electronics

MDPI

Article

Real-Time Embedded Control of Vehicle Dynamics Using ESP32: A Discrete Nonlinear Approach

Antonio Navarrete Guzmán [1,2], Cuauhtémoc Acosta Lúa [3,4,*], J. A. García-Rodríguez [5,6], Carlos Vidrios-Serrano [2] and Marco A. Meza-Aguilar [6]

[1] Department of Electrical and Electronic Engineering, National Technological of Mexico/Technological Institute of Tepic, Tepic 63175, Mexico; anavarrete@ittepic.edu.mx
[2] Academic Unit of Basic Sciences and Engineering of the Autonomous University of Nayarit, Tepic 63000, Mexico; carlos.vidrios@uan.edu.mx
[3] Department of Technological Sciences, University of Guadalajara, La Cienega University Center, Av. Universidad No. 1115, Col. Lindavista, Ocotlán 47820, Mexico
[4] Center of Excellence DEWS, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy
[5] CUSUR, Department of Computational Sciences and Technological Innovation, University of Guadalajara, Ciudad Guzmán 49000, Mexico; julio.garciar@cusur.udg.mx
[6] Department of Electrical and Electronics, National Technological of Mexico/Technological Institute of Ciudad Guzman, Ciudad Guzmán 49100, Mexico; marco.ma@cdguzman.tecnm.mx
* Correspondence: cuauhtemoc.acosta@cuci.udg.mx or cuauhtemoc.acosta@academicos.udg.mx

Abstract: This article explores the application of the Espressif ESP32 System-On-Chip (SoC) for managing vehicle dynamics through real-time digital proportional–integral (PI-like) control. We present the development of advanced driving assistance algorithms for Active Front Steering (AFS) and Rear Torque Vectoring (RTV) on this cost-effective, commercially available embedded system. Using digital PI-like control algorithms designed for AFS and RTV, the primary ESP32 board receives and processes steering signals, executing a discrete-time control model of the vehicle dynamic to enable dynamic adjustments to steering and torque. To enhance simulation realism, a secondary ESP32 is employed to generate the steering signal, effectively mimicking a steer-by-wire system via its analog output ports. This configuration facilitates the simulation and evaluation of control algorithms in a realistic test environment, ensuring enhanced vehicle dynamic stability and maneuverability under various conditions. Additionally, simulations are conducted using MATLAB 2023a and CarSim 2017.1 to compare the efficacy and benefits of the implementation. Our objective is to establish a platform for evaluating discrete controllers capable of real-time vehicle operation. This methodology accelerates and reduces the cost of improving vehicle system stability and responsiveness, enabling the immediate verification and fine-tuning of control parameters as needed.

Keywords: real-time control; digital control algorithms; vehicle stability; automotive control systems; embedded systems

check for updates

Citation: Guzmán, A.N.; Lúa, C.A.; García-Rodríguez, J.A.; Vidrios-Serrano, C.; Meza-Aguilar, M.A. Real-Time Embedded Control of Vehicle Dynamics Using ESP32: A Discrete Nonlinear Approach. *Electronics* **2024**, *13*, 3967. https://doi.org/10.3390/electronics13193967

Academic Editors: Dariusz Kania, Alexander Barkalov, Remigiusz Wiśniewski and Larysa Titarenko

Received: 15 September 2024
Revised: 3 October 2024
Accepted: 7 October 2024
Published: 9 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The automotive industry is undergoing a profound technological transformation, driven by integrating advanced technologies that enhance vehicle stability, efficiency, and connectivity. A foundational framework for this integration is provided in [1], where the classification of integrated vehicle control architectures supports technologies like Rear Torque Vectoring (RTV) and Active Front Steering (AFS). Further, ref. [2] highlights the complexities of integrating control systems in automated vehicles, emphasizing the role of microcontrollers in managing diverse subsystems efficiently in real-time.

Central to this revolution are microcontrollers, which play a pivotal role in managing a wide array of systems in modern vehicles. These systems include entertainment, navigation, and vehicle stability systems like RTV and AFS. Studies such as [3,4] address

nonlinear control approaches for AFS systems, focusing on maintaining vehicle stability under uncertain dynamic conditions. Their findings highlight the relevance of precise attitude control and real-time correction, enabled by advanced microcontrollers. Similarly, refs. [5,6] explore the application of Steer-by-Wire (SbW) systems and in-wheel motors for electric vehicles, demonstrating how real-time torque distribution can significantly enhance lateral stability. The research in [7,8] further strengthens the case for integrating control systems by comparing different torque vectoring strategies and their impact on hybrid and electric vehicles. These studies provide a deeper understanding of the performance trade-offs in implementing complex vehicle dynamics control. Microcontrollers process data from numerous sensors in real-time, enabling immediate corrective actions that improve vehicle control and safety.

AFS dynamically adjusts the front wheel angle based on the vehicle speed and driving conditions, optimizing the steering response and stability in challenging environments. RTV, by independently distributing torque to each rear wheel, enhances traction and stability during cornering or slippery conditions, thus reducing the risk of skidding. The embedded systems supporting these functions rely on features such as high-precision microcontrollers, integrated sensors, advanced control algorithms, fast actuators, and reliable communication protocols like the CAN bus. These systems must operate reliably in harsh conditions, ensuring vehicle stability and consistent performance.

Despite substantial progress in the development and simulation of advanced control systems for vehicle dynamics, many studies rely on sophisticated computing platforms for analysis and validation. However, these simulations often fail to fully replicate real-world conditions, such as variable weather, mechanical wear, and nonlinear system behaviors. For instance, ref. [9] emphasizes the limitations of simulations by highlighting how discrete-time sliding mode controllers for AFS and RTV are evaluated under simulated conditions using CarSim. These models provide a robust platform for testing under varying parameters and external perturbations but lack the variability of real-world conditions, such as mechanical degradation over time. Similarly, ref. [10] introduces machine learning-based lateral stability control algorithms, demonstrating how advanced methods can fine-tune vehicle control parameters. Although these algorithms perform well in simulations, their adaptability in real-world settings is constrained by the computational limitations of vehicle-embedded systems. The neural network-based inverse optimal control method proposed in [11] shows promise, but its effectiveness in a real-world setting is contingent on the hardware's ability to manage real-time computations efficiently. In [12], energy-saving control observers for electric vehicles are tested through MATLAB-Simulink simulations, yet discrepancies could arise when deployed in real-world environments due to hardware limitations and environmental unpredictability.

Moreover, the use of high-performance computing systems in simulations introduces another challenge when transitioning to vehicle-embedded systems, which are more resource constrained. While field-programmable gate arrays (FPGAs) have been used in other applications, such as deep learning for Pollen Grains Classification [13], custom convolutional neural network accelerators [14], and Vector Accelerator Unit for Caravel [15], adopting such technology in the automotive industry is often cost prohibitive. Hence, validating control algorithms on low-cost embedded systems remains essential.

To bridge the gap between theoretical simulations and practical applications, it is crucial to validate vehicle dynamics control algorithms directly on embedded systems. For instance, ref. [16] discusses the implementation of an embedded data acquisition system designed for analyzing vehicle vertical dynamics, using the Arduino microcontroller platform. This system successfully captures real-time data, proving that microcontroller-based platforms can handle vehicular control applications. Similarly, ref. [17] develops a low-cost data acquisition system using Arduino and accelerometers, achieving high accuracy in vehicle dynamic data acquisition, with a marginal error of 2.19% in road tests. These studies support the viability of low-cost, energy-efficient platforms like the ESP32 for validating automotive control algorithms in real-world environments.

In this research, we propose testing automotive control algorithms on the ESP32 microcontroller board by Espressif. Ref. [18] provides a comparative analysis of low-cost microcontroller modules, highlighting the superior performance of the ESP32 in embedded system design. Its dual-core architecture and wireless connectivity make it well suited for real-time data processing, an essential aspect of advanced vehicle control systems. Similarly, ref. [19] explores the use of the ESP32 for data processing in embedded applications, emphasizing its integration with Internet of Things (IoT) modules and smart sensors, which are vital for handling large data volumes and communicating with higher-level control systems in real-time. The ESP32, known for its robust computational capabilities, features dual cores based on the Xtensa® architecture, capable of 32-bit processing at up to 240 MHz. Its integrated Wi-Fi and Bluetooth connectivity enhances integration with external devices and networks, crucial for real-time data exchange in modern automotive systems. Additionally, its general-purpose input/output (GPIO) pins, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and Universal Asynchronous Receiver–Transmitter (UART) interfaces facilitate connections with various sensors and actuators, which are critical for monitoring and controlling vehicle dynamics. The ESP32's energy-efficient design, along with its ample SRAM and flash memory for processing complex algorithms and sensor data, makes it ideal for real-world automotive applications, ensuring sustained performance under challenging conditions.

