POLITECNICO DI TORINO

MASTER's Degree in AUTOMOTIVE ENGINEERING

MASTER's Degree Thesis

DESIGN AND IMPLEMENTATION OF A CONTROL ARCHITECTURE FOR THE SEMI-AUTONOMOUS CYCLE RICKSHAW

Supervisors

Prof. Alessandro VIGLIANI

Dr. Angelo Domenico VELLA

Prof. Dr.-Ing. Klaus BOGENBERGER [TUM]

M.Sc. Evald NEXHIPI [TUM]

M.Sc. Mario ILIC [TUM]

Candidate

Haroun KHALED

JULY 2026

Acknowledgment

I would like to express my sincere gratitude to all those who supported me throughout the work on this thesis.

First and foremost, I would like to thank my supervisors from the Department of Mechanical and Aerospace Engineering at Politecnico di Torino. I am deeply grateful to Prof. Alessandro Vigliani and Dr. Angelo Domenico Vella for their continuous guidance, technical expertise, and constructive feedback throughout the entire duration of this project. Their availability and willingness to discuss both conceptual and practical challenges at every stage of the work have been invaluable.

My gratitude extends to the external supervisors at the Technical University of Munich, Evald Nexhipi, M.Sc., and Mario Ilic, M.Sc., for their academic support and for the opportunity to carry out this work between TU Munich and Politecnico di Torino.

I would also like to express my appreciation to Prof. Dr.-Ing. Klaus Bogenberger for providing the institutional framework and research environment that made this thesis possible.

Abstract

Urban mobility increasingly requires transport solutions that are sustainable, space-efficient, and flexible enough to operate across different infrastructure types. This thesis investigates a semi-autonomous cycle rickshaw as a novel vehicle concept for passenger and freight transport on both cycling and road infrastructure. The work focuses on the challenge of reliably executing pre-planned trajectories despite physical limitations such as restricted steering range and low-speed operation. To address this problem, a hierarchical control architecture is designed, implemented, and evaluated on the rickshaw prototype. The architecture combines high-level trajectory execution logic with low-level control of steering, braking, and, additionally, electric assistance, structured as a model predictive controller for steering, a PID controller for longitudinal motion, and an explicit safety supervision layer. The control design accounts for actuator limits, vehicle motion constraints, and feedback-based trajectory tracking under varying operating conditions, and is supported by the integration of the wheel-speed and steering-angle sensing required for closed-loop operation.

The proposed approach is implemented on the physical prototype and assessed in controlled experiments using metrics for tracking accuracy, stability, safety, and responsiveness. The experimental results show that the integrated control chain executes the planned trajectories with small lateral deviation on straight and constant-curvature segments, the tracking error concentrating on the transitions between segments of different curvature. In addition, the safety supervision layer keeps the steering, rate, and speed commands within the prescribed bounds throughout every run, with no unstable or unsafe responses. A comparison of four controller profiles, namely Baseline, Comfort, Tracking, and Robust, indicates that each reproduces its intended trade-off: the Tracking profile minimises lateral deviation, the Comfort profile produces the smoothest actuation, and the Robust profile remains the most stable under modelled delay and noise.

Overall, the thesis provides a systematic assessment of the controllability and practical feasibility of the semi-autonomous cycle rickshaw. The findings indicate that the vehicle can execute pre-planned trajectories with sufficient accuracy, stability, and safety to support its intended role as a low-speed urban transport platform within the validated operating domain, while also highlighting the extension to higher speeds, representative payloads, and a full quantitative comparison of the controller variants as the principal directions for future work.

Contents

1. Introduction

Urban areas face growing pressure to develop transport systems that are simultaneously sustainable, space-efficient, and capable of serving both passenger mobility and freight distribution. The expansion of e-commerce, the demand for first- and last-mile logistics, and the scarcity of road space in dense environments have intensified the need for vehicle concepts that can operate where conventional solutions are impractical in terms of footprint, energy consumption, and operational flexibility. Lightweight and partially automated vehicle concepts represent a promising response to these challenges. Among them, the semi-autonomous cycle rickshaw developed at the Chair of Traffic Engineering and Control of the Technical University of Munich is a particularly interesting platform, since it combines the compactness and energy efficiency characteristic of cycle-based vehicles with automated driving functions suitable for both passenger and freight transport in urban environments (**Margreiter**).

The practical feasibility of such a concept, however, depends critically on its controllability. As a non-standard platform operating at low to moderate speeds, the rickshaw is subject to significant physical limitations, including a restricted steering range and actuator delays. Its low operating speed makes it especially sensitive to disturbances, because effects such as static friction in the steering mechanism, surface irregularities, and payload variations are not overcome by inertia as they would be at higher speed, and a delay that would be negligible on a fast vehicle translates into a proportionally larger tracking error over the short distances travelled. These factors together make reliable trajectory execution a non-trivial problem. This thesis addresses that problem through the design, implementation, and evaluation of a control architecture for the semi-autonomous cycle rickshaw. The architecture pursues three concrete objectives. The first is to execute pre-planned trajectories accurately and stably within the steering and speed limits of the platform, by combining a high-level trajectory-tracking layer with low-level actuation control of the steering and longitudinal channels. The second is to enforce the physical and safety bounds of the vehicle at all times through a dedicated supervision layer, so that no commanded action can exceed the admissible operating envelope. The third is to validate the complete system experimentally on the physical prototype under realistic low-speed operating conditions, rather than in simulation alone. Together these objectives define a control solution that is not only functional in principle but demonstrably reliable on the real vehicle.

Figure 1.1: The semi-autonomous cycle rickshaw prototype developed at the Chair of Traffic Engineering and Control, TU Munich (**Margreiter**).

