Grenze International Journal of Engineering and Technology, January Issue

Matlab-based Modeling of Center Back Steering Wheel Behavior for Development of Custom Driving Simulator

B. Nagashri[1], Dr. S.G Shivaprasad Yadav[2], Vaishnavi Dinesh[3], Karthik Thejendra[4] and Dr. Viswanath Talasila[5]
[1-5]Ramaiah Institute of Technology/Department of Electronics and Telecommunication, Bangalore, India
Email: nagashreeram13@gmail.com,
shivaprasad@msrit.edu,vaishu16koundinya@gmail.com,viswanath.talasila@msrit.edu,karthik.thejendra@gmail.com

***Abstract*—Advanced driving simulators are used in vehicle design, intelligent highway planning, and studies of driver behavior under various conditions. They thoroughly test driver skills to prevent accidents and injuries. This project addresses the need for immersive virtual experiences by creating a driving simulator with custom-designed hardware. It involves modeling center-back steering wheel positioning, illustrating the same using hardware and employs Python scripts to extract telemetry data, providing a realistic and engaging driving experience.**

I. INTRODUCTION

Driving simulators serve a variety of purposes by combining motion platforms, pedals, steering wheels, and visual displays to simulate different driving scenarios:

Driver Training

Simulators replicate conditions like extreme weather, night driving, and heavy traffic to enhance driving skills and simulate real-world challenges. [4]

Research

Using simulators, researchers examine how drivers behave when they are tired, distracted, or impaired by alcohol, for example. [6]

Vehicle Development

ore building physical prototypes, automakers use simulators to test advanced driver-assistance systems (ADAS), evaluate performance, and improve vehicle designs. [1] [2] [3]
Hazard Perception: Using simulators, drivers can improve their reaction times and decision-making abilities by learning how to identify and react to potential risks. [6]

Customization

Based on specific requirements, simulators can be used for a wide range of applications, including emergency response training, military vehicle operation, and professional driver training. [2] [3] [6] Training, research, and entertainment can all be conducted in a safe and controlled environment with the help of driving simulators.

Grenze ID: 01.GIJET.11.1.503_1

II. MOTIVATION/BACKGROUND THEORY

A. Motivation

This project uses advanced software and custom designed hardware in an attempt to accurately replicate real-world driving conditions in an effort to improve realism and immersion in driver education and research. It features entertaining elements such as virtual cities and races to make learning enjoyable. User experience and control are improved by custom hardware designs like realistic steering wheels with feedback systems, while MATLAB integration optimizes software for robust vehicle dynamics modeling.

B. Background Theory

Real – time Telemetry data

Continuous data collection and transmission from a simulated vehicle in a virtual environment is known as real-time telemetry in car simulations. Parameters such as speed, RPM, acceleration, steering angle, tire traction, engine temperature, and fuel consumption are monitored. This data is useful for monitoring vehicle dynamics, providing feedback, evaluating and improving performance. High-frequency data collection results in real-time visualization, logging, and processing that help to fine-tune simulation settings, identify problems, and enhance driver abilities. The following figure 1 shows the telemetry system of Altair.