The implementation of this real-time system encounters several practical challenges in utilizing two ESP32 boards, one for steering signal simulation and the other for vehicle dynamic modeling. One significant challenge is converting the steering signal from digital to analog using the digital-to-analog converters on the first ESP32 board. Using this approach, we can directly observe and verify a smooth analog signal using an oscilloscope, reducing the need for a PC for signal analysis. Additionally, the hardware limitations of the ESP32 boards present challenges. Although the ESP32 is a powerful microcontroller, its processing capacity and available memory are finite. This sometimes restricts the complexity of the algorithms we could implement, particularly in scenarios requiring high-frequency data sampling or advanced control strategies. These limitations also affect the real-time performance of the control algorithm, as we have to balance the computational load and response time. Furthermore, environmental factors, such as power supply stability and external electromagnetic interference, pose additional challenges that could affect the accuracy of the control signals. These considerations necessitate rigorous testing and calibration to ensure the system maintains reliable performance under various conditions. Overall, while the ESP32 boards provide a viable platform for our project, these challenges highlight the importance of careful system design and optimization to achieve the desired performance and accuracy in real-time control applications. The ability to verify signals using an oscilloscope enhances our capability to troubleshoot and refine the system independently from a PC, ensuring greater reliability and efficiency in the implementation process.

Our approach overcomes the limitations of traditional simulations by directly testing on real embedded systems, utilizing a digital PI-like controller to manage vehicle dynamics effectively. We conduct simulations in MATLAB to compare continuous-time and discrete-time models, ensuring the accuracy of the digital PI-like control algorithm, in addition to real-system testing. Furthermore, we implement the integration between CarSim and MATLAB to simulate a more realistic driving environment. The advanced vehicle modeling capabilities of CarSim allow us to closely replicate real-world driving scenarios, providing a more robust validation of the digital PI-like controller's performance. This integration ensures that the controller operates reliably under realistic conditions, further bridging the gap between theoretical models and practical applications. The simulations conducted in MATLAB and CarSim aim to validate the implementation of the ESP32, demonstrating that its behavior is similar to that of these two simulation scenarios. This highlights the ESP32's capabilities for vehicle dynamics control, enabling real-time responses, which is essential for automotive applications. However, the inherent characteristics of embedded systems and their processing capacity limit the implementation and programming of the

ESP32, despite its notable performance. Nonetheless, its approach aligns with practices used in the automotive industry, facilitating the transition from theoretical solutions to practical applications.

2. Methods

This section outlines the integration of the vehicle dynamic model with a digital PI-like control system, implemented across various simulation platforms. We begin by developing the model using continuous-time dynamics, establishing a solid foundation for understanding its behavior. The continuous-time approach allows us to model the smooth, ongoing changes in the vehicle's state variables, such as position, velocity, and heading angle, over time. Next, we discretize the continuous-time model using the explicit Euler method.

Both implementations are conducted in MATLAB. To ensure accurate behavior, we co-simulate CarSim and MATLAB. CarSim is a software developed by Mechanical Simulation Corporation that specializes in simulating vehicle dynamics. It provides high-fidelity models that accurately represent vehicle behavior under various driving conditions. Automotive engineers widely utilize CarSim for tasks such as vehicle control system development, performance analysis, and stability testing. The software incorporates advanced algorithms and physics-based simulations, offering detailed insights into vehicle handling, stability, and response to driver inputs. MATLAB is a high-level programming language and interactive environment developed by MathWorks, primarily used for numerical computing, data analysis, and algorithm development. It provides a vast array of built-in functions, toolboxes, and visualization capabilities, making it an essential tool for engineers and researchers. MATLAB is particularly valuable in control system design, signal processing, and simulation, allowing users to efficiently prototype and test algorithms. In this setup, CarSim provides advanced vehicle dynamics, while MATLAB manages the implementation of the control algorithm. CarSim's high-fidelity environment enables us to simulate real-world vehicle behavior with precise physics-based dynamics, while MATLAB offers the flexibility needed for developing and refining control algorithms. The co-simulation process involves exchanging data between MATLAB and CarSim at each time step. This ensures that we apply the control inputs computed in MATLAB to the vehicle model in CarSim, and then feed back the resulting vehicle states into MATLAB for further processing. This feedback loop facilitates real-time performance in the simulations, allowing us to observe the control system's responses under realistic driving scenarios. The controller design is based on implementing Active Front Steering (AFS) and Rear Torque Vectoring (RTV), which are methodologies that facilitate vehicle control. To enhance driving performance, we implement control strategies that restrict certain driver actions. We design these controls to align with the vehicle's optimal trajectory, ensuring it operates within its performance limits. We assess the effectiveness of these controls by analyzing the vehicle's motion using the discretized approach of the continuous vehicle model, ensuring that the trajectory adheres to optimal dynamic performance.

A digital PI-like controller serves as the foundation for the digital algorithm's design. Vehicle control systems utilize this particular controller due to its exceptional simplicity, straightforward integration, and minimal computational cost.

It is important to clarify that the intention here is not to verify the validity of the ESP32 board, as numerous examples have previously demonstrated its wide range of capabilities and practicality. The goal is to implement and evaluate automotive algorithms and models and validate their efficiency in embedded systems for real-time testing that verifies control strategies specifically designed for vehicle dynamics. This involves verifying that the implemented algorithms satisfy the criteria of stability, dynamic response, and reliability, thereby showcasing their efficacy and dependability in real-world driving scenarios.

2.1. Continuous-Time Model of the Automobile

The bicycle model simplifies vehicle dynamics by reducing the system to two primary tires: the front and rear wheels. This approach retains essential details while offering an accurate mathematical representation of vehicle dynamics. It effectively evaluates vehicle stability and control under typical conditions by capturing critical longitudinal and lateral movements. To maintain simplicity, the model overlooks vertical movements and rotations (pitch and roll) [20].

Vehicle dynamics are significantly influenced by tire forces and their interactions with the road. The steering angle and tire–road interactions determine the traction forces applied by the front and rear tires. These forces, crucial for stability and control, are characterized by the Pacejka formula [21], which provides a comprehensive analysis of tire behavior.

Assumptions made in the vehicle model include the following:

- The vehicle body maintains symmetry along the longitudinal plane.
- Roll motion is disregarded.
- The directional angles of sliding are kept relatively small.

The angle δ defines the orientation of the front tire, which is a crucial aspect of a vehicle's steering mechanism. Accurately modeling the behavior of the front tire is essential for understanding vehicle steering and stability, as it plays a vital role in controlling direction.

We establish Cartesian coordinates at a reference point C within the inertial X and Y coordinates to analyze the vehicle's motion. The slip angle ψ is then introduced to describe the angle between the direction of the vehicle's velocity vector and the orientation of the vehicle itself.

As is presented in Figure 1, the slip angle ψ is crucial for determining the rotation of the vehicle's local x-axis around the inertial X-axis. Using a small-angle approximation for ψ, we can derive the equations governing the vehicle's longitudinal velocity v_x, lateral velocity v_y, and yaw rate ω_z.

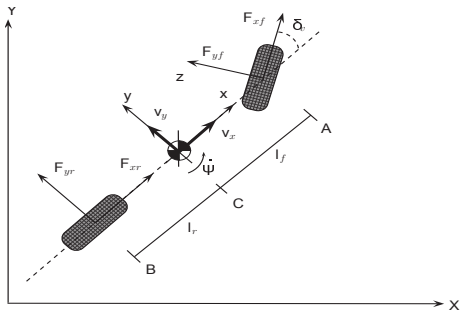


Figure 1. Schematic representation of the bicycle model used in vehicle dynamics modeling.

