the

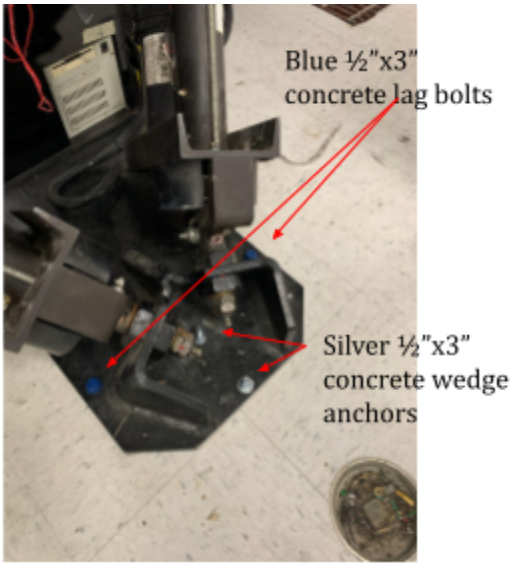


Figure 15: Fastener Configuration for MOOG base


Figure 16: Old Location of Simulator

left, front, and right side of the simulator to create a nearly surrounded environment for the driver. The new location is also over solid ground rather than on steel plates, providing many new benefits for the system. The solid floor reduces noise pollution since it does not shake when the simulator moves. Additionally, the solid floor reduces vibrations from the motion platform when driving the simulator, providing smoother motion when turning or accelerating. Being rigidly connected to the floor, the simulator is much safer and does not risk tipping over. Instead of using concrete wedge anchors that do not directly fix the simulator to the ground, ½" x 3" concrete lag bolts were installed in six bolt holes and ½" x 3" concrete wedge anchors were installed in the other six. While the MOOG Owner's Manual stated that ½" concrete lag bolts be used to fix the base to the ground, two different methods of fastening were required because the diameter of the bolt holes differed. The two outermost bolt holes on each vertex of the base were wide enough to accommodate the 0.6" threaded diameter of the ½" concrete lag bolts, but the two bolt holes on the centerline were too small. Instead, ½" concrete wedge anchors were able to be installed through the smaller holes and secured with a nut and washer. The fastener configuration is shown in figure 15. Figure 16 shows the old location of the simulator over top of steel grates, and Figure 17 reveals the current location and final enclosure over solid flooring.

Figure 17: Current location and final enclosure

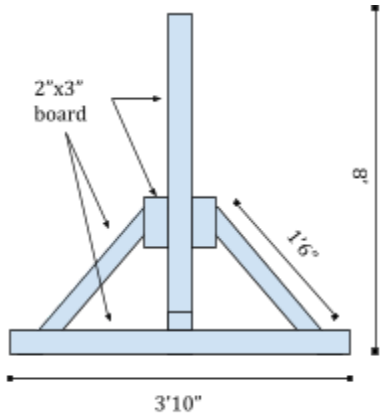


Figure 18: Side View of Screen Wall

The new location offers many benefits for the driving simulator and the overall system; however, it also presents some obstacles. Currently, all three projectors are placed in a position to maximize their display on the screens and maintain an unobstructed view while also avoiding the many ducts and pipes on the ceiling along with the rebar in the

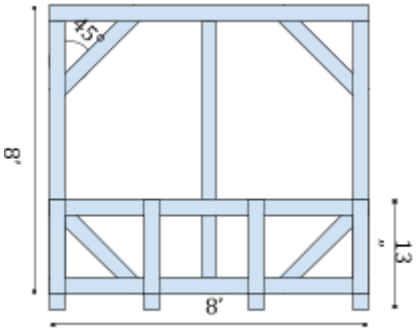


Figure 19: Front View of Screen Wall

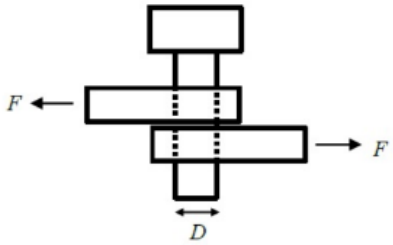
concrete ceiling. Further work needs to be done to properly align the projectors to the screens. Additionally, the simulator is secured into the floor in a position that provides enough space for people to walk and navigate around the system but also avoids the rebar in the ground. Lastly, the screen supports have been built to optimize the visual experience of the simulator but still fit within the new enclosure. A rough diagram of the wall design is included in figures 17 and 18. Wood 2x3" studs were cut to length and used to construct the wall. Zinc galvanized exterior wood screws were used to fasten the wall together.

5. Mathematical Analysis and Experimental Validation

5.1 Mathematical Analysis

The team made the decision to relocate the simulator from its previous position. The simulator was previously placed on a large grate with an empty pit below, making it impossible to secure the platform directly into a floor. The previous teams had used long wooden beams to secure the platform into the floor surrounding the grate by threading the beams through the MOOG platform. This configuration was very undesirable as the vibrations of the simulator were exacerbated by the grate shaking, causing the simulator to move erratically and be very loud. Further, the wood beams would be degraded by rubbing against and inhibiting the movement of the MOOG actuators. Since the MOOG platform would be drilled directly into the floor, a new mechanical numerical analysis was necessary to ensure the safety and longevity of the new position. The MOOG platform moves, so there are two modes of failure that could occur that must be mathematically predicted. These two modes include static failure due to compressive or shear force and fatigue due to cyclic loading.

To analyze static failure, a single shear stress equation can be used. Single shear equations will be used because the bolt will pass through the MOOG base and the floor, as represented in Figure 17. The bolts that are used in this configuration are ½ inch in diameter. The compressive load is assumed to be a maximum force of 3150 lbs which comprises a 2100 lbs MOOG base weight, a 1050 lbs car, and a driver and passenger. The simulator is capable of moving 15 degrees in any direction, creating a shear force of 844 lbs. This shear force is also dispersed onto several bolts. The minimum shear strength of a ½ inch bolt is 1800 lbs, making static failure due to shear force very unlikely, if not nearly impossible. Using the Goodman method to analyze potential bending failure due to fatigue, the bolt cycles through periods of zero shear force and a maximum shear force of 844 lbs. A factor of safety of 74 was calculated



$$\tau = \frac{F}{A} = \frac{F}{\pi r^2}$$

Figure 20: Shear stress diagram of the bolts with diameter D and force F acting on it

with the assumption that the full shear force would be acting on one bolt, indicating infinite life without failure.