1.1. The Semi-Autonomous Cycle Rickshaw

The vehicle under study is a three-wheeled fully electric rickshaw developed as a research platform for connected and automated urban transport, designed to carry passengers, freight, or a combination of both (**Margreiter**). It is built on a delta configuration, with a single steered front wheel and two driven rear wheels, and is powered entirely by an electric drivetrain. Its sensor suite, on-board computing units, and Autoware-based software stack place it at an intermediate point between a conventional cargo bicycle and a fully automated road vehicle, combining the compactness and low energy consumption of cycle-based transport with the automated driving functions required for reliable passenger and freight service in urban environments.

It occupies an intermediate position between micromobility devices and conventional automated road vehicles: compact and energy-efficient in its physical form, yet equipped with steering, braking, and propulsion actuators, onboard computing units, a multi-sensor perception suite, and an Autoware-based[1] software architecture for automated driving (**Margreiter**).

Its reduced footprint and reliance on widely available, affordable components make it particularly suited to constrained urban environments such as university campuses, inner-city districts, or narrow streets, where larger automated vehicles are impractical.

These characteristics also position it as a lower-cost alternative to more complex automated platforms, with potential applicability in both developed and developing urban contexts. Despite these advantages, the vehicle introduces important control challenges. Its restricted steering range limits the set of feasible trajectories, while low-speed operation amplifies the influence of actuator delays, friction, and external disturbances.

[1] Autoware Universe is an open-source autonomous driving software stack maintained by the Autoware Foundation. `https://autowarefoundation.github.io/autoware-documentation/`

Safe transport of passengers or freight therefore requires not only path-following capability, but robust and stable execution under realistic operating conditions. The rickshaw thus serves simultaneously as the application context and the experimental basis for the research presented in this thesis.

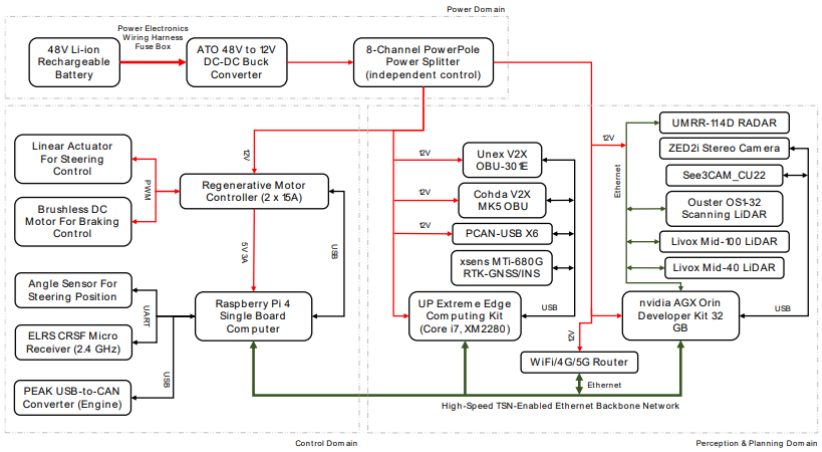

Figure 1.2: Main hardware components of the rickshaw: LattePanda compute unit, ESP32-EVB micro-controller, sensor suite, and actuation system (**Margreiter**).

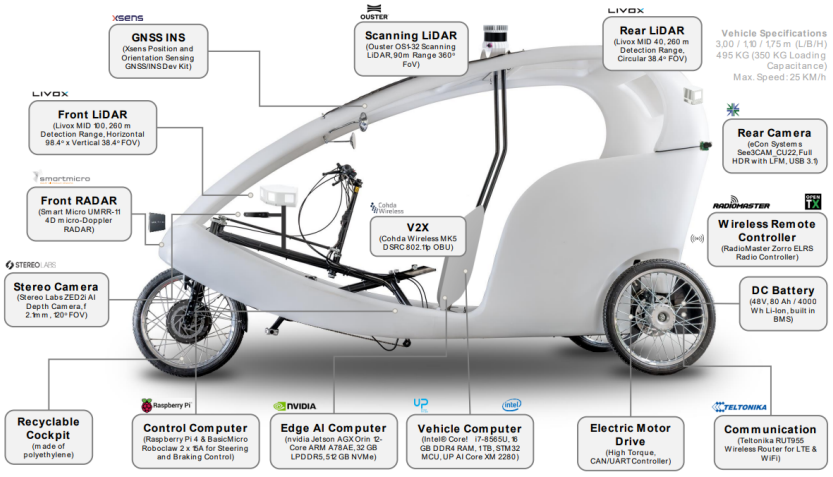

Figure 1.3: Hardware architecture block diagram showing the two-layer computing structure and the communication links between subsystems (**Margreiter**).

1.2. Thesis objectives and measurable goals

This thesis pursues a set of technical, implementation-oriented, and evaluative objectives. The first objective is to design a hierarchical control architecture that enables the semi-autonomous cycle rickshaw to execute pre-planned trajectories in a stable, safe, and structured manner. This architecture combines high-level trajectory execution with low-level

control of steering, braking, and longitudinal motion, and explicitly accounts for physical constraints such as steering saturation, actuator delay, and low-speed operating limits.

The second objective is to implement this architecture within the existing software and hardware environment of the vehicle, including the Autoware-based control chain, ROS 2 communication, and the interface toward embedded actuation via micro-ROS.

The third objective is to extend the prototype with the sensing capabilities required for closed-loop validation, in particular real wheel-speed feedback and steering-angle measurement, whose sensor selection and mechanical integration form part of the work presented in this thesis. The fourth objective is to validate the complete system through controlled experiments on the physical rickshaw prototype, demonstrating stable and accurate trajectory execution under realistic operating conditions.

These objectives are consistent with the current development status of the project: the longitudinal and lateral control architecture has already been designed, tuned, and evaluated in a first series of tests; the software integration between Autoware, the trajectory controller, and the micro-ROS actuation interface has been prepared; and appropriate candidates for the speed and steering sensors have been identified and mechanically designed.

To make the contribution of the thesis assessable in a rigorous and reproducible manner, the above objectives are translated into the following measurable goals.