The longitudinal velocity v_x represents the component of the vehicle's velocity along its local x-axis, while the lateral velocity v_y represents the component along its local y-axis. The yaw rate ω_z describes the rate at which the vehicle rotates around its vertical axis.

The continuous-time mathematical model, considering external forces, is presented as follows [22]:

$$\dot{v}_x = v_y \omega_z + \frac{\mu_x}{m}\left(F_{xf} + F_{xr}\right) \tag{1}$$

$$\dot{v}_y = -v_x \omega_z + \frac{\mu_y}{m}\left(F_{y,f} + F_{y,r}\right) \tag{2}$$

$$\dot{\omega}_z = \frac{\mu_y}{J_z}\left(l_f F_{y,f} - l_r F_{y,r}\right) \tag{3}$$

where m, J_z are the mass and inertia of the vehicle; the lengths of the vehicle's front and rear are denoted as l_f and l_r, respectively; the friction coefficients of the road are μ_x and μ_y;

the lateral forces, frontal and rear, are $F_{y,j}$; and the longitudinal forces, frontal and rear, are $F_{x,j}$, with $j = f, r$.

2.2. Discrete-Time Model of the Automobile

To implement control algorithms on the ESP32, it is essential to transform the vehicle model from a continuous-time to a discrete-time representation. This transformation process, known as discretization, is achieved using the explicit Euler method. The explicit Euler method is chosen due to its simplicity and computational efficiency, making it particularly suitable for embedded systems with limited resources [23,24].

The construction of the ESP32 can then effectively utilize these discrete-time equations. The change is necessary to evaluate the efficacy of control algorithms in real-world conditions, assess the performance of vehicles in practical scenarios, and overcome the limitations associated with computer simulations. The explicit Euler method ensures a rapid and effective implementation. Although it may have limitations in terms of stability and accuracy, especially when dealing with large time steps, choosing an appropriate time interval can still yield sufficient performance for this research. The discrete model yields the following results:

$$\begin{aligned} v_{x,k+1} &= v_{x,k} + T_s v_{y,k}\omega_{z,k} + \mu_x \frac{T_s}{m}\left(F_{xf,k} + F_{xr,k}\right) \\ v_{y,k+1} &= v_{y,k} - T_s v_{x,k}\omega_{z,k} + \mu_y \frac{T_s}{m}\left(F_{y,r,k} + F_{y,f,k}\right) \\ \omega_{z,k+1} &= \omega_{z,k} + \frac{T_s}{J_z}\left(\mu_y\left(l_f F_{y,f,k} - l_r F_{y,r,k}\right) + M_{z,k}\right) \end{aligned} \tag{4}$$

where T_s denotes the sampling period; the subscript k indicates that the quantity is calculated at the sampling instant $t_k = kT_s$, where k is an integer; and $M_{z,k}$ represents the driver assistance controller action for RTV.

The force laterals depend on the slip angles of the front and rear tires:

$$\alpha_{f,k} = \delta_k + \frac{v_{y,k} + l_f\omega_{z,k}}{v_{x,k}}; \qquad \alpha_{r,k} = -\frac{v_{y,k} - l_r\omega_{z,k}}{v_{x,k}}.$$

The proposed controller scenario, which relies on the implementation of Active Front Steering (AFS) and Rear Torque Vectoring (RTV), can effectively simplify the vehicle's model (4).

To optimize performance, it is recommended that the vehicle maintains a consistent and relatively high forward velocity, avoiding braking. The model simplifies control system design by excluding complex vertical and rotational dynamics such as pitch and roll. This approach ensures efficient and accurate management of stability and maneuverability, focusing on lateral speed and skidding. Thus, the model incorporates both AFS and RTV while maintaining a constant longitudinal velocity.

To include the AFS action, which physically adds to the steering angle with $\delta_k = \delta_{c,k} + \delta_{d,k}$, we describe the position of the wheels concerning the road surface, comprised of the sum of the steering $\delta_{d,k}$ and the AFS angle $\delta_{c,k}$. As $\delta_{c,k}$ appears as an argument of the tire characteristic function $F_{y,f,k}$, the design of the control law considers variations in the longitudinal force

$$\Delta_{c,k} = F_{y,f,k} - F_{y0,f,k}; \qquad F_{y0,f,k} = F_{y,f,k}(\alpha_{f0,k}) \tag{5}$$

due to $\delta_{c,k}$, where

$$\alpha_{f0,k} = \delta_{d,k} - \frac{v_{y,k} + l_f\omega_{z,k}}{v_{x,k}}.$$

The controller incorporates the difference in longitudinal force and the lateral force generated by the angle $\delta_{c,k}$ it imposes.

In implementing this methodology and incorporating Equations (4) and (5), and considering the constant longitudinal velocity, the model can be rewritten as

$$\begin{aligned} v_{y,k+1} &= v_{y,k} - T_s v_{x,k}\omega_{z,k} + \mu_y \frac{T_s}{m}\left(F_{y0,f,k} + F_{y,r,k} + \Delta_{c,k}\right) \\ \omega_{z,k+1} &= \omega_{z,k} + \frac{T_s}{J_z}\left(\mu_y\left(l_f\left(F_{y0,f,k} + \Delta_{c,k}\right) - l_r F_{y,r,k}\right) + M_{z,k}\right). \end{aligned} \tag{6}$$

It is important to highlight that after computing a control value $\Delta_{c,k}$, the assumed value for $F_{y,f,k}(\alpha_{f,k})$ at time t_k is set to $F_k^\circ = \Delta_{c,k} + F_{y,f,k}(\alpha_{f0,k})$.

Then, to find the actual control $\delta_{c,k}$, we invert the function $F_{y,f,k}$. This inversion process includes saturating F_k° to the value $F_{f,max}$ to ensure it does not exceed this limit. Here, $F_{f,max} > \text{sign}(F_k^\circ)$, where $\text{sign}(\cdot)$ represents the standard sign function, resulting in $\alpha_{f,k} = F_{y,f}^{-1}(F_{f,max} > sign(F_k^\circ)) = \alpha_{f,max} > sign(F_k^\circ)$. Therefore, given F_k°, the calculation is performed [24]:

$$\delta_{c,k} = \begin{cases} -\delta_{d,k} + \dfrac{v_{y,k} + l_f\omega_{z,k}}{v_{x,k}} + F_{y,f}^{-1}(F_k^\circ) & \text{if } |F_k^\circ| \le F_{y,max} \\ -\delta_{d,k} + \dfrac{v_{y,k} + l_f\omega_{z,k}}{v_x} + \alpha_{f,max} sign(F_k^\circ) & \text{otherwise} \end{cases} \tag{7}$$

The lateral force function is provided by the simplified Pacejka model [21], the Pacejka formula offers a precise and reliable representation of tire performance across various operating circumstances. This facilitates the creation of simulations that are more authentic and realistic. Integrating the Pacejka model into the vehicles' design can greatly enhance their handling and stability:

$$F_{y,j,k}(\alpha_{j,k}) = D_j \sin\left(C_j \arctan(B_j\alpha_{j,k})\right), \quad j = f, r \tag{8}$$

where B_j (stiffness factor), C_j (shape factor), and D_j (peak factor) are the model parameters adjusted using experimental data.

2.3. Formulation of the Control Problem

The objective of this research is to implement and evaluate control algorithms and models for automobile dynamics, verifying their efficiency in embedded systems using the ESP32. The control algorithm employs a digital PI-like controller applied to the vehicle dynamic, operating under the assumption of a constant, high longitudinal velocity. This simplification allows for a mathematical representation that focuses on lateral dynamics and skidding, without the complications associated with varying longitudinal velocity and braking.

The digital PI-like control formulation aims to minimize tracking errors by comparing the actual vehicle's behavior with that of an ideal vehicle that accurately follows the driver's steering wheel signal. The digital PI-like controller adjusts the vehicle's dynamics to align with the desired behavior of the ideal vehicle, utilizing AFS and RTV actions to correct any detected deviations. Implementing the digital PI-like controller on an ESP32 platform leverages its real-time control capabilities, ensuring a high degree of precision and effectiveness in managing vehicle dynamics.