5.2 Experimental Validation

The new location of the simulator offers a safer and more secure location for the system but has a few more vertical constraints. The location it resides in allows it to be bolted into the concrete ground instead of resting loosely on top of a hollow grate such as in its old space. However, the lab has air ducts and pipe constraints that stoop lower in the new location, causing a redimensioning of the simulator space. Because of this, an experiment was conducted by testing the maximum movement of the simulator when altering each degree of freedom separately. At its max with each DOF, the position of the simulator was measured to calculate the tipping moment and ensure that the simulator is safe to operate, as displayed in Table V. In addition, a secondary test was performed by gradually increasing the value of input to cause the simulator to move in one direction slowly to ensure that it fits under the constraints discussed above. In Table V, the tipping moment was found to be around 70 kip·in, or 5.83 kip·ft. This

Table V: Tipping Moments of the Simulator

	Center of Mass (in)			Moments (lb in)			
	x	y	z	Mx (lb in)	My	Mz	$M_{g/N}$ (lb in)
Rest	0.00	32.75	-23.06	-72639	0	0	-72,639
Pitch	0.00	30.50	-21.73	-68449.5	0	0	-68,449.5
Roll	-1.10	32.14	-22.66	-71379	0	3465	-67,914
Heave	0.00	35.29	-22.48	-70812	0	0	-70,812
Surge	0.00	32.46	-22.88	-72072	0	0	-72,072
Sway	0.00	31.70	-22.89	-72103.5	0	0	-72,103.5

tipping moment when compared to a force acting at the maximum height of the simulator, thus creating the greatest moment, allows us to determine the maximum force that can act on the simulator. The moment of the center of gravity ($M_{g/N}$) must be greater than the moment of the pushing force (M_{push}), and after determining the moment of the center of gravity, it was determined that the maximum force that could act on the simulator has a magnitude of approximately 452 lbs. In addition, the shear force that the screws create, which was found to be 844 lbs, act in the opposite direction of the pushing force, meaning that a pushing force of 1296 lbs can act on the simulator without tipping it.

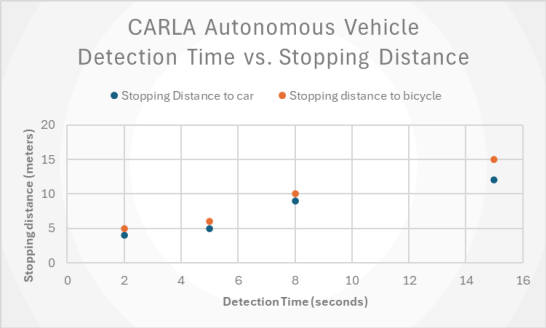


Figure 21: Experimental results from CARLA primary test: Autonomous vehicle detection time and stopping distance from interfering vehicle

To begin testing the autonomous portion of the simulator, two python programs, manual_control_carsim.py and automatic_carsim.py, were used to fabricate both a manually operated vehicle and a self-driving vehicle into the CARLA virtual environment. In each trial, one vehicle was manually controlled to interfere with the autonomous vehicle's path. The manual vehicle was set to drive in front of the autonomous vehicle at various distances from the AV, creating a range of different times that the software had to react to the situation and either stop or continue driving. This experimental setup was meant to replicate situations similar to a car backing out of a hidden driveway very close to the AV, or a car performing an illegal U-turn at a traffic light. The AV was set to an Infiniti QX80, which was the closest vehicle in CARLA's library to the Subaru Forester used on the simulator, and was held constant. The manually driven vehicles were selected to be a Toyota Camry and a bicycle. The Camry represents a majority of passenger vehicles driven by the public, and the bicycle was chosen to test if the software would stop at different distances according to vehicle size similar to human driving. The purpose of this is to observe the capabilities of the autonomous vehicle and its reaction time. Figure 21 shows the results in a graph of the detection time in seconds versus the stopping distance of the autonomous vehicle in meters. It was concluded that at each detection time, the stopping distance was greater for the bicycle than the car. This experiment is one of many that can be carried out with the use of autonomy. Once CARLA and the manual car

simulator are working simultaneously, further experimentation such as real world scenario testing can be carried out.

5.3 Operations Manual

With the current progress on the simulator, the system is able to operate in two parts. The driving simulator can be operated manually using assetto corsa and logitech controls, while the autonomous aspect can be run on CARLA through the ubuntu terminal. The end goal of this project is to be able operate the simulator autonomously in the CARLA virtual environment, but for now they are run separately and therefore have two separate parts of the operations manual which can be found in **Appendix A.**

6. Conclusions and Future work

From the experimental validation and mathematical analysis, the team was able to successfully relocate the simulator and create a much safer and more immersive environment to be able to execute performance assessments of autonomous driving software. Regarding the mechanical improvements to the simulator, a cyclical loading analysis of the bolts used to secure the simulator to the ground indicates infinite life according to the Goodman method. Compared to the previous method of using the wooden beams, this is much safer, with a factor of safety well beyond what is common practice in engineering design. Degradation of the securing components is no longer a concern. For the CAVE system, additional calculations need to be made to find the correct focal distance to position the projector screens on the left and right side of the enclosure in order to correctly align all three displays. Relocation of the projectors will need to take into consideration the location of rebar as well as the HVAC ducts and other overhead components in the area. A performance assessment of the new ADS computer showed large improvements in frame rate as well as latency and responsiveness. Frame rates for CARLA improved from 3-5 fps to the maximum 60 fps, allowing for use of the virtual environment and Python clients without loss of continuity or lag. This will also allow for the effective use of CARLA for performance assessments of AV software since the system is not bottlenecked by the rendering speed of its rendering capabilities.

For the software improvements, the experiment conducted with CARLA's autonomous driving software shows that it consistently reacts in a manner that prevents an accident from occurring in a wide range of dangerous simulated scenarios. Despite varying the size of the accident vehicle, it can be seen that CARLA will stop within at least 4 meters from the point of collision in all

levels of detection time. This also demonstrates how multiple different Python clients can run concurrently within the CARLA server, allowing for a lot of flexibility in experimental design. With the autonomous driving client running in one window, clients controlling traffic, weather, and sensors can allow us to study how the autonomous driver interacts with multiple environmental factors at once. This is one of many different performance assessments and experiments that can be conducted using the CARLA software, demonstrating its usefulness in studying and developing future autonomous driving algorithms.

Future work on this system should focus on integrating the ROS bridge to allow CARLA to control the MOOG motion system. The majority of the software and mechanical work done this semester served as a bridge between the construction of a manual driving simulator and a fully autonomous driving simulator. With the simulator in a more permanent, secure location and the successful installation of CARLA, this will allow future groups to focus more on the software and less on mechanical improvements. These improvements include aligning the projector displays with the curved screen by mounting a custom projector mount behind the simulator and using more responsive, higher resolution gaming projectors and replacing the Logitech components with regular components found in a car, while retaining the ability to interact with AC and CARLA and receive and transmit digital data to the ADS computer. An expansion of the CARLA autonomous driving software could include incorporating a cooperative merge system within CARLA, allowing the simulator to communicate with other virtual vehicles in the simulator in order to merge without collision. This cooperative merge system will have to incorporate sensors and expand upon CARLA's LIDAR capabilities, requiring many adaptations to the initial code, as well as an understanding of how to virtually replicate realistic traffic scenarios.