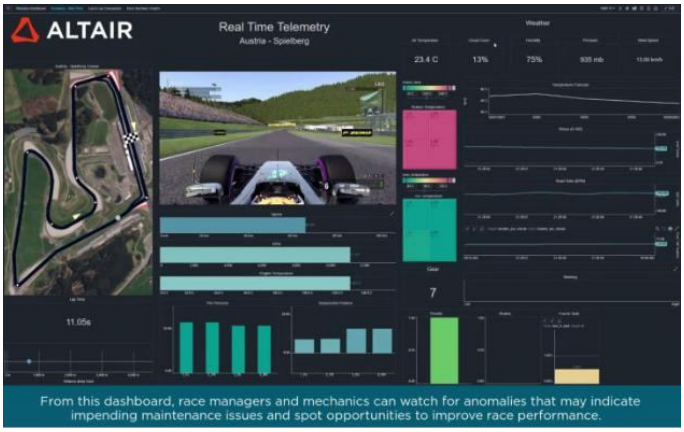


Figure 1: Altair telemetry visualization

Center – back positioning and Force – feedback

When the driver releases the steering wheel, it automatically returns to the neutral position, maintaining stability and control of the vehicle. This is known as the "center-back position" of the wheel. This is accomplished in conventional mechanical systems by means of the steering linkage geometry. Power steering is used in modern cars, and hydraulic or electric systems help to bring the wheel back to the center position.

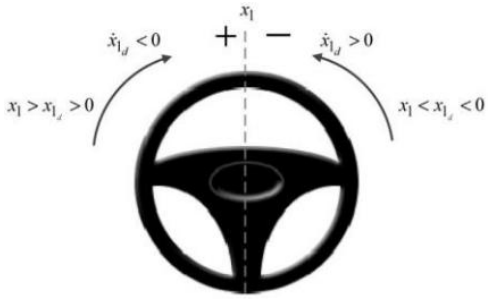


Figure 2: Centre – back behaviour of a steering wheel

Through the generation of resistive forces through the steering wheel or pedals in response to virtual interactions, force feedback in driving simulators simulates real driving sensations. Enhancing realism, this mimics steering resistance, vehicle vibrations, and road texture.

Improved driving skills in a safe environment are made possible by the feedback, which offers prompt, clear cues about driving actions and vehicle responses. Force feedback systems, seen in figure 3 can be adjusted and customized to suit user preferences,simulate specific driving conditions accurately.

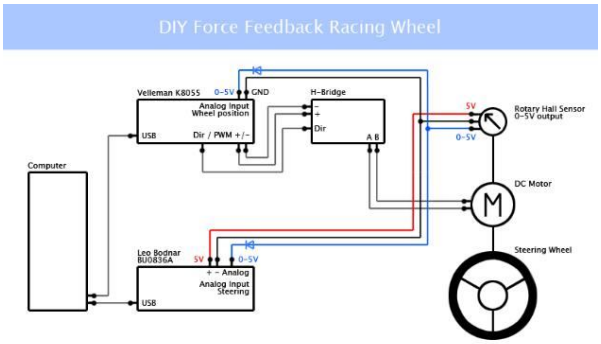


Figure 3: Force – feedback generation for a steering wheel

III. System Design for Modeling the Proposed Driving Simulator

A. Simulation of a rotational mass-spring- damper system with a proportional controller

System Parameters Definition

Define parameters like moment of inertia (J), damping coefficient (b), spring constant (K), and desired position (setpoint). 2. Controller Parameters Definition: Set the proportional gain (Kp) for the proportional controller. 3. Simulation Parameters Definition: Define the time step (timestep) and total simulation time (total_time). 4. Initialization: Initialize arrays to store angular position (theta) and angular velocity (omega) over time. 5. Simulation Loop: For each time step: a) Compute the error (desired position - current position). b) Compute control action (Kp * error). c) Compute angular acceleration (alpha) using the system's equation of motion. d) Update velocity (omega) and position (theta) using Euler's method. 6. Plotting: After the loop, plot angular position and velocity over time.

B. Proportional Controller for a DC Motor

Motor Parameters Definition

Define the DC motor parameters: armature resistance (R), armature inductance (L), motor constant (K), rotor moment of inertia (J), damping coefficient (b), and back EMF constant (Ke). 2. Controller Parameters Definition: Set the proportional gain (Kp). 3. Setpoint Definition: Define the desired angular position (theta_setpoint). 4. Simulation Parameters Definition: Set the time step (timestep) and total simulation time (total_time). 5. Initialization: Initialize arrays for angular position (theta), angular velocity (omega), and applied voltage (V). 6. Simulation Loop: For each time step: Compute error (theta_setpoint - current theta). b) Compute control action (Kp * error). c) Compute back EMF (Ke * omega). d) Compute applied voltage (control action + back EMF). e) Compute torque (applied voltage * K). f) Compute angular acceleration (alpha) using torque, damping, and inertia. g) Update omega and theta using Euler's method. 7. Plotting: Plot angular position, angular velocity, and applied voltage over time.