[leftmargin=8em, labelindent=0pt, labelwidth=7.5em, font=**]

Tracking accuracy: **quantified through lateral deviation, heading error, speed-tracking error, and root-mean-square trajectory error along straight and curved segments.**

Control quality: **assessed through indicators such as overshoot, settling time, steering command smoothness, and rate of change of actuation signals.**

Robustness: **evaluated under non-ideal conditions including measurement noise, command delay, progress mismatch, and payload variation representative of low-speed urban operation.**

Constraint compliance: **verified through adherence to steering and speed limits, bounded actuator behaviour, and the absence of unsafe or unstable control responses.**

A further objective of methodological significance is the establishment of a consistent validation workflow linking trajectory planning, control execution, sensor integration, and prototype testing. This includes the preparation of representative test scenarios, the use of comparable metrics across controller variants, and the generation of results that support a structured comparison between baseline and improved strategies. In this sense, the thesis aims not only to deliver a functioning controller, but also to provide a systematic basis for judging the practical feasibility of the semi-autonomous cycle rickshaw within its intended low-speed urban operating domain.

1.3. Thesis Outline

The remainder of this thesis is organised as follows. Chapter 2 presents a review of the relevant literature, covering vehicle modelling approaches for three-wheeled platforms, tra-

jectory tracking algorithms, low-level control strategies for steer-by-wire systems, and feedback control methods. The chapter concludes with the identification of the research gap that motivates this work. Chapter 3 describes the vehicle platform and its hardware and software architecture in detail, and develops the mathematical vehicle model — kinematic, dynamic, and rollover geometry — that underpins the control design. Chapter 4 presents the design and implementation of the proposed control architecture, covering longitudinal PID control, lateral MPC, safety supervision, and the full sensor integration including the embedded firmware, signal conditioning, and micro-ROS actuation interface. Chapter 5 presents the experimental validation results on the physical rickshaw, including the four controller profiles and their comparative analysis. Chapter 6 concludes the thesis, summarises the contributions and outlines directions for future work. The overall structure is illustrated in Figure 1.4.

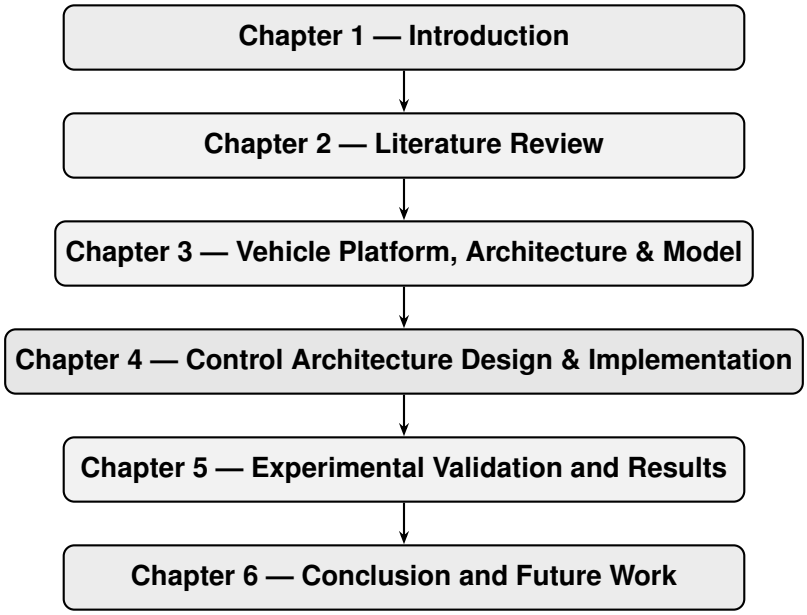

Figure 1.4: Overview of the thesis structure and chapter organisation.

2. Literature Review

The design of a control architecture for a semi-autonomous cycle rickshaw lies at the intersection of several established research areas: the modelling of three-wheeled and constrained ground vehicles, the design of trajectory-tracking algorithms, the low-level control of steer-by-wire systems, and the broader comparison between open-loop and closed-loop feedback strategies. None of these areas has been developed with explicit reference to a non-tilting, low-speed cycle rickshaw operating on both cycling and road infrastructure. The literature instead provides a collection of partial solutions, each focused on a specific subsystem or vehicle category, which must be selected, adapted, and combined to form a coherent control architecture for the platform considered in this thesis.

The purpose of this chapter is to review these contributions in a structured manner, with an emphasis on the elements that are relevant for the rickshaw and on the limitations that motivate the present work. Section 2.1 situates the rickshaw within the broader landscape of automated vehicles for urban passenger and freight transport. Section 2.2 surveys vehicle models for three-wheeled platforms, with particular attention to single-track approximations and to indices used to describe rollover and lateral stability. Section 2.3 discusses trajectory tracking algorithms, ranging from purely geometric pursuit laws to model predictive formulations, while Section 2.4 addresses low-level control strategies for steer-by-wire systems. Section 2.5 compares open-loop and closed-loop feedback strategies, and Section 2.6 analyses how the physical constraints typical of low-speed three-wheeled vehicles influence controllability, stability, and safety. Section 2.7 then synthesises these observations into a research gap statement that explicitly justifies the scope of the thesis.