Appendices

Appendix A - Operations manual

OPERATIONS MANUAL

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

- 1.1 System Overview
- 1.2 Project References
- 1.3 Organization of the Manual

2.0 SYSTEM OPERATIONS OVERVIEW

- 2.1 System Operations
- 2.2 Hardware Operation of simulator
- 2.3 Software Operation of simulator
  - 2.3.1 Software set up of manual driving simulator
  - 2.3.2 software set up of autonomous driving projection
- 2.4 Operation of system
- 2.5 Possible faults
  - 2.5.1 hardware faults
  - 2.5.2 software faults
- 2.6 Safety

3.0 RUN DESCRIPTION

- 3.1 Running system and powering down
- 3.2 Disclaimer

1.0 GENERAL INFORMATION

1.1 System Overview

The Autonomous Driving Simulator (ADS) located in the VICTOR lab is a high functioning system that is operated for the purpose of progressing a manual simulator to become autonomous.

- Major functions performed by the system include realistic manual driving simulator capability and autonomous car simulator 180-degree projection
- The architecture of the system consists of a MOOG base motion system with 6 degrees of freedom including pitch, yaw, roll, heave, surge, lateral.
- To control the actuators, a Python script was developed according to the MOOG User Manual. Documentation of this code is provided in the UVA VICTOR ADS GitHub. The script allows for manual control of the MOOG base as well as automatic control using telemetry data from a simulation program. Example code for manual control is provided in a later section.
- For automatic control, telemetry data from a simulation is translated into rotation and translation using specific values for each degree of freedom. Each value is then sent to the MOOG computer via USB Serial Transfer. The MOOG computer then moves each actuator concurrently to achieve the specified motion.
- To control motion inside the simulation program, a Logitech steering wheel, gearshift, and pedals are used.
- Autonomous driving is accomplished using the Linux Ubuntu 22.04 build of the CARLA software. CARLA makes use of the same Python script used to control the MOOG base, as well as additional ROS 2 messengers and subscribers to send and receive telemetry data.

Operational status:

- Operating and continuing development

1.2 Project References

The system is owned by the VICTOR lab at the University of Virginia and has been operated on by Hudson Burke, Julia Blackin, Emma Dalkin, Brian Luong, and Marlee Reinhard under supervision of professor Tomonari Furkawa.

1.3 Organization of Manual

The manual is divided into several parts to reflect the duality of the system:

1. Hardware set up of manual simulator
2. Software set up of manual simulator
3. Software set up of autonomous projection
4. Running the system
5. Possible faults

2.0 SYSTEM OPERATIONS OVERVIEW

2.1 System Operations

The system is intended to model the behavior of a fully autonomous vehicle within a virtual environment. With the current progress on the simulator, the system is able to operate in two parts. The driving simulator can be operated manually using Assetto Corsa and Logitech controls, while the autonomous aspect can be run on CARLA through the Ubuntu terminal. The end goal of this project is to be able operate the simulator autonomously in the CARLA virtual environment, but for now they are run separately and therefore have unrelated operation manuals.

2.2 Hardware Operation of simulator

- Plug in the MOOG base.
    I. Connect the green plug to the wall outlet located on the front side of the car
    II. Attach the red plug to the power strip situated on the back side of the car
    III. Confirm that all power strip switches are turned on and securely plugged in.
    IV. Once completed, the fan and lights inside the car should activate.
    V. Locate the battery enable switch (a silver lever on the front left of the MOOG base) and turn this switch on to enable the battery.
- Power on the MOOG computer located behind the left wall of the simulator underneath table
    I. Toggle the switch on one side and press the button until the power light illuminates on the other side.
    II. Ensure that the keyboard and mouse are properly plugged into the base of the computer.
    III. Switch on both the MOOG monitor (left) and control monitor (right)
- Enable 270-degree CAVE system projection.
    I. Standing on the seats of the car, hit the power button on the bottom of each of the 3 hanging projectors until the light turns from red to blue.
- Initialize the projection display
    I. If any projector shows an "HDMI not found" error, switch out HDMI to DisplayPort adapters at the base of the computer until the issue is resolved.
    II. Make sure the HDMI cord is plugged into the HDMI 1 port on each projector.
    III. Displays may be misaligned upon initial powerup. This can be remedied in the Display Settings of Windows or Ubuntu by reconfiguring each display's virtual location.
    IV. The order in which each projector is connected to the ports on the ADS computer's GPU matters. Check that the main ADS computer is plugged into the first port and that each projector is connected in the order they should appear on screen.
- Make sure Logitech controls are on: white light illuminated on bottom of steering wheel and green light illuminated on gear shift

2.3 Software operation of simulator

2.3.1 Software set up of manual simulator

- Complete system powerup
    I. After the computer is turned on in the previous step, the monitor should display the GRUB boot menu, select "Windows Boot Manager."
    II. Sign in as the ADS team using the password "ADS2024"

- Set up a manual simulator environment using Steam.
    I. Open Steam (blue application with wrench icon)
    II. Log into the program with username: uvavictorads and password: AutonomousDrivingSim2024.
    III. Go to “your library” and find Assetto Corsa. Open program.
    IV. Once loaded, click main menu-> drive-> select a car-> select a track-> change settings (manual, automatic, damage, etc)
    V. On the Logitech steering wheel, press the green button.
        a. Adjust camera POV using the eye button.
- Set up a manual simulator environment using Assetto Corsa Content Manager. This method assumes you have already logged into the Steam account and all relevant updates have been completed. This is the preferred method, as it is much easier and faster to start the program.
    I. Open the Assetto Corsa Content Manager.
    II. Select desired car and track on the upper left.
    III. You can select the option for practice laps or races with computer controlled racers.
    IV. Select “Drive” in the bottom right corner.
    V. On the Logitech steering wheel, press the green button.
        A. Adjust camera POV using the eye button.
- Connect the moog base to the simulator environment

    I. **Make sure there are no obstructions around the simulator or persons within the marked area, except for the driver and passenger in the car’s cockpit.**
    II. Open VS code terminal and type the following lines:
        - $ Python assetto_moog_control.py
        - If the assetto corsa game screen does not show on the projected screens around the simulator, run a manual code: “from assetto_moog_control import stage_screen”

    III. The MOOG base will initialize, moving to the current initial position of the virtual car in the AC environment.

    IV. Shift out of neutral using the right paddle shifter on the Logitech steering wheel.

    V. The simulator can now be driven using the pedals and the steering wheel.

        - **At any time the simulator malfunctions or seems unsafe, use the Emergency Stop switch in the center console to terminate the simulation. The MOOG base will return to its initial parked position. Remain in the car until the base stops moving.**

2.3.2 Software set up of autonomous driving simulator

This section is intended to demonstrate the future autonomous capabilities of the simulator. Operating the autonomous part of this requires the use of Ubuntu which cannot run simultaneously with windows. Only run the autonomous driving simulator at the end or beginning of the use of the manual simulator due to the lengthy set up times and switch between the two.