C. Centre – Back Steering Wheel behaviour in MATLAB

System Parameters Definition

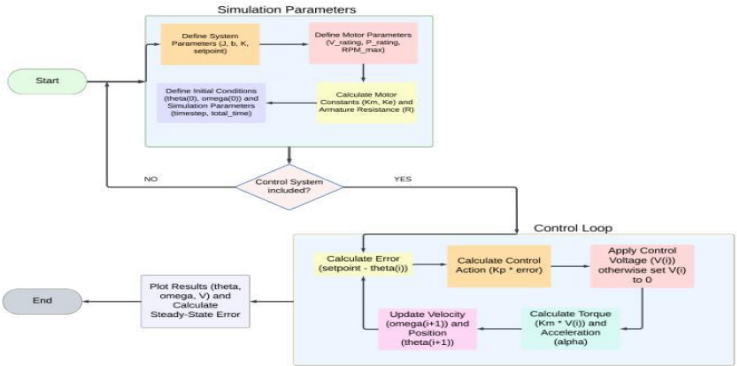


Figure 4: Flowchart of Center-Back positioning of the steering wheel using MATLAB

Define moment of inertia (J), damping coefficient (b), spring constant (K), and desired position (setpoint). 2. DC Motor Parameters Definition: Define voltage rating (V_rating), power rating (P_rating), and maximum speed (RPM_max). Calculate motor constants (Km, Ke, R). 3. Initial Conditions: Set initial position (initial_position) and velocity (initial_velocity). 4. Simulation Parameters Definition: Set time step (timestep) and total simulation time (total_time). 5. Transfer Function Definition: Define the system's transfer function (sys_tf). 6. Plotting Responses: Plot impulse and step responses. Compute step response metrics (settling time, overshoot, rise time). Plot Bode and pole-zero plots. 7. Proportional Controller Definition: Define proportional gain (Kp). 8. Simulation Initialization: Initialize arrays for angular position (theta), angular velocity (omega), and applied voltage (V). 9. Simulation Loop: For each time step: a) Compute error (setpoint - theta). b) Compute control action (Kp * error). c) Compute applied voltage from control action. d) Compute torque using applied voltage and Km. e) Compute angular acceleration (alpha) using torque. f) Update omega and theta using Euler's method. 10. Plotting Results: Plot angular position, angular velocity, and applied voltage over time

D. Circuit Diagram

- Connect Potentiometers: Potentiometer 1 (Accel) to A0. Potentiometer 2 (Brake) to A1. Potentiometer 3 (Clutch) to A2.
- Connect Rotary Encoder: OUT-A to pin 0 and OUT-B to pin 1.
- Connect BTS7960: Short L_EN and R_EN to pin 8. LPWM to pin 9. RPWM to pin 10.
- Power Connections: 5V and Gnd of Arduino to BTS7960, Potentiometers, and Rotary Encoder. B+ and B- to 12V motor power supply. M+ and M- to the DC motor.