cc

headercolor **Paper ID**	**Vehicle type**	**Main topic / focus**	
RodriguezLicea20181	3-wheel 1F2R delta tricycle	Rollover index & braking	Defines RI for delta tricycl

headercolor **Paper ID**	**Vehicle type**	**Main topic / focus**	
Pandey20176	3-wheel front-steer mobile robot	Trajectory tracking (kinematic + PID)	Shows real 3-wheel fron
Stability3W20133	3-wheel vehicles (front / rear single)	Controllability & stability	Analyses over/
Jianmin20104	Passenger car with SbW	Steer-by-wire control strategies	Proposes SbW ste
Perdana20245	SbW / X-by-wire vehicles (review)	Overview of SbW controllers	Reviews PID, SMC, MI
Divelbiss19978	Car–trailer system	Trajectory tracking (LTV-LQR + PID)	Designs time-varying LQR + P
Beal201311	Passenger car at handling limits	MPC handling envelope	Introduces MPC-based h

headercolor **Paper ID**	**Vehicle type**	**Main topic / focus**	
Martins200812	Unicycle-like mobile robot	Adaptive trajectory tracking	Develops kinematic + adap
Kolter201014	Autonomous car (extreme driving)	Mixed open/closed-loop LQR	Shows open-loop replay fails,
Li201715	4-wheel passenger car	3D MPC + active braking	Proposes 3D stability
Akar201416	4-wheel passenger car	Adaptive differential-braking rollover control	Designs switching LTR-I
Dempster202318	Urban autonomous car (bicycle model)	Unified NMPC planning + tracking	Formulates real-time NMPC th
Edelmann20112	Tilting 3-wheel vehicle	Dynamics & stability of tilting three-wheeler	Models and analyses ti

headercolor **Paper ID**	**Vehicle type**	**Main topic / focus**	
IntegralPartition7	Automotive SbW steering test rig	Integral-partition PID for SbW	Proposes integral-pa
Bernardini20099	Drive-by-wire passenger car	Hybrid MPC yaw stabilisation	Shows closed-loop hybrid
Odenthal199910	High-CG 4-wheel vehicle	Nonlinear steering + braking rollover avoidance	Combines steering and b
Luu201913	Passenger car with ACC	Longitudinal spacing control + PID	Designs spacing la
Rahman202517	Autonomous vehicles (survey)	Survey of trajectory planning & tracking	Classifies methods

Table 2.1: Overview of the literature reviewed in this ch
main focus, and key contribution.

2.1. Automated Vehicles for Urban Passenger and Freight Transport

The progressive automation of urban mobility is no longer restricted to passenger cars. Over the past decade, a growing body of work has investigated automated platforms designed specifically for the constraints of dense urban environments, including delivery robots, automated shuttles, and lightweight goods vehicles operating on shared infrastructure. The motivations are well documented: increasing urbanisation, the rapid growth of e-commerce, the need to decarbonise short-distance transport, and the pressure on road space caused by the coexistence of private cars, public transport, freight vehicles, and active modes of mobility. In this context, smaller, slower, electrically driven vehicles are seen as a complement to conventional traffic rather than a direct replacement.

Within this landscape, the semi-autonomous cycle rickshaw developed at the Chair of Traffic Engineering and Control of the Technical University of Munich occupies a deliberately atypical position. As described in the platform's reference publication (**Margreiter**), the vehicle is conceived as an independent platform capable of carrying both passengers and freight on cycling and road infrastructure. It does not behave like a conventional car, since its dimensions, mass, and operating speed are much closer to those of a cargo bicycle; nor does it behave like a small last-mile delivery robot, since it is intended to share infrastructure with motorised vehicles and to transport human passengers, this is the practical reason why the existing literature does not provide an off-the-shelf control architecture for the platform.

Most published work on automated urban vehicles can be divided into three large groups. The first contains studies on conventional automated cars, focusing typically on perception, planning under traffic, and stability at moderate-to-high speeds, as exemplified by the unified planning and tracking framework of Dempster et al. (DEMPSTER et al., 2023). The second group considers small mobile robots and delivery platforms, where the dominant problems are kinematic path following and obstacle avoidance at very low speed; the three-wheeled front-steering robot of Pandey et al. (PANDEY et al., 2017) and the unicycle-like platform of Martins et al. (MARTINS et al., 2008) fall within this category. The third group addresses three-wheeled motorised vehicles, where attention has historically been concentrated on tilting concepts (EDELMANN et al., 2011) or on rollover-prone delta tricycles studied from a stability and braking perspective (RODRÍGUEZ LICEA et al., 2018; "Stability of Three-Wheeled Vehicles with and without Control System" 2013).

The semi-autonomous rickshaw does not belong cleanly to any of these groups. Its operational envelope and physical layout share features with all three, but its target use case — predictable, low-speed, mixed passenger and freight transport on shared urban infrastructure — places it in a regime that has received comparatively little dedicated attention. This observation is the starting point for the literature review developed in the remainder of this chapter.

2.2. Vehicle Models for Three-Wheeled Platforms

A control architecture is only as reliable as the model on which it is based. For ground vehicles, the literature offers a continuum of representations ranging from purely kinematic constructions, where forces and tyre slip are neglected, to high-fidelity multibody models incorporating suspension, tyre dynamics, and aerodynamics. The appropriate level of detail depends on the operating speed, the manoeuvres of interest, and the computational resources available for the controller.

2.2.1. Kinematic and Single-Track Representations

For the low-speed regime that characterises the rickshaw, kinematic models derived from the bicycle (single-track) representation are widely accepted as a reasonable trade-off between accuracy and simplicity. In the standard bicycle model, the two front wheels and the two rear wheels of a four-wheeled vehicle are merged into a single equivalent wheel located on the longitudinal symmetry plane, and the vehicle motion is described by the position of a reference point, the heading angle, and the steering angle of the equivalent front wheel.

This formulation transfers naturally to a three-wheeled platform with a single front steering wheel and two rear wheels, since the rear axle can be replaced by a single equivalent wheel placed in the symmetry plane. Rodríguez Licea et al. (RODRÍGUEZ LICEA et al., 2018) adopt precisely this approach for a delta tricycle and use it as the basis for a dynamic single-track model in which longitudinal forces, lateral tyre forces, and yaw motion are explicitly modelled. The same authors define a normalised *rollover index* (RI) based on the load distribution between the rear wheels, which provides a scalar indicator of lateral stability that can be embedded in a control law. A comparable derivation is presented for tilting three-wheelers in (EDELMANN et al., 2011), although in that case the relevant degree of freedom is the lean angle rather than the load transfer.