Operation of autonomous driving simulator:

- Shut computer and monitors completely to switch from manual to automatic.
- Turn on the computer base and the control monitor.
- Boot the computer to Ubuntu from the GRUB boot menu
    I. Ubuntu should be the first option on the manager list.
- Open Ubuntu terminal
    I. You should see a blank terminal showing only "(base) uvavictorads@uvavictorads"
    II. Change directory to carla-simulator through "cd carla-simulator"
    III. The command line will now look like this (base) uvavictorads@uvavictorads:~/carla-simulator$
- Run CARLA to set up virtual environment
    I. Type "./CarlaUE4.sh" into command line
- Run autonomous driving simulator program
    I. In new Ubuntu terminal type:
        - Conda activate envp38 #to activate python 3.8 virtual environment
        - Cd Carla-simulator
        - Cd PythonAPI
        - Cd examples
        - Run code: python automatic_carsim.py

2.4 Operation of system

- If both hardware and software have been successfully set up, one team member sits in the driver seat, one runs the code lines on MOOG computer, one watches for faults, and all others stand back behind the tape markers on the ground to ensure safety
- Select "start engine" on assetto corsa, run the lines of code in the VS code terminal to begin the simulation
- In the event of a crash or engine failure, the driver must hit the emergency stop button in the center console
- To reset the code after one of these events, in the terminal type "moog.reset()"
- To kill the code on the MOOG computer: ctrl C

2.5 Possible faults

2.5.1 Hardware faults

If the system is not running properly, ensure that all wires are plugged in correctly and the Logitech controls are initialized. The most common hardware fault is incorrect wiring of the USB Serial Transfer cables. A pinout is provided below.

2.5.2 Software faults

On the computer next to the MOOG computer, there is a list of possible faults. In the event of a fault, the error that is occurring within the system will be highlighted in red. List of typical faults and how to fix them:

- Drive bus fault – check that the emergency stop is disengaged and the circuit breaker corresponding with the MOOG base's main power cable has not been tripped..
- Thermal fault, torque monitor, and envelope monitor – turn off the system and let it cool down.
- Bad Batt/home switch circuit fault - check the silver battery switch on the MOOG base.

- COM2 serial communication error - If the Python script is still running in the terminal, press Ctrl+C to terminate the program. Rerun the line “python assetto_moog_control.py” in the terminal.

2.7 Safety

In March 2024, the simulator was moved to its permanent location and bolted into the ground with ½” lag bolts. Operating the simulator can be a bumpy ride but if the actuators are displaced significantly and/or the base of the moog platform becomes not secure in any way, cease use of the simulator until resecured. Refer to the official operation manual for MOOG system (brown binder located on or under the table with the monitors). An example is shown below.

2.0 RUN DESCRIPTION

This system can currently be used to collect data on how vehicles interact with their environment under certain circumstances.

3.1 Running the system and powering down

- Running the system
    I. After all setup is complete, position one person in the driver's seat and one behind the MOOG and control computers. The person behind the computer can hit start on the assetto corsa game as the person in the driver's seat presses the throttle to begin the course. Avoiding collisions, the driver can operate the vehicle at varying acceleration and direction to experience both the sensation and haptics of a real driving scenario.
- Powering down
    I. To safely power down the system, reverse all steps from part 2 and ensure that there are no wires remaining plugged in. Manually turn off projectors, the computer, and the monitors and turn off any switches flipped on.

3.2 Disclaimer

The driving simulator has been operated and tested safely many times. It was built and programmed by students in the MAE department at the University of Virginia under the guidance of Tomonari Furkawa.

Appendix B

Block Diagram for MOOG Computer

1.2.3 COMMUNICATIONS

Communication to and from the Moog control electronics to the System Controller can be done via Ethernet or RS-485 Serial Interface. Both interfaces offer the ride storage and real time options.

1.2.3.1 MOTION CUEING (MDA) (optional)

Moog's motion drive algorithms (MDA) are designed to generate the most accurate motion cue possible within the physical constraints of the motion system. The inputs to the MDA are the vehicle specific forces, angular velocities and accelerations. The output of the motion drive algorithms is the three angular and three linear positions of the motion system (roll, pitch, heave, yaw, surge, sway).