The controls $\Delta_{c,k}$ and $M_{z,k}$ are utilized to impose the desired behavior on the lateral and yaw dynamics. The control of the longitudinal velocity $v_{x,k}$ is not addressed here. Therefore, the model (6) can be rewritten in the following form:

$$\begin{aligned} v_{y,k+1} &= f_{1,k} + g_{11}\Delta_{c,k} \\ \omega_{z,k+1} &= f_{2,k} + g_{12}\Delta_{c,k} + g_{13}M_{z,k} \end{aligned} \tag{9}$$

with

$$f_{1,k} = v_{y,k} - T_s\left(v_{x,k}\omega_{z,k} - \mathcal{F}_y\right) \quad f_{2,k} = \omega_{z,k} + T_s\mathcal{F}_z \quad g_{11} = T_s\frac{\mu_y}{m}$$
$$g_{12} = T_s\frac{l_f\mu_y}{J_z} \quad g_{13} = \frac{T_s}{J_z} \quad \mathcal{F}_y = \frac{\mu_y}{m}\left(F_{y,f0,k} + F_{y,r,k}\right) \quad \mathcal{F}_z = \frac{\mu_y}{J_z}\left(l_f F_{y,f0,k} - l_r F_{y,r,k}\right).$$

The control problem revolves around tracking the desired reference signals $v_{yr,k}$ and $\omega_{zr,k}$, representing lateral and yaw velocities, respectively. These signals are generated based on an ideal vehicle dynamics model, and the references are bounded by their variations.

The tracking errors of the output are given by

$$e_{vy,k} = v_{y,k} - v_{yr,k}, \quad e_{\omega z,k} = \omega_{z,k} - \omega_{z,rk}.$$

Taking one step forward in the error analysis and considering the discrete model (9), the error dynamics can be expressed as

$$\begin{aligned} e_{vy,k+1} &= f_{1,k} + g_{11}\Delta_{c,k} - v_{yr,k+1} \\ e_{\omega z,k+1} &= f_{2,k} + g_{12}\Delta_{c,k} + g_{13}M_{z,k} - \omega_{zr,k+1}. \end{aligned} \tag{10}$$

To mitigate system nonlinearities and ensure zero steady-state error, we propose incorporating a digital PI-like control action. Unlike the traditional PI controller designed primarily in the frequency domain, the digital PI-like controller in this work focuses on rejecting the known system dynamics while adding proportional control to the error.

The digital PI-like controller includes a proportional action, which adjusts the control signal based on the current error between the desired and actual system states. The implementation of the integral action makes the primary difference. Instead of using a continuous-time transfer function approach, the digital PI-like controller introduces integral action through a variable change, which accumulates tracking errors in discrete time. The following equations express this:

$$\begin{aligned} Ie_{vy,k+1} &= Ie_{vy,k} + T_s e_{vy,k} \\ Ie_{\omega z,k+1} &= Ie_{\omega z,k} + T_s e_{\omega z,k} \end{aligned} \tag{11}$$

In these equations, $Ie_{vy,k}$ and $Ie_{\omega z,k}$ represent the integral errors for lateral velocity and yaw rate, respectively, and T_s is the sampling time. This formulation allows for the gradual accumulation of tracking errors, contributing to the correction of steady-state errors without necessitating continuous-time models.

We design the digital PI-like controller primarily to reject the system's known dynamics and handle real-world vehicle nonlinearities. Its main goal is to ensure zero steady-state error while maintaining simplicity and robustness, making it suitable for embedded system implementation with limited computational resources, such as the ESP32.

Methodology for Setting Constants: Instead of using a specific predefined method like Ziegler-Nichols, we determine the control constants through an iterative tuning process. We rely on simulation tools, including MATLAB and CarSim, to model various driving scenarios and vehicle dynamics. We adjust the constants using the following criteria: ensuring the vehicle remains stable during turns and under varying road conditions; responsiveness, optimizing the system's response time to steering inputs; and testing the algorithm under different vehicle dynamic conditions, including variations in speed, tire slip, and external disturbances like road friction.

This iterative approach allows for fine-tuning the proportional and integral gains to achieve the desired control performance while ensuring the controller remains computationally efficient and suitable for real-time execution on the ESP32 platform.

Using the integral errors, the control inputs are designed to ensure that the system follows the desired reference values. The control inputs $\Delta_{c,k}$ and $M_{z,k}$ are given by

$$\begin{aligned} \Delta_{c,k} &= g_{11}^{-1}\left(k_{11}e_{vy,k} + k_{10}Ie_{vy,k} - f_{1,k} + v_{yr,k+1}\right) \\ M_{z,k} &= g_{13}^{-1}(k_{21}e_{\omega z,k} + k_{20}Ie_{\omega z,k} - f_{2,k} - g_{12}\Delta_{c,k} + \omega_{zr,k+1}) \end{aligned} \tag{12}$$

substituting (12) into (10), the dynamics of the error become

$$\begin{aligned} e_{vy,k+1} &= k_{11}e_{vy,k} + k_{10}Ie_{vy,k} \\ e_{\omega z,k+1} &= k_{21}e_{\omega z,k} + k_{20}Ie_{\omega z,k}. \end{aligned} \tag{13}$$

To analyze the stability of our digital PI-like control system, we employ the concept of a Lyapunov function. The Lyapunov function V_k is designed to capture the state of the system, specifically, the tracking errors and integral errors. The change in this function over a time step, referred to as the Lyapunov increment, is given by the expression

$$\Delta V_k = V_{k+1} - V_k$$

This increment provides insight into the system's stability: if $\Delta V_k < 0$, it indicates that the system's errors are decreasing over time, suggesting asymptotic stability.

To prove the stability of the digital PI-like control, we consider the following Lyapunov function:

$$V_k = |e_{vy,k}| + |e_{\omega z,k}| + |I_{evy,k}| + |I_{e\omega z,k}|. \tag{14}$$

We compute the Lyapunov increment, which represents the change in the Lyapunov function V_k over a time step in a discrete system. Specifically,

$$\begin{aligned} \Delta V_k &= V_{k+1} - V_k \\ &= |e_{vy,k+1}| + |e_{\omega z,k+1}| + |I_{evy,k+1}| + |I_{e\omega z,k+1}| - |e_{vy,k}| - |e_{\omega z,k}| \\ &- |I_{evy,k}| - |I_{e\omega z,k}|. \end{aligned} \tag{15}$$

Substituting in (16), the dynamics of the error (13), integral actions (11), and controller actions (12), we obtain

$$\Delta V_k = |k_{11}e_{vy,k} + k_{10}Ie_{vy,k}| + |k_{21}e_{\omega z,k} + k_{20}Ie_{\omega z,k}| \tag{16}$$

$$+ |Ie_{vy,k} + T_s e_{vy,k}| + |Ie_{\omega z,k} + T_s e_{\omega z,k}| - |e_{vy,k}| - |e_{\omega z,k}| - |I_{evy,k}| - |I_{e\omega z,k}| \tag{17}$$

and using the triangle inequality, we have

$$\Delta V_k \leq |k_{11}e_{vy,k}| + |k_{10}Ie_{vy,k}| + |k_{21}e_{\omega z,k}| + |k_{20}Ie_{\omega z,k}| \tag{18}$$

$$+ |T_s e_{vy,k}| + |T_s e_{\omega z,k}| - |e_{vy,k}| - |e_{\omega z,k}| \tag{19}$$

considering $\gamma_1 = (k_{11} + T_s - 1)$, $\gamma_2 = (k_{21} + T_s - 1)$, $\gamma_3 = k_{10}$ and $\gamma_4 = k_{20}$, we obtain

$$\Delta V_k \leq \gamma_1|e_{vy,k}| + \gamma_2|e_{\omega z,k}| + \gamma_3|Ie_{vy,k}| + \gamma_4|Ie_{\omega z,k}|. \tag{20}$$

Therefore, the errors are asymptotically stable when $\gamma_i < 0, i = 1, \ldots 4$.

Representing the increment ΔV_k, the expression (20) is reduced to the following inequalities:

$$|e_{vy,k+1}| \leq (\gamma_1 + 1)|e_{vy,k}| \tag{21}$$

$$|e_{\omega z,k+1}| \leq (\gamma_2 + 1)|e_{\omega z,k}| \tag{22}$$

$$|I_{evy,k+1}| \leq (\gamma_3 + 1)|Ie_{vy,k}| \tag{23}$$

$$|I_{e\omega z,k+1}| \leq (\gamma_4 + 1)|Ie_{\omega z,k}| \tag{24}$$

If $\gamma_i + 1 < 1$, this ensures that $e_{vyk}, e_{\omega zk}$, $I_{evy,k}$ and $Ie_{\omega z,k} \to 0$.