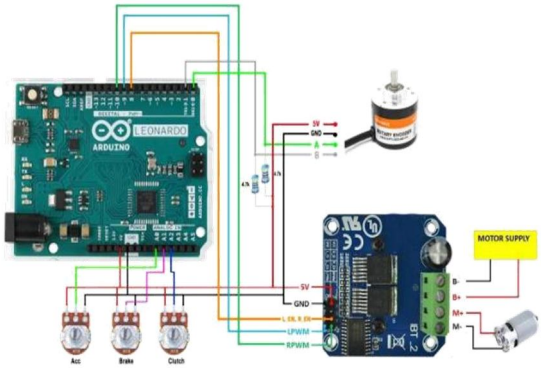


Figure 5: Underlying circuit diagram of the steering wheel and pedals

E. Block Diagram

As seen in figure 6, the simulation or game provides telemetry data through UDP which is fed into the circuitry. The microcontroller and sensors use this data and generate centre-back position and this can be sensed in the steering wheel and pedals setup which hosts the underlying circuitry and in turn we can control the simulation or game. Figure 7 shows a simplified view of the setup.

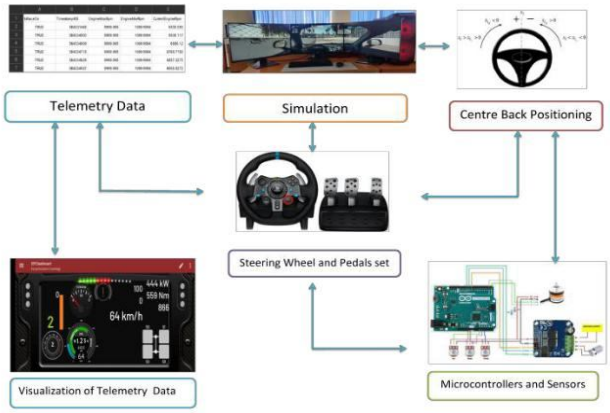


Figure 6: Block diagram of the entire setup

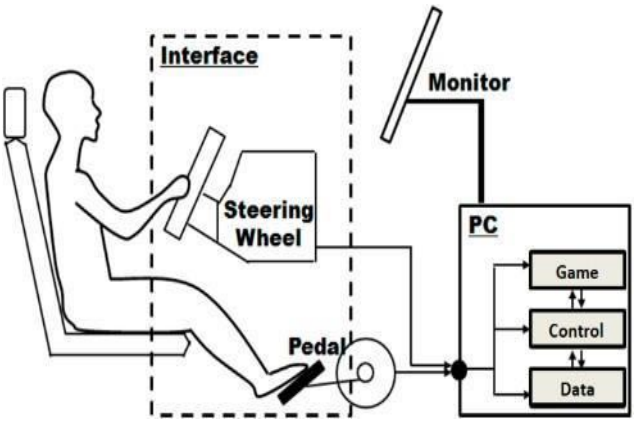


Figure 7: Simplified view of the setup

IV. Implementation of The Driving Simulator Design

A. Algorithm for Telemetry Data Extraction

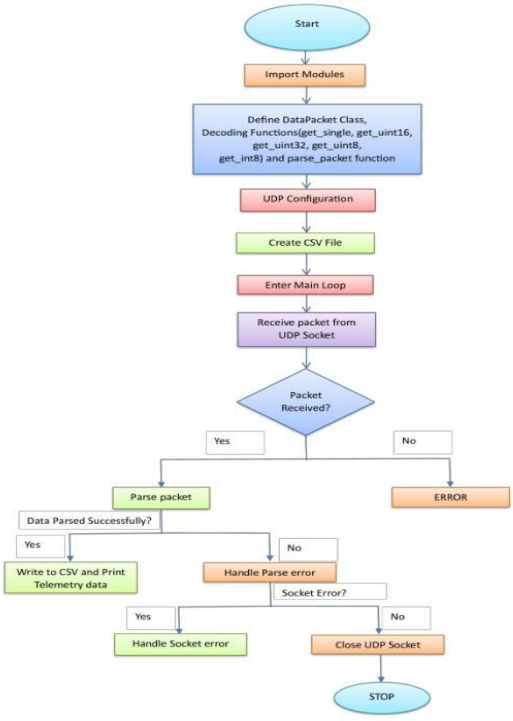


Figure 8: Flowchart of Telemetry Data Extraction

The algorithm used is for receiving and parsing telemetry data from a racing game (specifically Forza Horizon 4) using UDP sockets and storing it in a CSV file. 1. Import Necessary Modules: The code begins by importing required modules such as socket, struct, time, and csv. 2. DataPacket Class: Defines a class DataPacket which will hold the decoded telemetry data. However, it's currently an empty class. 3. Decoding Functions: These functions (get_single, get_uint16, get_uint32, get_uint8, get_int8) are responsible for extracting specific data types from the byte stream received over UDP. They utilize the struct module to unpack binary data. 4. BufferOffset: Defines a constant BufferOffset with a value of 12. This is used to calculate the index for accessing specific fields in the packet. 5. Packet Parsing Function (parse_packet): This function takes the raw packet received from the game, extracts various telemetry data using the decoding functions defined earlier, and populates an instance of the DataPacket class with the decoded data. If any error occurs during parsing, it catches the IndexError exception and prints an error message. 6. Configure UDP Socket: Sets up a UDP socket to