The use of a single-track model implies several assumptions that must be examined critically before the model is applied to the rickshaw. First, the model neglects roll motion of the body around its longitudinal axis, which is acceptable as long as the vehicle does not approach its rollover limit. Second, lateral tyre forces are typically described by a linear function of the slip angle, which holds at low lateral acceleration but degrades close to the friction limit. Third, the geometry assumes that the rear wheels follow the same path as the equivalent rear wheel in the symmetry plane, which is exact at zero speed but only approximate when the wheelbase and track width are comparable, as in the rickshaw.

For the low-speed urban operating domain considered in this thesis, these assumptions are reasonable. The vehicle is expected to operate at speeds well below those at which lateral tyre forces saturate, and the manoeuvres of interest are gentle turns and lane-changes rather than aggressive evasive actions. The single-track kinematic model is therefore retained as the baseline representation, with a dynamic extension reserved for the analysis of stability-related phenomena.

2.2.2. Dynamic Models with Roll and Lateral Load Transfer

When the analysis must include the risk of rollover or the influence of lateral load transfer on tyre forces, the kinematic single-track model is no longer sufficient. The lateral–yaw–roll bicycle model is the standard tool for this case and is used for instance in (LI et al., 2017; AKAR and DERE, 2014), where the rollover risk is quantified through the load transfer ratio (LTR), defined as the difference between the vertical loads on the left and right wheels normalised by their sum. An LTR magnitude approaching unity indicates that one side of the vehicle is about to lift off the ground, and is therefore a direct measure of impending rollover.

Several control architectures exploit a predicted value of LTR, or an equivalent rollover-related index, as a constraint within a model predictive controller or as a trigger for differential braking. Li et al. (LI et al., 2017) propose a three-dimensional dynamics control framework that combines lateral stability and rollover prevention through active braking, while Akar and Dere (AKAR and DERE, 2014) introduce a switching rollover controller coupled with online identification of the vehicle parameters. In the specific context of three-wheeled vehicles, Rodríguez Licea et al. (RODRÍGUEZ LICEA et al., 2018) apply a similar idea by designing a rear differential braking strategy whose activation is based on the predicted RI of a delta tricycle.

The rollover model underlying the controller of Odenthal et al. (ODENTHAL et al., 1999), illustrated in Figure 2.1, separates the vehicle into an unsprung mass m_1 with centre of gravity CG_1 (axle, wheels, and chassis) and a sprung mass m_2 with centre of gravity CG_2 (vehicle body), which rolls about a dedicated roll axis located at height h_R above the ground. The roll angle ϕ displaces CG_2 laterally, amplifying the overturning moment relative to a rigid-body assumption. The tyre vertical loads $F_{z,L}$ and $F_{z,R}$ are the primary outputs used to define the rollover coefficient.

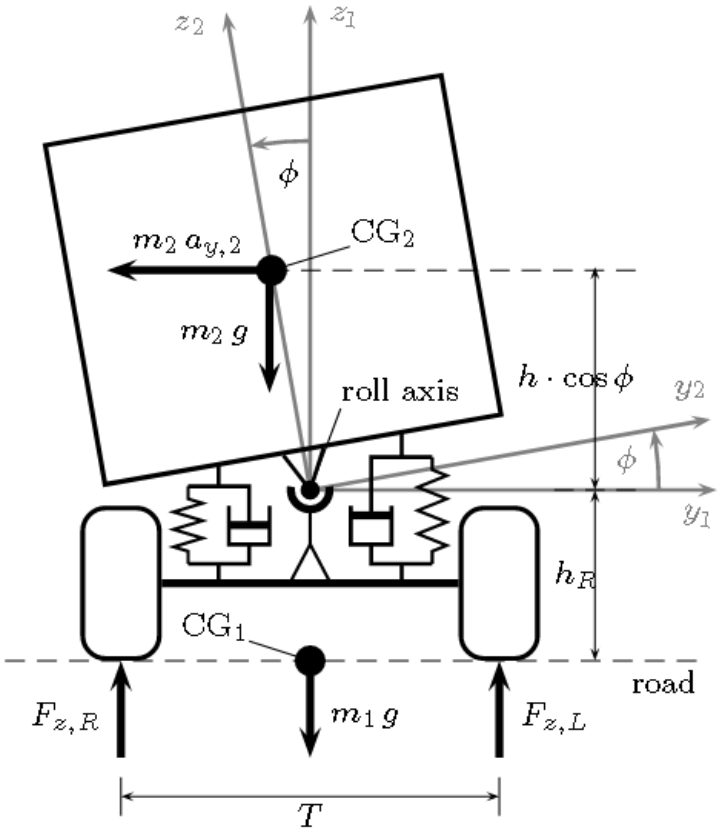

Figure 2.1: Two-mass vehicle rollover model: unsprung mass m_1 at CG_1, sprung mass m_2 at CG_2, roll angle ϕ about the roll axis at height h_R, track width T, and tyre vertical reactions $F_{z,L}$ and $F_{z,R}$ (source: (ODENTHAL et al., 1999)).

From the moment balance about the tyre contact patches and the equilibrium of vertical forces, Odenthal et al. (ODENTHAL et al., 1999) define the rollover coefficient

$$R \;=\; \frac{F_{z,R} - F_{z,L}}{F_{z,R} + F_{z,L}} \;=\; \frac{2\,m_2}{m\,T}\left[\left(h_R + h\cos\phi\right)\frac{a_{y,2}}{g} \;+\; h\sin\phi\right], \tag{2.1}$$

where $a_{y,2}$ is the lateral acceleration at CG_2, h is the height of CG_2 above the roll axis, and T is the track width. For small roll angles ($\phi \ll 1$) this reduces to the linearised form

$$R \;\approx\; \frac{2\,(h_R + h)}{T}\,\frac{a_{y,2}}{g}, \tag{2.2}$$

which shows that the ratio of the effective CoG height $(h_R + h)$ to the track width T is the dominant geometric parameter governing rollover risk — a result consistent with the rigid-body analysis of Section 3.4.3, where the same ratio h_{cg}/b appears in Equation (3.17). The two-mass model of Odenthal et al. (ODENTHAL et al., 1999) extends this result by capturing the additional overturning contribution of the sprung mass roll displacement ϕ, which is absent in the rigid-body approximation.