3. Implementation of Control Algorithms on the ESP32

We construct the simulation using CarSim version 2017.1 and MATLAB version R2023a to evaluate the performance of the intended control algorithm under real-time conditions on the ESP32 platform, which we program using the Arduino IDE version 2.1.0 due to its user-friendly interface and ease of programming in C. The objective is to examine the vehicle's dynamic behavior, specifically focusing on lateral velocity and yaw rate. This addition highlights the integration of CarSim and MATLAB with the ESP32 platform, emphasizing the practical benefits of using Arduino IDE for programming in C, especially for real-time simulations.

In the CarSim environment, we set up the vehicle dynamic model to operate in a continuous time framework and compare it with discrete time modeling using the explicit Euler method. The study in [23] demonstrates the effectiveness of the explicit Euler model when applied to a small sample size. For this implementation, we set the sample time to $T_s = 0.1$ ms.

Using CarSim, we simulate the detailed vehicle dynamics, providing a realistic testbed for the control algorithms. MATLAB is used to interface with CarSim, allowing us to implement and compare the discrete-time control algorithms developed for the ESP32. This integration enables us to simulate and analyze the vehicle's behavior under various driving conditions and control scenarios in a highly controlled and repeatable manner.

On the ESP32 board, the discrete-time vehicle dynamic model utilizes a $T_s = 1$ ms sampling period, ensuring model precision and avoiding code overflow issues. This configuration allows us to evaluate the ESP32 performance in automotive dynamics control applications, leveraging current computational hardware capabilities to adjust the sampling period to lower values without encountering overflow problems. This is typically challenging in automotive embedded systems, where resource optimization and cost efficiency are prioritized.

Validating behavior in more realistic scenarios involves utilizing the embedded system's features, such as its communication interfaces and I/O capabilities, enabling tests under variable conditions and ensuring robust system performance.

This work's proposed implementation allows for code optimization to enhance efficiency, providing a thorough evaluation of the vehicle dynamic model's behavior under real operating conditions. Additionally, real-time signal analysis is conducted, adding an extra validation layer to ensure the system responds appropriately to environmental dynamics.

Validation is performed using an oscilloscope to measure the analog signals from the ESP32. Although automotive systems commonly use CAN bus for communication, this study omits such communication to validate in real-time without additional computational systems, avoiding extra clock cycles or communication issues, and focusing on optimizing the proposed control code. The ESP32 features two integrated DAC channels, which convert digital signals into analog signals with an 8-bit resolution, generating voltages in the range of 0 to 3.3 volts. We use the analog outputs of the ESP32, monitored by an oscilloscope, to track control signals and the steering wheel inputs, without the additional cost of serial communication.

To enable accurate signal representation from the ESP32's DAC, which outputs only positive integer signals, we apply a bias. This bias adjusts the DAC output range, ensuring that signals are displayed correctly and can be accurately read by the oscilloscope.

Scaling the signal range enhances visualization, providing reliable data for system validation and capturing environmental dynamics effectively.

For instance, we consider the process of generating the steering signal δ_d, shown in Figure 2. This signal operates in a double-step manner, which can lead to abrupt changes that may cause the vehicle to slip (illustrated in Figure 3). The δ_d values range from $\pm 100°$.

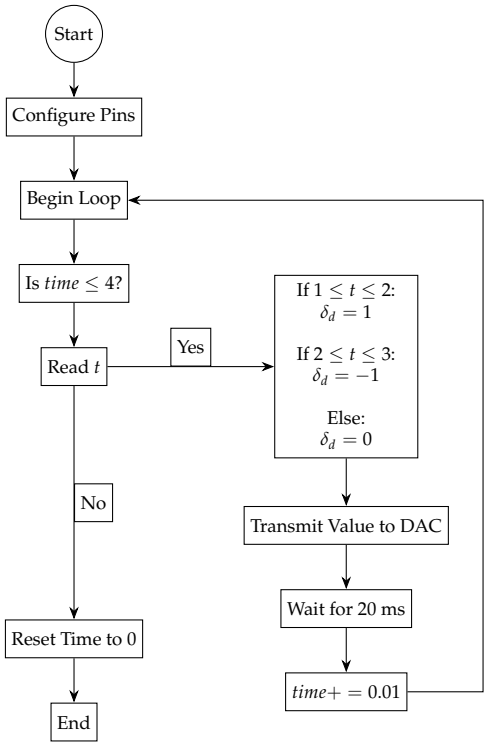
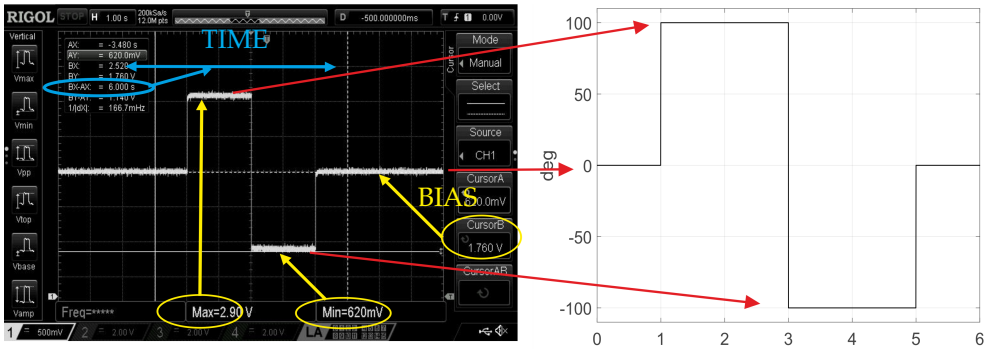


Figure 2. Simulation process of the steering signal on ESP32.



(**a**) (**b**)

Figure 3. Steering angle δ_d: (**a**) signal measured on the oscilloscope with the following values 500 mV/div and 1 s/div, (**b**) signal generated in MATLAB.

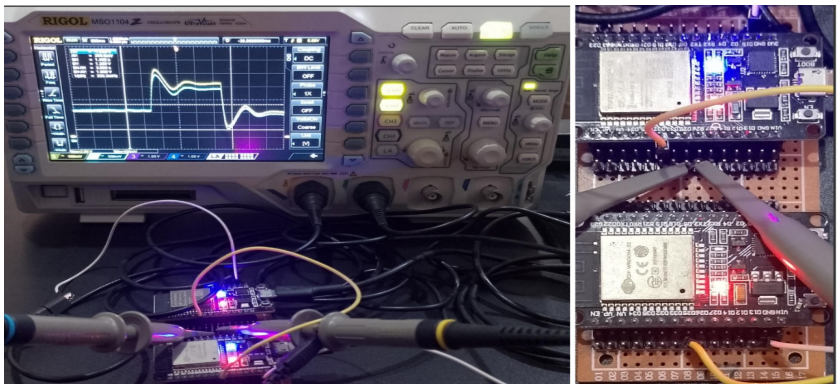


Figure 4. Configuration using two ESP32 boards interconnected via analog and digital pins. The setup emulates the steering signal, while the vehicle dynamics control obtains the response through analog pins, with the produced signals displayed on an oscilloscope.

Figure 3a displays the signal from GPIO 25. To represent 0°, a bias of 1.760 V is used, resulting in a peak voltage of 2.9 V for 100° and a minimum voltage of 620 mV for −100°. These values are compared with MATLAB outputs in Figure 3b.

In our setup, GPIO 25 generates the steering signal, which is transmitted via GPIO 32 to another ESP32 board. On this secondary board, signals from GPIO 25 and GPIO 26 are captured (see Figure 4). This configuration allows us to emulate the steering signal and obtain the response through analog pins, facilitating the verification and comparison of signals generated by the first ESP32 with those recorded by the second, thereby enabling thorough evaluation of the system's performance.