receive telemetry data from Forza Horizon 4. It binds the socket to a specific IP address (127.0.0.1) and port (5300). 7. CSV File Handling: Creates a CSV file (telemetry_data.csv) and writes the header with field names for telemetry data. 8. Main Loop: Enters an infinite loop to continuously receive and process telemetry data packets from the game. Inside the loop: a) It receives a packet from the UDP socket. b) It parses the received packet using the parse_packet function. c) If parsing is successful, it writes the parsed data to the CSV file using the DictWriter object and flushes the file. d) It prints some telemetry data to the console for debugging or monitoring purposes. e) Catches socket.error exceptions and prints an error message if any socket-related errors occur. 9. Socket Cleanup: Finally, when the loop exits (which can happen when the script is terminated), it closes the UDP socket. This algorithm continuously listens for telemetry data from Forza Horizon 4, decodes it, stores it in a CSV file, and prints some of the data to the console for monitoring.

	A	B	C	D	E
1	IsRaceOn	TimestampMS	EngineMaxRpm	EngineIdleRpm	CurrentEngineRpm
2	TRUE	584331468	9999.995	1099.9994	6535.093
3	TRUE	584334500	9999.995	1099.9994	6538.117
4	TRUE	584334609	9999.995	1099.9994	6686.12
5	TRUE	584334718	9999.995	1099.9994	6769.7183
6	TRUE	584334828	9999.995	1099.9994	6857.2275
7	TRUE	584334937	9999.995	1099.9994	6955.9272

Figure 9: Telemetry data logged into a CSV file

Figure 10: Forza Horizon telemetry data visualization

V. Results and Discussion

A. Explanation of the Results with Relevant Plots

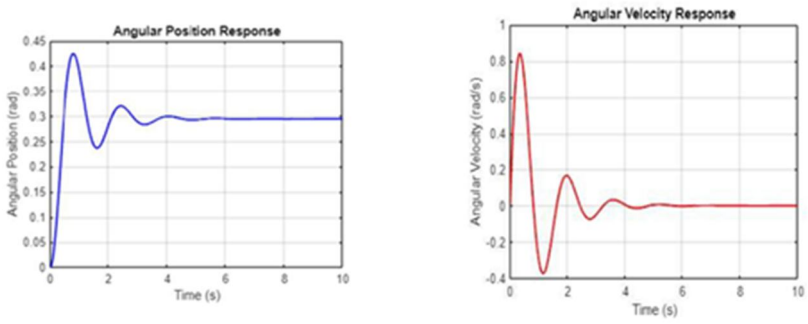


Figure 11: Angular position, Angular velocity response of the Rotational Mass – Spring – Damper system in MATLAB

The above figure 11 describes the variation of the angular velocity and angular position of a Rotational Mass – Spring – Damper system which is controlled by Proportional controller to bring it to a pre-defined set point angle within a particular time frame. This is to verify the parameters of the Rotational Mass – Spring – Damper system and estimate how much time it takes to reach a particular angle.