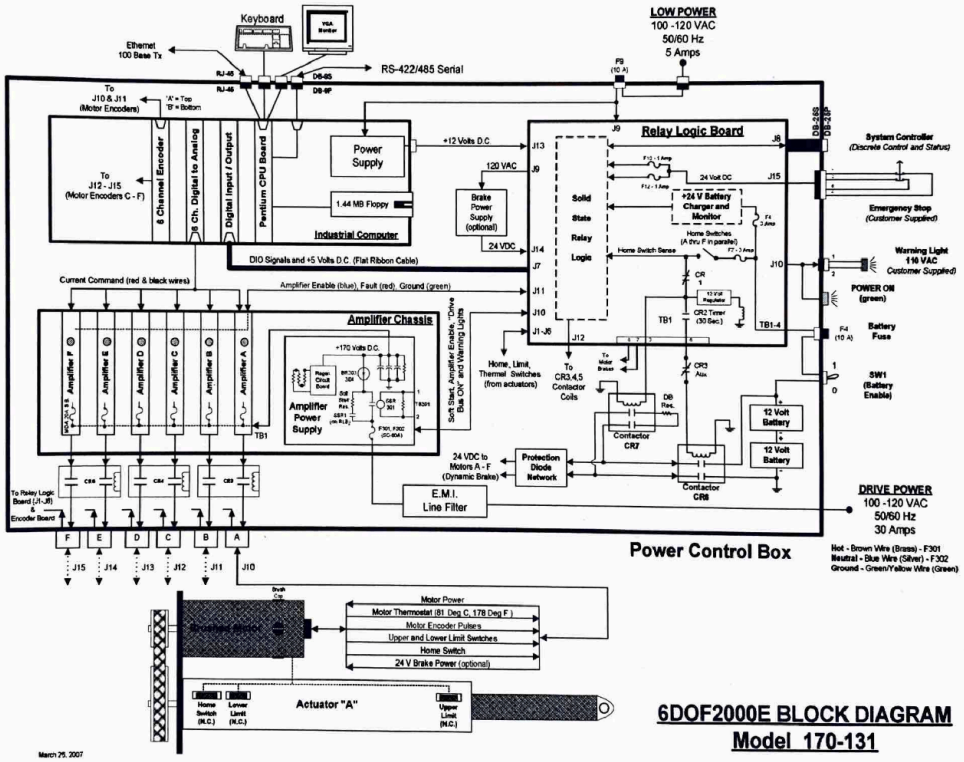


Figure 1-2
Motion Control System Block Diagram

Communications Flowchart for MOOG Computer

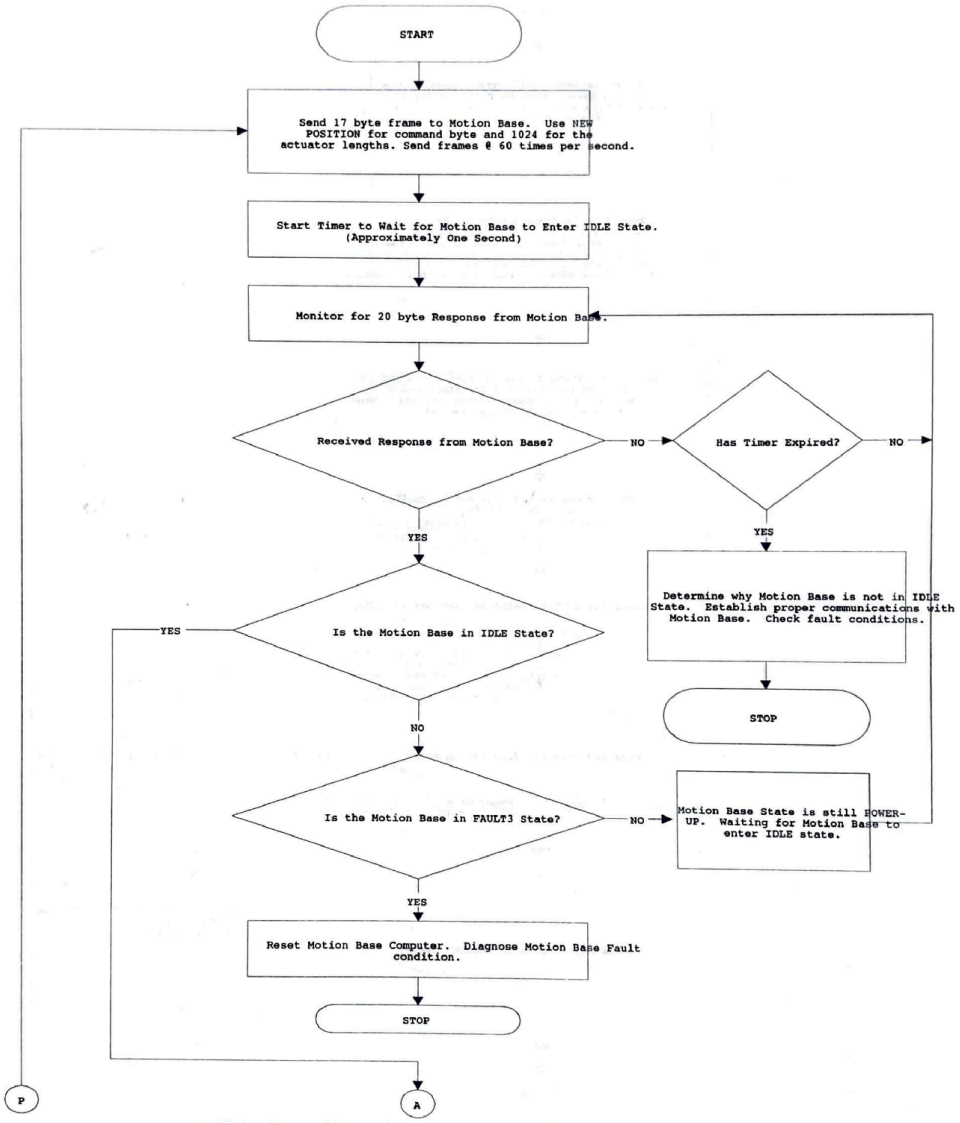


Figure 3-2, Communications Flowchart (Page 1 of 7)

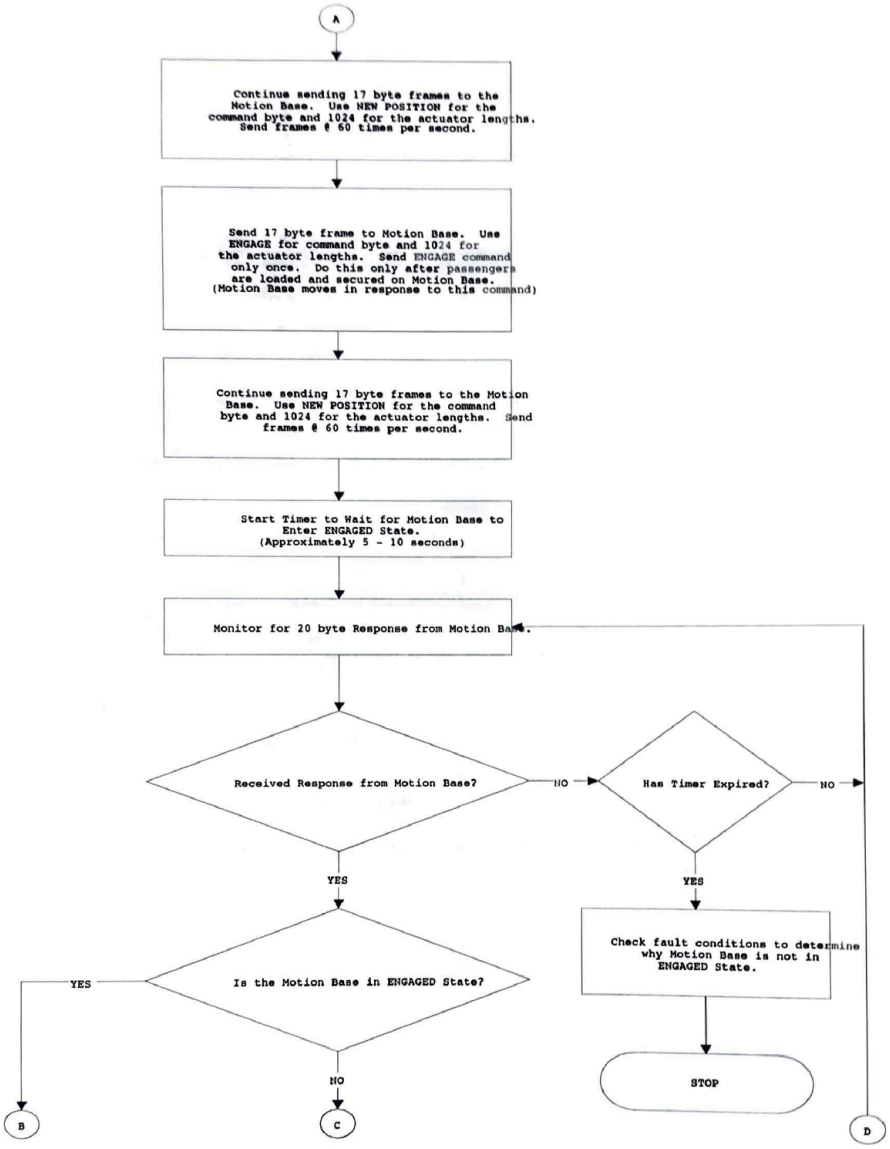


Figure 3-2, Communications Flowchart (Page 2 of 7)