Figures 5 and 6 compare the lateral velocity and yaw rate in an open-loop system as generated by both MATLAB-CarSim and the ESP32 board. These figures focus on the system's performance in an open-loop configuration, allowing us to observe the response of the vehicle's dynamics without feedback control. Specifically, Figures 5a and 6a present the signals from the ESP32 board, highlighting how the open-loop control behaves in a real-world hardware environment. This reveals potential discrepancies or noise introduced by the physical implementation on the ESP32. Meanwhile, Figures 5b and 6b shows the signals from MATLAB-CarSim, which serve as a reference for comparing the simulation results with real-time data.

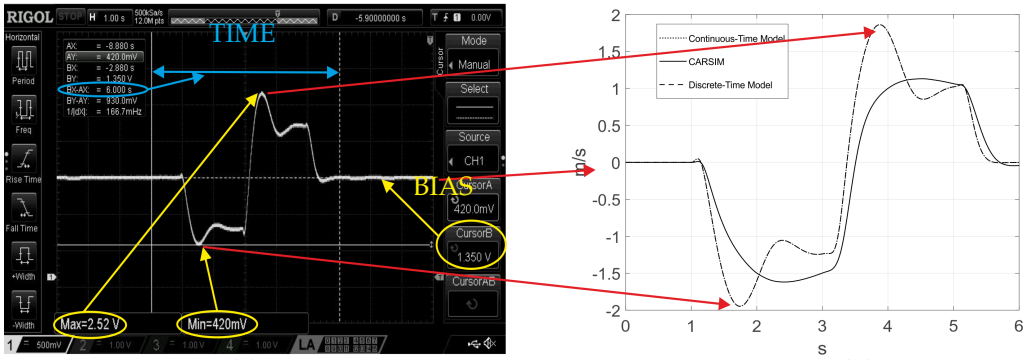
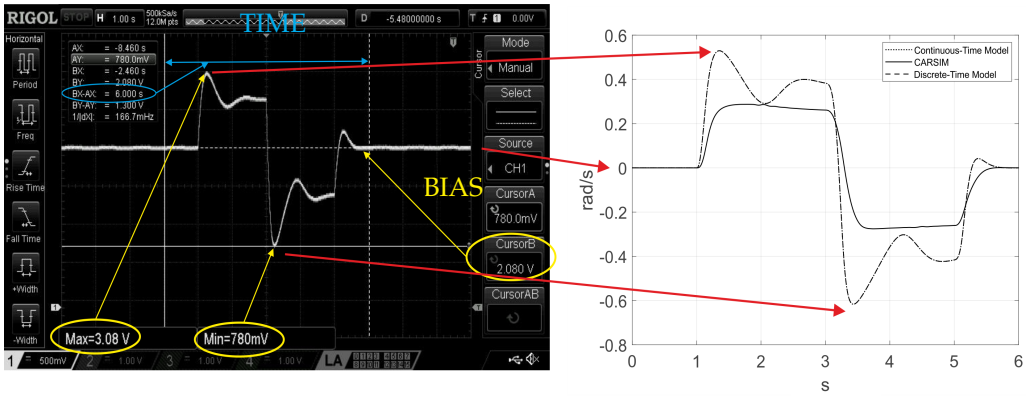


Figure 5. Open-loop lateral velocity $v_{y,k}$, (**a**) signal measured on the oscilloscope with the settings: 500 mV/div and 1 s/div, (**b**) signal generated in MATLAB.



(**a**) (**b**)

Figure 6. Open-loop yaw angular velocities $\omega_{z,k}$, (**a**) signal measured on the oscilloscope with the settings: 500 mV/div and 1 s/div, (**b**) signal generated in MATLAB.

This analysis of both simulation and real-time conditions provides a deeper understanding of the system's behavior under open-loop conditions, laying the foundation for further evaluation in closed-loop scenarios.

We have analyzed the signals in these figures to provide a clear understanding of the system's behavior under both simulation and real-time processing conditions. Specifically, Figures 5a and 6a present the signals produced by the ESP32 board, showcasing the practical implementation of the open-loop control on the hardware. This allows us to observe how the system responds in a real-world scenario, highlighting any potential discrepancies or noise the hardware introduces.

Conversely, Figures 5b and 6b illustrate the signals generated by MATLAB and CarSim. These figures show the system's theoretical response based on mathematical models and simulations executed in MATLAB. However, in the case of CarSim, the vehicle's performance shows improvement, as the software does not rely on a dynamical model but rather on experimental data from dynamical tables. This difference underscores the importance of comparing these two sets of figures to validate the accuracy and reliability of simulations against actual hardware performance. In this comparison, we aim to identify any significant differences between the simulated and real-world responses. Such an analysis is vital for fine-tuning the control algorithms and ensuring that the system performs optimally in practical applications. The detailed examination of these figures provides valuable insights into the system's dynamics and the efficacy of the control strategies employed.

4. Discussion

The implementation of the controller is illustrated in the flowchart of the ESP32 program, which includes both the model and the controller as depicted in Figure 7. This flowchart outlines the sequence of operations that define the program's functionality. Furthermore, the reference model is crafted with consideration for the driving conditions of an ideal vehicle, facilitating the generation of lateral velocity and yaw rate tracking error signals necessary for control parameter computations and variable updates within the main loop.

The ESP32 board, with its two DAC channels, requires reprogramming to modify the port's output signals for every set of results, thereby capturing various aspects of the system's behavior. This reprogramming process enables a thorough evaluation of the controller's response under different conditions and simplifies the identification of potential enhancements or necessary adjustments in the control algorithm.

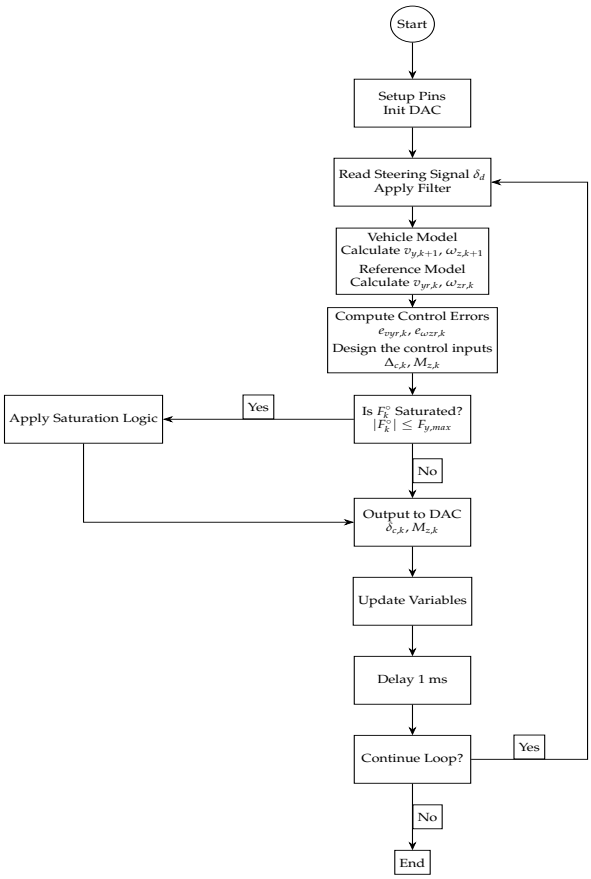


Figure 7. Simulation of the control process and vehicle dynamic modeling in discrete time, implemented on a microcontroller such as the ESP32.

The values used for implementation in both the ESP32 and MATLAB are based on typical parameters for a midsize vehicle, validated through prior research and empirical data. This validation involves the Pacejka model parameters derived from experimental tire data obtained from industry-standard data sets and published studies. Data from tire test benches and real-world vehicle tests across various road conditions (dry, wet, and slippery) ensure realistic tire behavior. Furthermore, simulation results are validated using CarSim, comparing outputs to realistic vehicle dynamics [25]: $m = 1800$ kg, $l_f = 1.38$ m, $l_r = 1.53$ m, $J_z = 2386$, $C_{yf,k} = 1.41$, $C_{yr,k} = 1.51$, $B_{yf,k} = 7.2$, $B_{yr,k} = 11$, $D_{yf,k} = 8854$, $D_{yr,k} = 8394$, $\mu = 0.95$.

The parameters for the reference vehicle are as follows: $B_{yfr} = 16$, $B_{yrr} = 16$, $D_{yfr} = 10{,}000$, $D_{yrr} = 10{,}000$.