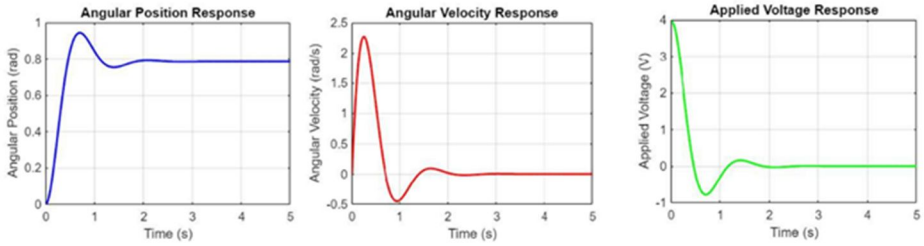
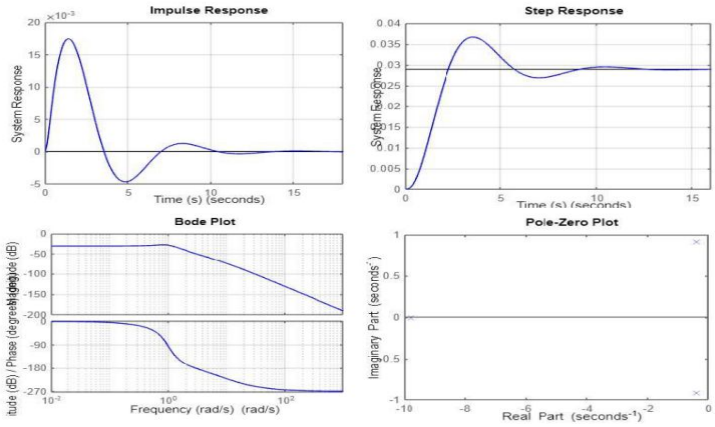


Figure 12: Angular position, Angular velocity and Applied voltage responses of the DC Motor using MATLAB

The above figure 12 describes the variation of the applied voltage, angular velocity and angular position of a DC Motor which is controlled by Proportional controller to bring it to a pre-defined set point angle within a particular time frame. This is to verify the parameters of the DC motor system and estimate how much time it takes to reach a particular angle.



Settling time = 7.8200, overshoot = 26.9307, rise time = 1.4653

Figure 13: Impulse, Step responses, Bode plot and Pole Zero map of the Center back positioning system using P controller

The above figure 13 describes the Impulse Step responses, Bode plot and Pole Zero map of the Center back positioning system using a Proportional controller. The impulse response of the system is plotted to visualize the system's behaviour under an impulse input. The step response of the system is plotted to visualize its response to a step input. Step response metrics such as settling time, overshoot, and rise time are calculated and displayed as settling_time = 7.8200, overshoot = 26.9307, rise_time = 1.4653. The Bode plot of the system is generated to visualize its frequency response. The pole-zero plot of the system is generated to visualize the locations of poles and zeros, which help indicate the stability of the system. Here, the system is stable as the poles are in the negative plane.

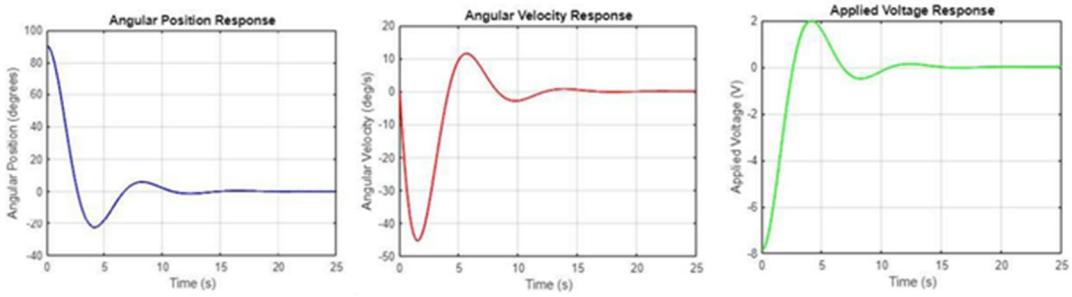


Figure 14: Angular position, Angular velocity, and Applied voltage response of the Center back positioning system using P controller