For the rickshaw, the integration of such an indicator at design time is attractive, but

its quantitative value is limited by the very low operating speeds and the resulting modest lateral accelerations. The vehicle is unlikely to approach the rollover boundary during its intended use; consequently, the safety supervision adopted in this thesis does not implement a full rollover coefficient controller of the type proposed in Odenthal et al. (ODENTHAL et al., 1999) but rather a hard bound on the steering rate and a conservative soft bound on the lateral acceleration, derived from the rigid-body rollover threshold of Section 3.4.3. This simplified bound corresponds to the linearised form (2.2) under the additional assumption $m_1 \ll m_2$, which collapses the two-mass model to the single rigid body treated in Section 3.4.3. The two-mass formulation of Odenthal et al. (ODENTHAL et al., 1999) is nevertheless retained as a conceptual reference for future extensions of the platform to higher-speed or higher-payload operating conditions.

2.2.3. Implications for the Rickshaw

Taking the literature as a whole, the kinematic single-track model emerges as the natural starting point for the design of a controller intended for low-speed urban operation, and the dynamic single-track model with lateral load transfer as the appropriate framework for analysing stability-related phenomena. Both representations transfer to a three-wheeled vehicle by collapsing the rear axle into an equivalent wheel located on the symmetry plane, as confirmed by (RODRÍGUEZ LICEA et al., 2018; “Stability of Three-Wheeled Vehicles with and without Control System” 2013). The rickshaw inherits this structure and is treated, throughout the remainder of the thesis, as a kinematic single-track vehicle augmented with explicit bounds on its physical actuation.

2.3. Trajectory Tracking Algorithms

Trajectory tracking is the problem of computing, at each control instant, the actuation commands that drive a vehicle to follow a reference path while respecting a prescribed speed profile. The problem is conceptually simple but admits a wide variety of solutions, which differ in the way the reference is parameterised, in the type of feedback used, and in the manner in which physical constraints are handled. A comprehensive survey is provided by Rahman et al. (RAHMAN et al., 2025), who classify trajectory-tracking methods into geometric-based approaches (such as pure pursuit and Stanley control), feedback-based approaches (such as sliding-mode and adaptive controllers), and optimisation-based approaches (such as model predictive control). The survey concludes that model predictive formulations are particularly well suited to constrained tracking, although at the cost of higher computational complexity.

2.3.1. Geometric and Kinematic Tracking Laws

Geometric tracking laws compute a steering command from a small number of geometric quantities, typically a lookahead distance, the position of a target point on the reference path, and the current pose of the vehicle. Their main advantages are simplicity, predictable

behaviour, and the absence of any explicit model of the vehicle dynamics. They are widely used as baseline strategies for low-speed automated platforms.

For three-wheeled platforms in particular, Pandey et al. (PANDEY et al., 2017) demonstrate that a kinematic tracking law combined with low-level PID actuator controllers is sufficient to follow planned paths with small lateral error on a real autonomous front-steering mobile robot. The same idea, in a more general form, underlies the kinematic layer of the two-layer controller proposed in (MARTINS et al., 2008) for a unicycle-like mobile robot, where the geometric component is augmented with an adaptive dynamic compensation that improves robustness to parameter variations and external loads.

The principal limitation of purely geometric laws is that they ignore the dynamic behaviour of the vehicle. At low speed this limitation is not critical, because the dynamics are dominated by kinematics, but at higher speeds or in the presence of actuator delays the controller may fail to anticipate the response of the system and produce overshoots or oscillations. For the rickshaw, which operates at low speed and with a limited steering range, the geometric family is therefore retained as a reasonable baseline.

2.3.2. Linear and Nonlinear Optimal Control

A second family of trajectory tracking strategies relies on optimal control and explicitly incorporates a model of the vehicle dynamics. The classical example is the linear quadratic regulator (LQR), which is obtained by linearising the vehicle equations along the reference path and minimising a quadratic cost function. Divelbiss and Wen (DIVELBISS and WEN, 1997) apply this idea to a car–trailer system, designing a hierarchy in which a time-varying LQR provides high-level corrections and PID loops handle low-level actuation. The reported experimental results show that the actual trajectories follow the desired paths with very small deviations.

Beyond the LQR, more recent work explores model predictive control (MPC) as a unified framework that combines dynamic prediction, optimisation, and explicit handling of constraints. In the specific case of urban autonomous driving, Dempster et al. (DEMPSTER et al., 2023) formulate a unified nonlinear MPC that jointly performs trajectory planning and tracking under road, obstacle, and comfort constraints, using a nonlinear kinematic bicycle model. The reported results show that the controller is able to compute, in real time, feasible and collision-free trajectories that are also comfortable for the passengers.

The attractiveness of MPC for the rickshaw is twofold. First, the explicit handling of constraints — steering range, steering rate, acceleration bounds — naturally accommodates the physical limitations of the platform. Second, the predictive nature of the controller can compensate for the actuator delays that are unavoidable in a steer-by-wire system. The cost is computational: a full nonlinear MPC requires the solution of a non-convex optimisation problem at each control step, which is not always feasible on the embedded hardware available on the vehicle. For this reason, the practical implementation adopted in the thesis combines MPC for the lateral channel with a simpler PID structure for the longitudinal channel, mirroring the architecture provided by the Autoware stack.

2.3.3. Robust and State-of-the-Art Approaches

A third family of methods can be loosely described as robust or state-of-the-art. The work of Beal and Gerdes (BEAL and GERDES, 2013) introduces the concept of a handling envelope, defined in the yaw rate–slip angle plane, and uses MPC to keep the vehicle inside this envelope at the limits of handling. The hybrid MPC of Bernardini et al. (BERNARDINI et al., 2009) addresses yaw stabilisation in drive-by-wire passenger cars and shows that closed-loop predictive control prevents loss of stability where an open-loop bicycle model would diverge.