The gains of the digital PI-like controller are $k_{10} = -0.1$, $k_{11} = 0.97$, $k_{20} = -0.1$, $k_{21} = 0.97$.

Each figure provides a visual representation of the obtained results, facilitating interpretation and analysis of the effectiveness of the implemented controller on the ESP32. The red line indicates the vehicle velocity, while the blue line indicates the reference velocity.

The system's lateral velocity tracking capabilities are illustrated in Figure 8. In Figure 8a, the lateral velocity measurement recorded by the ESP32 is depicted. This figure highlights the real-time data acquisition capabilities of the ESP32 embedded system, demonstrating its efficacy in monitoring and processing vehicle dynamics during operation.

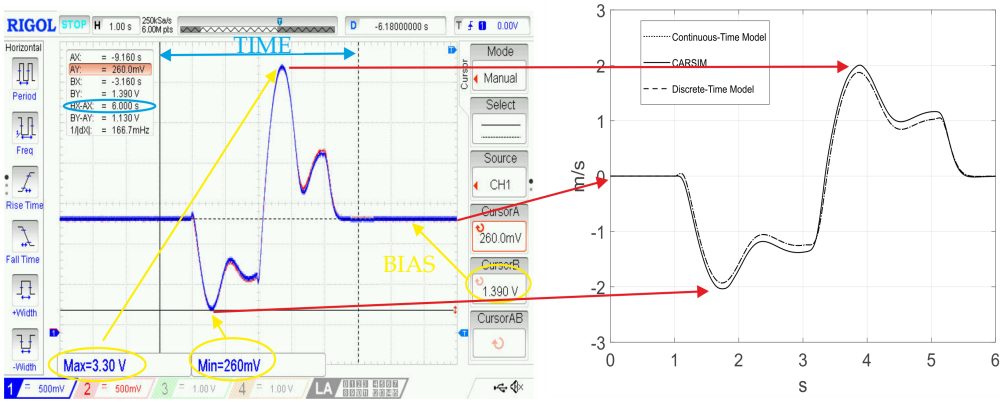


Figure 8. Lateral velocity $v_{y,k}$. (**a**) Signal measured on the oscilloscope with the following settings: 500 mV/div and 1 s/div. (**b**) Signal obtained from MATLAB.

In contrast, Figure 8b presents the lateral velocity measurement obtained through MATLAB software. These MATLAB-generated data serve as a benchmark, providing a reference for evaluating the accuracy and reliability of the ESP32 measurements. The data sets in Figure 8a,b allow us to thoroughly test how well the ESP32-based control system works compared to the MATLAB model. This comparison shows that the ESP32 works well for monitoring and controlling things in real-time, showing that the embedded system can provide accurate and quick results in situations where the vehicle dynamic is changing. Moreover, this evaluation validates the feasibility of employing a low-cost, commercially available SoC for advanced automotive control applications, potentially offering a cost-effective alternative to more expensive traditional systems.

The signal corresponds to the reference velocity, represented by the blue line, and the red line corresponds to the velocity of the vehicle.

Figure 9 illustrates the yaw rate tracking performance of the system. In Figure 9a, the yaw rate measurement conducted via the ESP32 is depicted, showcasing the embedded system's capability to acquire and process real-time data on vehicle dynamics. This figure demonstrates the precision and responsiveness of the ESP32 in capturing yaw rate changes during vehicle operation.

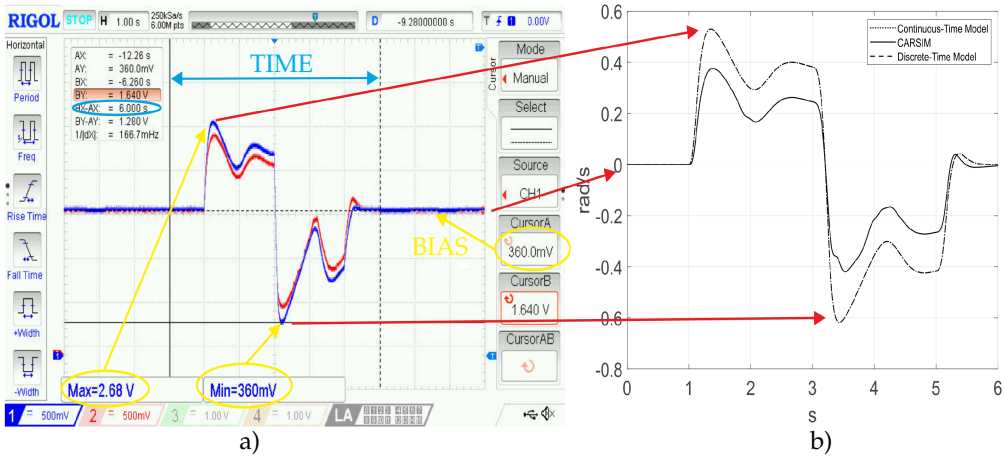


Figure 9. Yaw angular velocities $\omega_{z,k}$. (**a**) Signal measured on the oscilloscope with scale values of 500 mV/div and 1 s/div. (**b**) Signal obtained from MATLAB.

Conversely, Figure 9b displays the yaw rate measurement obtained through MATLAB software, which serves as a benchmark for comparison. The MATLAB-generated data provide a reference to evaluate the accuracy and reliability of the ESP32 measurements.

By comparing the data sets presented in Figure 9a,b, we can rigorously assess the performance of the ESP32-based control system against the MATLAB model. This comparison highlights the effectiveness of the ESP32 in real-time monitoring and control tasks, confirming that the embedded system can deliver accurate and responsive performance in dynamic vehicle scenarios.

This evaluation further validates the feasibility of utilizing a low-cost, commercially available SoC for advanced automotive control applications, potentially offering a cost-effective alternative to more expensive traditional systems.

The control action $M_{z,k}$ is illustrated in Figure 10, depicting its temporal evolution during the simulation. Figure 10a represents the signal generated by the ESP32, while Figure 10b shows the corresponding signal obtained through MATLAB software.

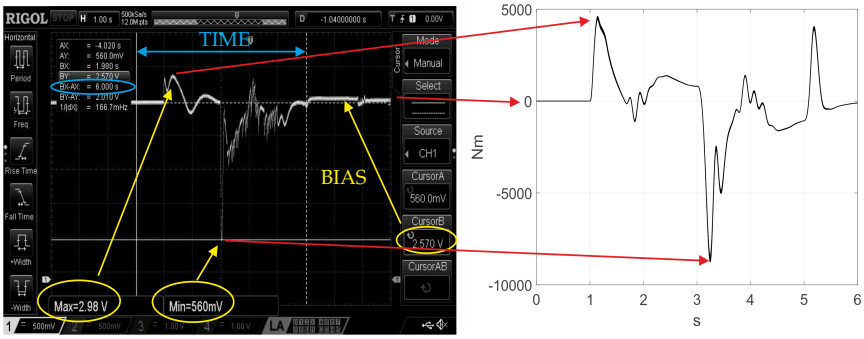


Figure 10. Control action $M_{z,k}$. (**a**) Signal measured on the ESP32 using an oscilloscope with scale values of 500 mV/div and 1 s/div. (**b**) Signal obtained from MATLAB.

The behavior of the control action $\delta_{c,k}$ is depicted in Figure 11, illustrating its variation over time during the simulation. Figure 11a represents the signal measured on the ESP32, while Figure 11b displays the signal generated by MATLAB software.



Figure 11. Control action $\delta_{c,k}$. (**a**) Signal measured on the ESP32 using an oscilloscope with scale values of 500 mV/div and 1 s/div. (**b**) Signal obtained from MATLAB.

5. Conclusions

This paper presents the utilization of a System-on-Chip (SoC) for implementing real-time digital proportional–integral (PI-like) control to manage vehicle dynamics. The real-time processing of the steering signal allows for dynamic adjustments to the vehicle's steering and torque. We demonstrate the development of driving assistance algorithms for Active Front Steering (AFS) and Rear Torque Vectoring (RTV) on this low-cost, commercially available embedded system. We generate the steering signal using another SoC to enhance realism, mimicking a steer-by-wire system through its analog output ports. This setup enables the simulation and evaluation of control algorithms in a realistic test environment, ensuring improved vehicle dynamic stability and maneuverability under

various conditions. Using the practical and high-performance digital PI-like control that the SoC platform offers, our goal is to build a platform for testing discrete controllers that can run vehicles in real-time. This approach expedites and reduces the cost of enhancing vehicle dynamic system stability and responsiveness, allowing for immediate verification and adjustment of control parameters. Overall, this study shows that the SoC has much potential as a low-cost, flexible solution for advanced vehicle dynamics control. It provides a useful way to improve the stability and responsiveness of vehicles dynamic by checking and changing control parameters immediately.