The above figure 14 describes the variation of the applied voltage, angular velocity, angular position of a Rotational Mass – Spring – Damper system connected to a DC Motor, which is controlled by Proportional controller to bring it to a zero degrees from a particular set angle of say, 90 degrees within a particular time frame. This is to verify the parameters of the behaviour of the system, and estimate the time taken to return to

centre position of 0 degrees. Also, Steady-state error for the angular position response is computed and displayed. The collected telemetry data shown in figure 15 can be analysed to gain insights into various aspects of the vehicle's performance and behaviour during gameplay. This includes examining parameters such as engine RPM, acceleration, velocity, tire temperatures, and suspension travel, among others.

A1 fx CurrentEngineRpm

	A	B	C	D	E	F	G	H	I
1	CurrentEn	Accelerati	Accelerati	Accelerati	VelocityX	VelocityY	VelocityZ	AngularVe	AngularVe
2	2223.002	-0.0509	-1.4149	1.509249	-0.12806	-0.19645	-8.01205	0.109452	-0.01063
3	2136.304	-0.3533	2.071204	1.310483	0.061239	0.159209	-7.78495	-0.10721	-0.00118
4	2075.326	-1.57172	-1.30164	1.081954	-0.10831	0.001844	-7.63483	-0.00568	0.0126
5	2066.714	0.286912	-0.89796	1.30767	-0.02902	-0.07352	-7.50419	0.003928	0.010229
6	2025.335	-0.43123	1.750744	1.681311	-0.0424	0.075828	-7.30944	0.076157	-0.00193
7	1978.358	0.112513	0.512783	1.27399	-0.02708	0.104626	-7.175	0.036391	0.006082
8	1941.696	-0.13472	-1.26394	1.10008	0.00611	-0.11269	-7.04424	0.034512	-0.00251
9	1906.778	-1.12156	0.936414	1.359936	-0.14571	-0.03568	-6.8848	0.01147	-0.01397
10	1857.789	0.953316	1.780971	1.589572	-0.04429	0.214984	-6.69931	0.036282	-0.01075
11	1801.124	0.961225	-2.0138	1.236175	0.036295	-0.0909	-6.58773	0.057353	-0.00963
12	1788.516	-0.28571	-0.79152	1.271856	-0.04021	-0.16059	-6.4714	-0.00661	-0.00936

Figure15: Extracted telemetry data from the simulation which is logged into a CSV file for further analysis

B. Prototype of the custom designed steering wheel and pedals

Figure 16: Pedals setup incorporating potentiometers connected to the underlying circuitry

This project successfully demonstrates the creation of a simulation-controllable custom steering wheel and pedal setup using readily available components.

- Developed accurate MATLAB models for the steering wheel and DC motor for control simulation.
- Implemented proportional controllers for precise speed control and center-back steering behavior.
- Built a functional setup using rotary encoders, DC motors, timing belts, pulleys, and potentiometers to enhance user experience.
- Extracted telemetry data, including engine RPM, vehicle speed, acceleration, and driver input, for analysis
- Various machine learning models can be employed to classify drivers into risky, cautious, and average categories based on their driving behavior. These models utilize telemetry data to identify patterns and characteristics associated with different driving styles.