A particularly instructive contribution from the perspective of this thesis is provided by Kolter et al. (KOLTER et al., 2010), who study extreme autonomous driving manoeuvres and explicitly compare pure open-loop replay of demonstrated commands with mixed open-loop / closed-loop strategies based on multi-model LQR. The authors report that pure open-loop replay fails to reach the target, while the mixed strategy robustly completes the manoeuvre even in the presence of model errors and noise. This result, although obtained on a passenger car at the limit of friction, is conceptually transferable to any vehicle whose model is uncertain — including the rickshaw — and provides one of the strongest justifications in the literature for the systematic use of feedback in trajectory tracking.

For the rickshaw, the present thesis does not attempt to compete with the most sophisticated formulations developed for high-performance passenger cars. The selected strategy reflects, instead, a pragmatic synthesis: a model predictive controller for the lateral channel, drawing on the constraint-handling capability emphasised by (RAHMAN et al., 2025; DEMPSTER et al., 2023), a PID controller for the longitudinal channel, in line with the two-level structure suggested by (LUU and LUPU, 2019), and an explicit safety supervision layer that incorporates the limits identified in the dynamic literature.

2.4. Low-Level Control for Steer-by-Wire Systems

A trajectory tracking controller produces a desired steering angle. Translating this set point into a physical motion of the front wheel is the responsibility of a low-level steering controller acting on a steer-by-wire (SbW) system. In a SbW architecture, the mechanical link between the steering wheel and the front wheels is replaced by an electrical interface: the desired steering angle is transmitted as an electrical signal to an electronic control unit, which drives a steering actuator and closes a local position loop using feedback from a steering angle sensor. The principle is described in detail by Jianmin et al. (JIANMIN et al., 2010) and reviewed by Perdana et al. (PERDANA et al., 2024), the latter providing a systematic overview of control strategies adopted for SbW vehicles.

2.4.1. Control Strategies Reported in the Literature

The review of Perdana et al. (PERDANA et al., 2024) classifies SbW control strategies into five families: proportional–integral–derivative (PID) control, sliding mode control (SMC), model predictive control (MPC), fuzzy control, and learning-based control. Each family is

associated with a specific set of advantages and limitations, which can be summarised as follows.

c

headercolor **Strategy**	**Advantages**
PID	Simple to tune; accurate tracking under nominal conditions; fast response w
SMC	High tracking precision; robust to disturbances and varying road conditions; improves
MPC	Manages uncertain dynamics; eliminates chattering; explicit constraint handling; useful fo
Fuzzy	Adapts to conditions difficult to model explicitly; handles nonlinear behav
Learning-based	Adapts to a wide range of operating conditions; can improve over time

Table 2.2: Comparison of steer-by-wire control strategies surveyed in (PERDANA et al., 2024).

PID controllers are simple, easy to tune, and provide accurate reference tracking under nominal conditions. Their main weakness is the difficulty of accommodating nonlinearities and model uncertainties without retuning, and their tendency to develop integral windup when the actuator saturates. Sliding mode controllers offer high tracking precision and strong robustness to disturbances and varying road conditions, but suffer from chattering of the control signal, which is detrimental to the actuator. Model predictive controllers manage uncertain dynamics, eliminate chattering, and provide an effective framework for safety-related backup and constraint handling, at the price of a high computational cost. Fuzzy and learning-based controllers add the capability of adapting to operating conditions that are difficult to model explicitly, but require careful design or extensive training data.

A particularly relevant contribution for the rickshaw is the integral-partition PID strategy proposed for an automotive SbW test rig in (*Research of Automotive Steer-by-Wire Control Based on Integral Partition PID Control* n.d.). The objective of the controller is to make the steering wheel return to the centred position quickly, smoothly, and without overshoot. The

strategy is based on the introduction of an error threshold ε: when the magnitude of the error exceeds ε, only the proportional and derivative actions are active, which provides a fast response without exciting the integral term and risking overshoot during large corrections; when the error falls below ε, the full PID is enabled, which removes the steady-state error and improves the final accuracy. The reported return-to-centre test, performed by displacing the steering wheel by ten degrees and releasing it, indicates that a conventional PID exhibits a noticeable overshoot and a settling time of approximately 2.4 s, while the integral-partition PID produces no overshoot and a settling time of approximately 0.63 s.

This strategy is directly transferable to the rickshaw and is identified, in the literature review presented in the kick-off material of the thesis, as the main candidate for the lateral overshoot reduction of the SbW steering loop. Its integration into the implemented control architecture is discussed in Chapter 4.

2.4.2. Longitudinal Control and Braking

The longitudinal channel of an automated vehicle is conceptually simpler than the lateral one but requires its own dedicated structure. A widely adopted approach is a two-level architecture, similar to that proposed by Luu and Lupu (LUU and LUPU, 2019) for adaptive cruise control vehicles. An outer longitudinal controller computes a desired acceleration or speed from the trajectory and from any spacing requirement; an inner actuator-level controller translates the desired acceleration into a torque command for the propulsion motor and into a pressure or position command for the friction brake. This two-level structure is also visible in the spacing control law plus PID throttle of (LUU and LUPU, 2019) and, more generally, in the longitudinal pipelines of automotive ADAS systems.

In an electric drivetrain, the propulsion motor can be operated in regenerative mode during deceleration, which provides both energy recovery and a smoother braking characteristic. The role of the friction brake is then restricted to the high-deceleration regime and to safety-critical situations, in which the response time of the regenerative loop is insufficient.

For the rickshaw, the selected longitudinal architecture follows the same two-level template. The outer loop is a PID controller acting on the speed error, with explicit limits on the commanded acceleration and on the rate of change of the velocity command. The inner loop, executed on the embedded micro-controller via the micro-ROS interface, translates the velocity command into a motor torque and, when required, into a brake activation. The detailed design and tuning of these controllers are presented in Chapter 4.