Future work may involve expanding the capabilities of the ESP32 platform for more advanced control strategies and system integrations, including communication via CAN bus and robust fail-safe mechanisms. For instance, exploring the implementation of sliding mode control algorithms or adaptive control techniques could further enhance the vehicle's dynamic performance. Optimizing communication and data exchange between multiple ESP32 boards could also enable more sophisticated control architectures or distributed control systems.

Considering the potential for autonomous driving applications, future research could focus on developing autonomous control algorithms and integrating them with the existing ESP32-based control platform. This could involve incorporating perception and decision-making components to enable autonomous navigation and maneuvering capabilities. Furthermore, investigating the integration of sensor fusion techniques and machine learning algorithms could provide a comprehensive approach to vehicle control, ensuring greater stability and efficiency across various driving conditions.

Author Contributions: Conceptualization, A.N.G. and M.A.M.-A.; Methodology, A.N.G., C.A.L., J.A.G.-R. and C.V.-S.; Validation, A.N.G. and C.V.-S.; Investigation, M.A.M.-A. and A.N.G.; Writing—original draft, A.N.G., C.A.L., J.A.G.-R., C.V.-S. and M.A.M.-A.; Writing—review and editing, A.N.G., C.A.L., J.A.G.-R., C.V.-S. and M.A.M.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kissai, M.; Monsuez, B.; Tapus, A. Review of integrated vehicle dynamics control architectures. In Proceedings of the 2017 European Conference on Mobile Robots (ECMR), Paris, France, 6–8 September 2017; pp. 1–8. [CrossRef]
2. Skrickij, V.; Kojis, P.; Šabanovič, E.; Shyrokau, B.; Ivanov, V. Review of Integrated Chassis Control Techniques for Automated Ground Vehicles. *Sensors* **2024**, *24*, 600. [CrossRef]
3. Bianchi, D.; Borri, A.; Di Benedetto, M.; Di Gennaro, S. Active Attitude Control of Ground Vehicles with Partially Unknown Model. *IFAC-PapersOnLine* **2020**, *53*, 14420–14425. [CrossRef]
4. Lúa, C.A.; Bianchi, D.; Di Gennaro, S. Nonlinear observer-based adaptive control of ground vehicles with uncertainty estimation. *J. Frankl. Inst.* **2023**, *360*, 14175–14189. [CrossRef]
5. Seo, Y.; Cho, K.; Nam, K. Integrated Yaw Stability Control of Electric Vehicle Equipped with Front/Rear Steer-by-Wire Systems and Four In-Wheel Motors. *Electronics* **2022**, *11*, 1277. [CrossRef]
6. Zhang, J.; Wang, H.; Zheng, J.; Cao, Z.; Man, Z.; Yu, M.; Chen, L. Adaptive sliding mode-based lateral stability control of steer-by-wire vehicles with experimental validations. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9589–9600. [CrossRef]
7. Zhang, C.; Chang, B.; Wang, J.; Li, S.; Zhang, R.; Ma, J. Robust Control Design of Active Front-Wheel Steering on Low-Adhesion Road Surfaces. *World Electr. Veh. J.* **2021**, *12*, 153. [CrossRef]
8. de Carvalho Pinheiro, H.; Carello, M.; Punta, E. Torque vectoring control strategies comparison for hybrid vehicles with two rear electric motors. *Appl. Sci.* **2023**, *13*, 8109. [CrossRef]
9. Acosta Lúa, C.; Di Gennaro, S.; Navarrete Guzmán, A.; Rivera Domínguez, J. Digital sliding mode controllers for active control of ground vehicles. *Asian J. Control* **2021**, *23*, 2129–2144. [CrossRef]
10. Wu, Z.; Kang, C.; Li, B.; Ruan, J.; Zheng, X. Dynamic Modeling, Simulation, and Optimization of Vehicle Electronic Stability Program Algorithm Based on Back Propagation Neural Network and PID Algorithm. *Actuators* **2024**, *13*, 100. [CrossRef]
11. Cespi, R.; Di Gennaro, S.; Castillo-Toledo, B.; Romero-Aragon, J.C.; Ramírez-Mendoza, R.A. Neural Network Inverse Optimal Control of Ground Vehicles. *Neural Process. Lett.* **2023**, *55*, 10287–10313. [CrossRef]

12. González-López, J.M.; Pérez, S.S.; Betancourt, R.O.J.; Barreto, G. Active Control for an Electric Vehicle with an Observer for Torque Energy-Saving. *World Electr. Veh. J.* **2023**, *14*, 288. [CrossRef]
13. Cisneros, S.O.; Varela, J.M.R.; Acosta, M.A.R.; Dominguez, J.R.; Villalobos, P.M. Pollen grains classification with a deep learning system GPU-trained. *IEEE Lat. Am. Trans.* **2021**, *20*, 22–31. [CrossRef]
14. Rivera-Acosta, M.; Ortega-Cisneros, S.; Rivera, J. Automatic tool for fast generation of custom convolutional neural networks accelerators for FPGA. *Electronics* **2019**, *8*, 641. [CrossRef] [PubMed]
15. Baungarten-Leon, E.I.; Ortega-Cisneros, S.; Jaramillo-Toral, U.; Rodriguez-Navarrete, F.J.; Pizano-Escalante, L.; Raygoza-Panduro, J. Vector Accelerator Unit for Caravel. *IEEE Embed. Syst. Lett.* **2023**, *16*, 73–76. [CrossRef] [PubMed]
16. Venceslau de Souto, J.I.; Barbosa da Rocha, Á.; Duarte, R.N.C.; de Moura Fernandes, E. Design and Implementation of an Embedded Data Acquisition System for Vehicle Vertical Dynamics Analysis. *Sensors* **2023**, *23*, 9491. [CrossRef]
17. González, A.; Olazagoitia, J.L.; Vinolas, J. A low-cost data acquisition system for automobile dynamics applications. *Sensors* **2018**, *18*, 366.
18. Kareem, H.; Dunaev, D. The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design. In Proceedings of the 2021 4th International Conference on Circuits, Systems and Simulation (ICCSS), Kuala Lumpur, Malaysia, 26–28 May 2021; pp. 130–135.
19. Babiuch, M.; Foltỳnek, P.; Smutnỳ, P. Using the ESP32 microcontroller for data processing. In Proceedings of the 2019 20th International Carpathian Control Conference (ICCC), Krakow-Wieliczka, Poland, 26–29 May 2019; pp. 1–6.
20. Rajamani, R. *Vehicle Dynamics and Control*; Mechanical Engineering Series; Springer: Cham, Switzerland, 2011.
21. Pacejka, H. *Tire and Vehicle Dynamics*; Elsevier: Amsterdam, The Netherlands, 2005. [CrossRef]
22. Wong, J.Y. *Theory of Ground Vehicles*; John Wiley & Sons: Hoboken, NJ, USA, 2001.
23. Guzmán, A.N.; Di Gennaro, S.; Domínguez, J.R.; Lua, C.A.; Loukianov, A.G.; Castillo-Toledo, B. Enhanced discrete-time modeling via variational integrators and digital controller design for ground vehicles. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6375–6385.
24. Lúa, C.A.; Di Gennaro, S.; Guzman, A.N.; Ortega-Cisneros, S.; Domínguez, J.R. Digital implementation via FPGA of controllers for active control of ground vehicles. *IEEE Trans. Ind. Electron.* **2019**, *15*, 2253–2264.
25. Acosta Lúa, C.; Di Gennaro, S.; Flores Jiménez, A.B.; Navarrete Guzmán, A. Robust dynamic control for electric vehicles with estimation of parametric uncertainties and external disturbances. *IEEE Access* 2024, *in press*.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.