VI. Conclusion

In the automotive industry, driving simulators are vital because they provide a secure, controlled environment for improving driving abilities, testing new technologies, and carrying out critical research and development. These simulators are now more widely available and cost-effective thanks to improvements in computer performance and cost-cutting measures, which produce lifelike environments that closely resemble actual driving situations. The capabilities of driving simulators are improved by the integration of telemetry data extraction, haptic feedback via DC motors, and specially designed hardware components, which offer a thorough and engaging experience. This project successfully illustrates how to use readily available components to create a custom steering wheel and pedal setup that can be controlled by simulation. Driving simulators play a significant role in driver training, automotive technology advancement, and innovative research because they enable detailed driver classification and performance analysis through accurate mathematical modeling, proportional control for precise speed and steering, and telemetry data extraction.

REFERENCES

[1] M. Damian, B. Shyrokau, X. Carrera Akutain and R. Happee, "Experimental Validation of Torque-Based Control for Realistic Handwheel Haptics in Driving Simulators," in IEEE Transactions on Vehicular Technology, vol. 71, no. 1, pp. 196-209, Jan. 2022, doi: 10.1109/TVT.2021.3125749.

[2] Katzourakis, M. Gerard, E. Holweg and R. Happee, "Design issues for haptic steering force feedback on an automotive simulator," 2009 IEEE International Workshop on Haptic Audio visual Environments and Games, Lecco, Italy, 2009, pp. 1-6, doi: 10.1109/HAVE.2009.5356141

[3] Ravindra, K., Kulkarni Yogesh and Gujrathi Ankit."Comparative Analysis of P, PI, PD, PID Controller for Mass Spring Damper System using Matlab Simulink."(2018).

[4] G. Bouyer, A. Chellali, and A. Lécuyer, "Inducing self-motion sensations in driving simulators using force-feedback and haptic motion," 2017 IEEE Virtual Reality (VR), Los Angeles, CA, USA, 2017.

[5] N. Naajihah Ab Rahman and N. Mat Yahya, "System Identification for a Mathematical Model of DC Motor System," 2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 2022, pp. 30-35, doi: 10.1109/I2CACIS54679.2022.9815461.

[6] Szalay, Zsolt & Gáspár, Péter & Kanya, Zoltan & Nagy, Dávid. (2013). Development of a Vehicle Simulator Based on a Real Car for Research and Education Purposes. doi: 10.1007/978-3-642-33738-3_31.

[7] S. Kaplan, M. A. Guvensan, A. G. Yavuz and Y. Karalurt, "Driver Behavior Analysis for Safe Driving: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 6, pp. 3017-3032, Dec. 2015, doi: 10.1109/TITS.2015.2462084.

[8] Prashanth, G. & Patro, Pratyush & Maram, Devakumar & Velamurali, Dr. (2015). Mathematical Modeling of Vibration in Steering Wheel Assembly of Commerical Vehicles. International Journal of Engineering Research and. V4. 10.17577/IJERTV4IS040973.

[9] Lindow, Friedrich & Kashevnik, Alexey. (2019). Driver Behavior Monitoring Based on Smartphone Sensor Data and Machine Learning Methods. 196-20.10.23919/FRUCT48121.2019.8981511.

[10] Dell'Amico, Mauro & Marzani, Stefano & Minin, Luca & Montanari, Roberto & Tesauri, Francesco & Mariani, Michele & Iani, Cristina & Tango, Fabio. (2007). Design of an Adaptive Feedback Based Steering Wheel. 180-188. 10.1007/978-3-540-73333- 1_23.

[11] Arslan, M.. (2016). Development of a Mathematical Model for the Steering Feel in Steer-by-Wire Systems.

[12] Honeycutt, Craig and John Sushko. ―Haptic Feedback Steering Wheel.‖ (2011).

[13] Modoni, Gianfranco & Sacco, Marco & Terkaj, Walter. (2016). A Telemetry-driven Approach to Simulate Data-intensive Manufacturing Processes.ProcediaCIRP.57.10.1016/j.procir.20 16.11.049.

[14] https://www.marketsandmarkets.com/Market-Reports/driving-simulator-market- 171814690.html