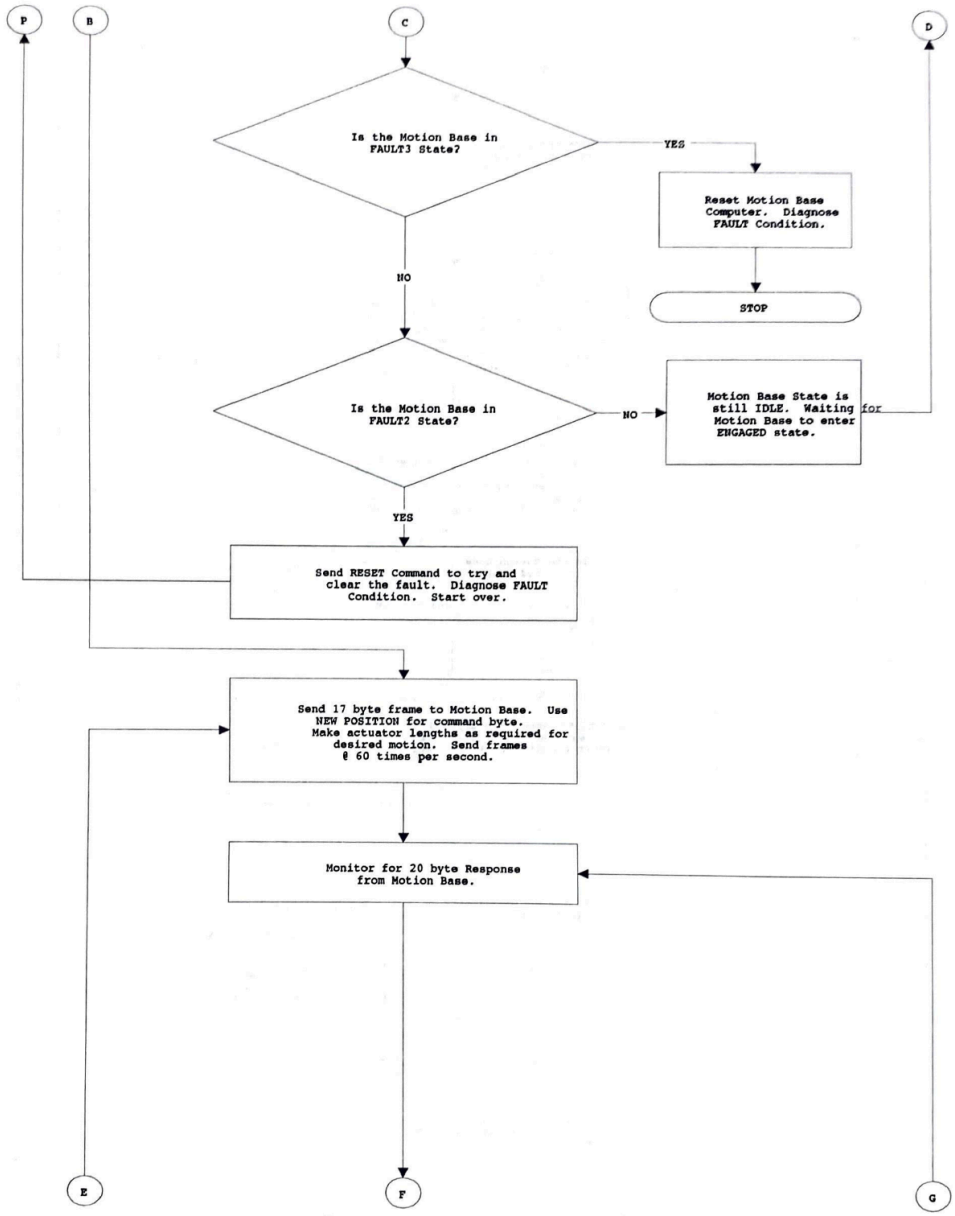


Figure 3-2, Communications Flowchart (Page 3 of 7)

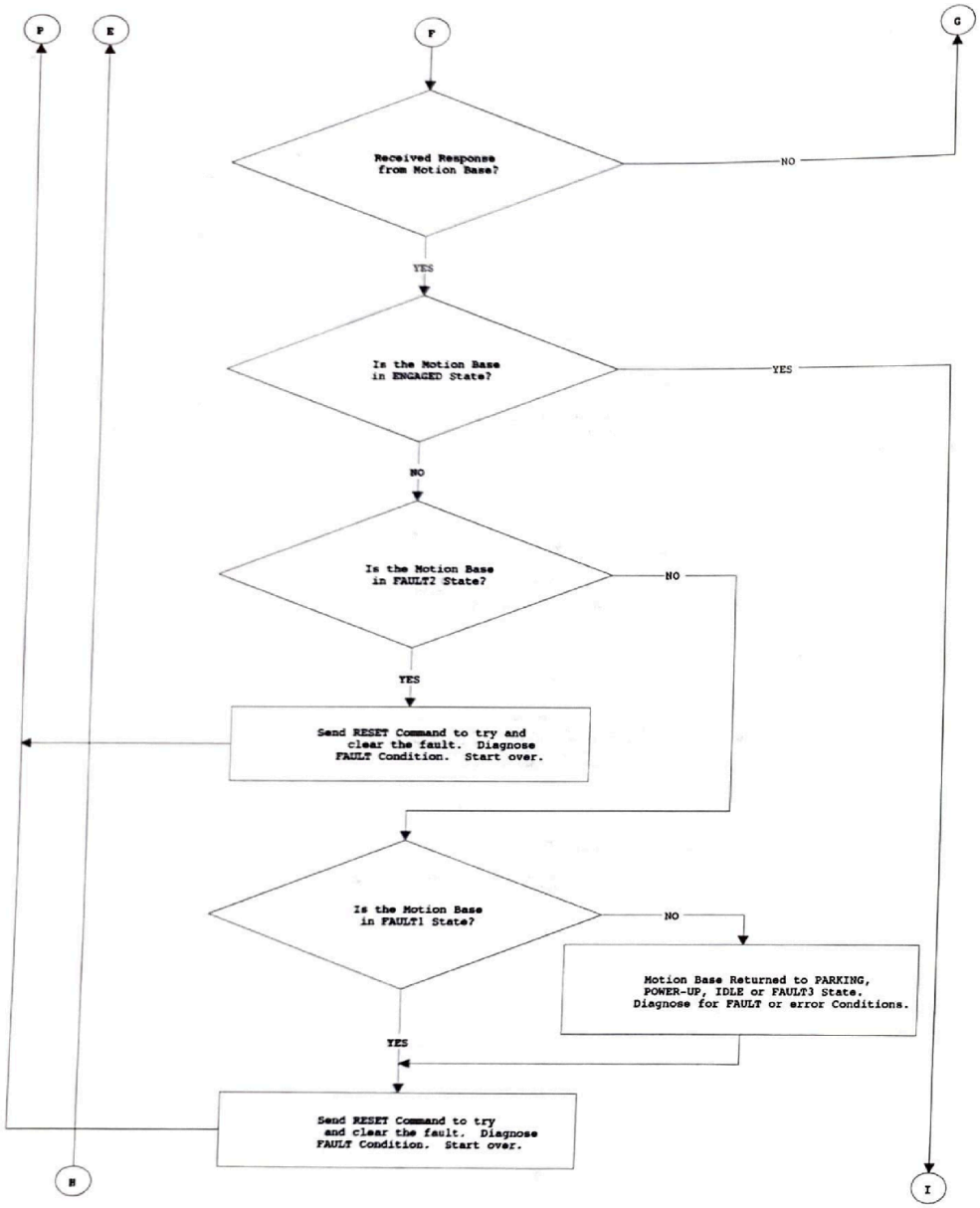


Figure 3-2, Communications Flowchart (Page 4 of 7)

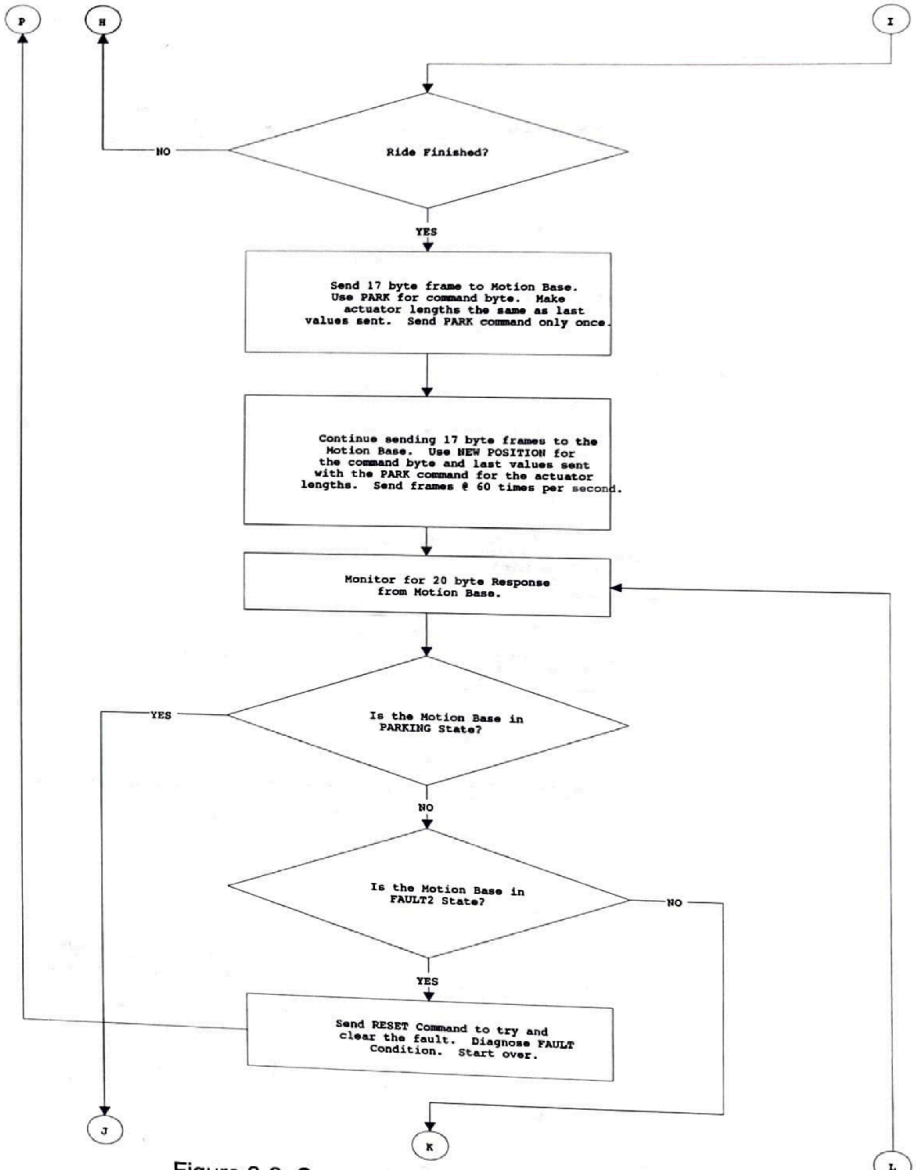


Figure 3-2, Communications Flowchart (Page 5 of 7)

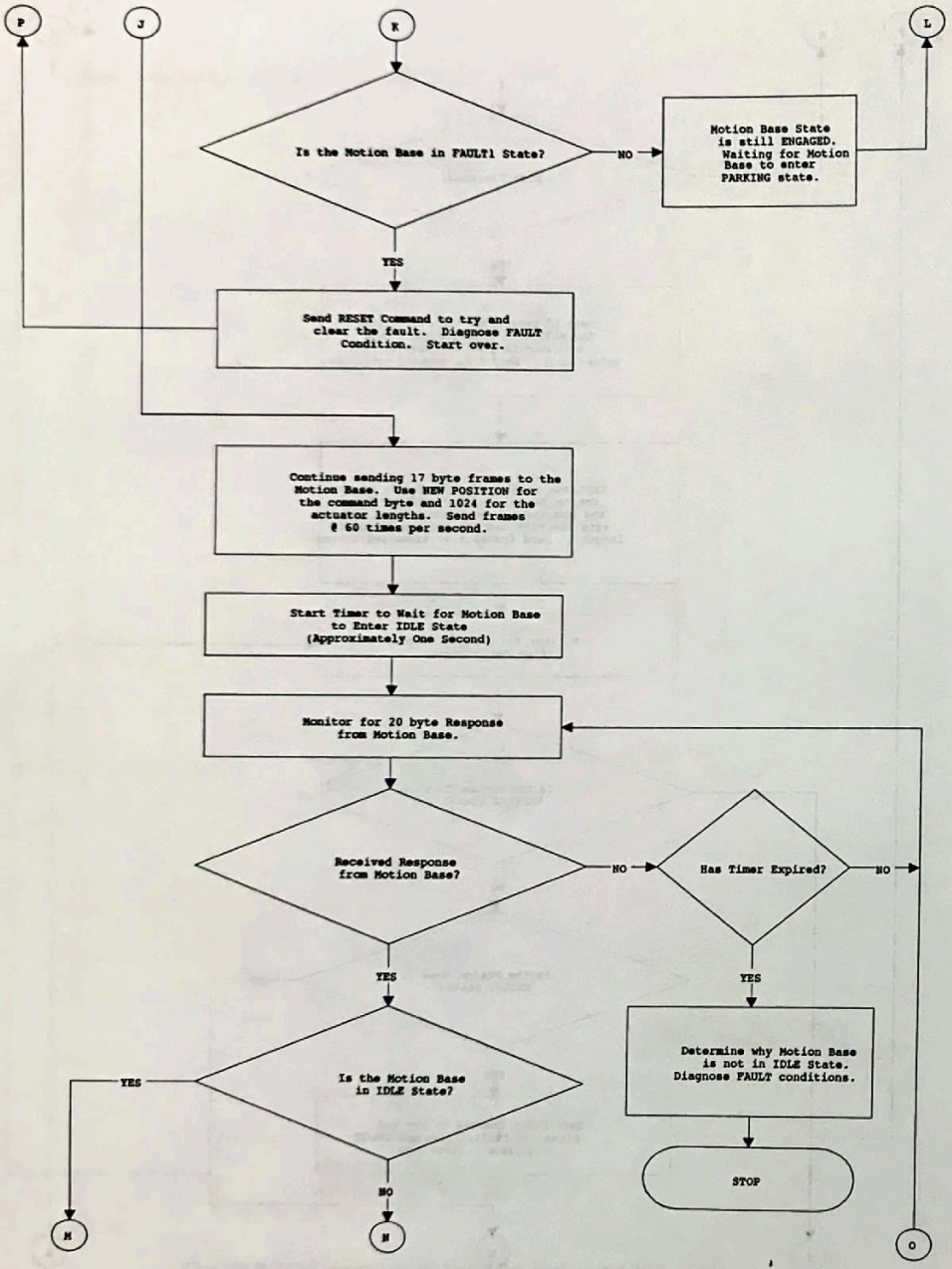


Figure 3-2, Communications Flowchart (Page 6 of 7)

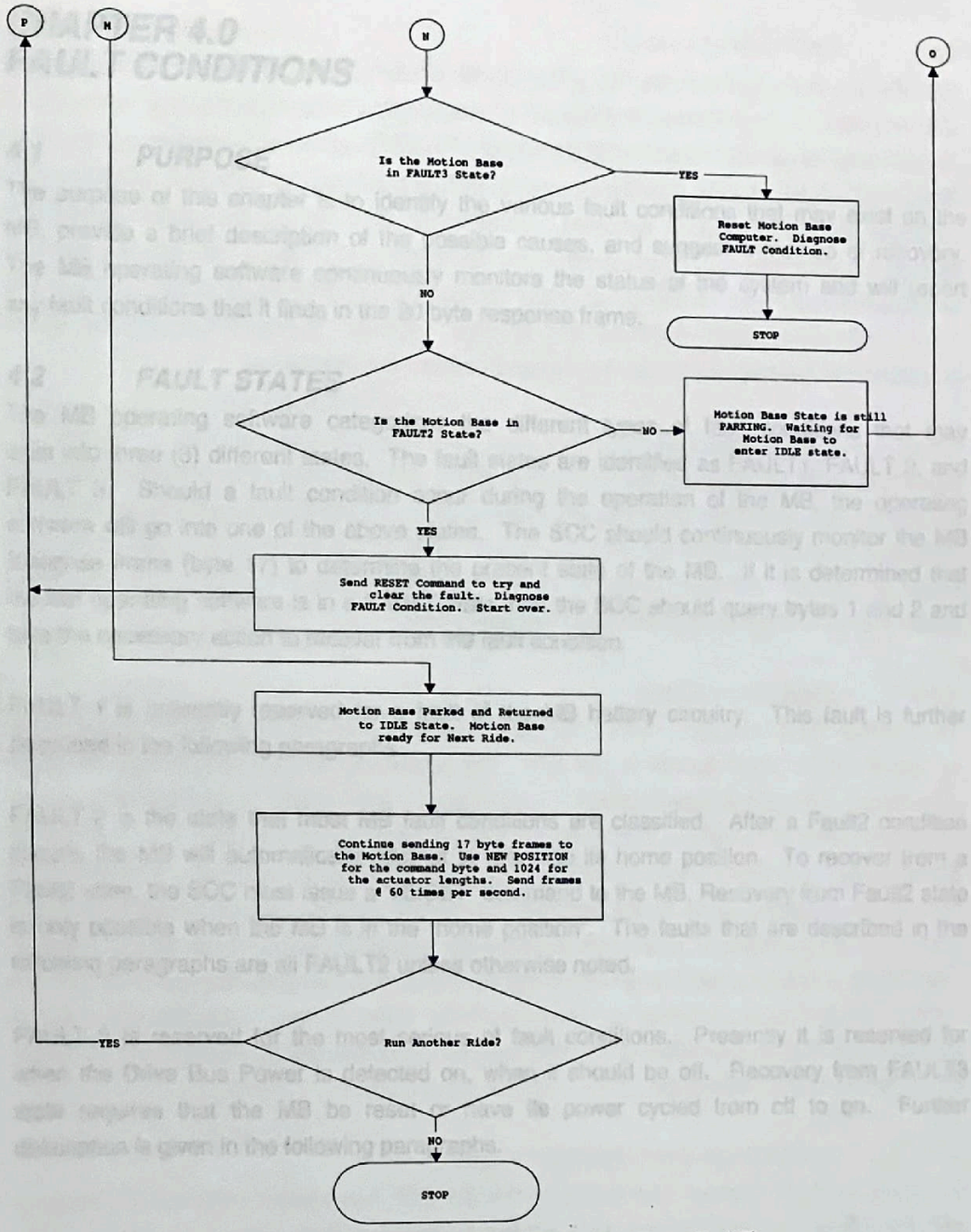


Figure 3-2, Communications Flowchart (Page 7 of 7)

References

Center for Sustainable Systems University of Michigan. (2023). (publication). *Autonomous Vehicle Factsheet*.

Dols, Juan & Molina, Jaime & Camacho Torregrosa, Francisco & Marín-Morales, Javier & Pérez-Zuriaga, Ana & García, Alfredo. (2016). Design and Development of Driving Simulator Scenarios for Road Validation Studies. Transportation Research Procedia. 18. 289-296. 10.1016/j.trpro.2016.12.038.

Dosovitskiy, A., Ross, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). *CARLA: An Open Urban Driving Simulator* (thesis). Computer Vision Center, Barcelona.

Report to Congress - NHTSA. (n.d.). https://www.nhtsa.gov/sites/nhtsa.gov/files/2023-06/Automated-Vehicles-Report-to-Congress-06302023.pdf

Wu, J., Wang, Y., Zhang, Z., Wen, Y., Zhong, L., & Zheng, P. (2022). A Cooperative Merging Control Method for Freeway Ramps in Connected and Autonomous Driving. *Sustainability*, *14*(18), 11120. https://doi.org/10.3390/su141811120

Favaró, F. M., Eurich, S. O., & Rizvi, S. S. (2019). "Human" Problems in Semi-Autonomous Vehicles: Understanding Drivers' Reactions to Off-Nominal Scenarios. *International Journal of Human-Computer Interaction, 35*(11), 956-971. doi:10.1080/10447318.2018.1561784

Ivleva V.; Holzmann S.; Venrooij J. and Zachmann G. Towards Seamless User Experiences in Driving Simulation Studies *In: Proceedings of the Driving Simulation Conference 2019 Europe VR*, *Driving Simulation Association*, Strasbourg, France, 2019, pp. 17-24