2.5. Feedback Strategies: Open-Loop vs. Closed-Loop

The choice between open-loop and closed-loop feedback strategies is one of the foundational decisions of any control design. In the trajectory tracking context, an open-loop strategy consists in replaying, at execution time, the commands generated by a planner or

recorded from a demonstration, without correcting them based on the actual state of the vehicle. A closed-loop strategy, in contrast, uses sensor measurements of position, heading, speed, and possibly other variables to compute corrections and drive the actual trajectory toward the reference.

The intuitive trade-off is well known. Open-loop strategies are simple to implement and easy to interpret, but they rely on the assumption that the vehicle and its environment behave exactly as anticipated; any model error, disturbance, or unmodelled delay accumulates over time and produces a drift between the planned and the actual trajectory. Closed-loop strategies are more complex but tolerate disturbances and modelling errors, since the feedback loop continuously corrects the deviations.

ccc

headercolor **Feature**	**Open-loop**	**Closed-loop**
Tracking accuracy	Lower	Higher
Disturbance rejection	Poor	Excellent
Implementation complexity	Simple	More complex
Robustness to model errors	None	High

Table 2.3: Comparison of open-loop and closed-loop feedback strategies for trajectory tracking.

The empirical confirmation of this trade-off is provided by Kolter et al. (KOLTER et al., 2010) in the context of extreme autonomous driving. The authors compare a pure open-loop replay of a demonstrated manoeuvre with a mixed open-loop / closed-loop strategy based on multi-model LQR, on a passenger car operating near the limits of friction. Pure open-loop replay fails to reach the target, even when the demonstration is accurate, because small differences in initial conditions, tyre behaviour, and surface friction accumulate over the duration of the manoeuvre. The mixed strategy, in contrast, robustly completes the manoeuvre despite the same model errors and noise. The same conclusion is reached, in less extreme regimes, by all the trajectory tracking studies considered in Section 2.3.

For the rickshaw, the relevance of these results is direct. The vehicle operates with a non-negligible actuator delay, with sensors of limited accuracy, and on surfaces whose properties (asphalt, paving stones, light gravel on bicycle paths) cannot be fully characterised in advance. A pure open-loop strategy would therefore be inappropriate as the final control solution. The architecture proposed in Chapter 4 is consequently a closed-loop one, with open-loop tests retained only as a baseline for comparison and for the analysis of specific failure modes.

2.6. Physical Constraints and Their Impact on Control

The physical constraints of a vehicle are not implementation details to be addressed at the end of the design process; they shape the very space of trajectories that the controller can execute and, ultimately, the set of operational scenarios in which the vehicle can be deployed. For the rickshaw, four families of constraints are particularly relevant: the limited steering range, the low operating speed, the bounds on actuator behaviour, and the stability-related limits associated with the specific three-wheeled layout.

c

headercolor **Constraint**	**Physical origin**	**Control implication**
Steering range $\|\delta\| \leq \delta_{\max}$	Front fork geometry and cabin volume.	Limits feasible trajectories; must be enforced in planner and controller.
Low operating speed	Urban context; current safety policy.	Amplifies actuator delay and friction effects; kinematic model remains valid.
Actuator limits	Motor torque, steering rate, brake capacity.	Enforced via saturation, rate limiting, or explicit MPC constraints.
Stability-related limits	Three-wheeled delta layout; centre-of-gravity height.	LTR/RI bounds on lateral acceleration; enforced by the safety supervision layer.

Table 2.4: Physical constraint families relevant to the rickshaw and their control implications.

2.6.1. Limited Steering Range

The maximum steering angle of the rickshaw is mechanically restricted, both by the geometry of the front fork and by the cabin volume that surrounds the steering column. The corresponding bound, denoted $|\delta| \leq \delta_{\max}$, is a hard constraint that the controller must respect at all times. From a practical standpoint, this bound translates into a minimum turning radius and, therefore, into a minimum admissible curvature for the reference paths produced by the planner. A trajectory that requires steering angles beyond $\delta_{\max}$ is simply infeasible, regardless of the controller used.

The literature on three-wheeled vehicles consistently identifies the steering range as one of the parameters that most strongly influence controllability and stability. The experimental study reported in ("Stability of Three-Wheeled Vehicles with and without Control System" 2013) shows that, for a non-tilting three-wheeled vehicle, comparatively small changes in steering angle, speed, or load distribution can shift the vehicle from a stable to an unstable behaviour, with strong over- and understeer tendencies. These observations underline the importance of explicitly representing the steering bound, both at the planning stage and within the controller, and of avoiding control laws that rely on saturating the steering input as a primary mechanism for tracking corrections.

2.6.2. Low Operating Speed

The rickshaw is intended for typical urban speeds — broadly, in the range of 0 to 20–30 km/h, with most operation taking place well below the upper bound. Low speed has two distinct effects on the control problem.

First, low speed simplifies the dynamics. The lateral tyre forces are far from their saturation, the centripetal accelerations associated with normal turns remain modest, and the kinematic single-track model retains an excellent predictive value. The risk of rollover is correspondingly small for the manoeuvres that the vehicle is expected to execute. Second, low speed amplifies the relative importance of actuator delays and friction. A delay of 100 ms, which is negligible at highway speeds in terms of distance travelled, corresponds to several centimetres at low speed and may become the dominant source of tracking error. Static friction in the steering mechanism, which is overcome almost instantaneously at speed, can cause noticeable hysteresis when the steering command varies slowly.

These effects have been documented in the steer-by-wire literature (PERDANA et al., 2024) and are explicitly addressed in the architecture proposed for the rickshaw through a delay-aware MPC formulation and through smoothing of the steering command, as discussed in Chapter 4.

2.6.3. Actuator Limits

Beyond the steering range, the actuators of the rickshaw are subject to additional bounds: a maximum steering rate, a maximum acceleration and deceleration in the longitudinal channel, and a maximum motor torque. These bounds must be respected by the controller, both for hardware integrity and for passenger comfort. The classical tools to enforce them are saturation, rate limiting, and, in optimisation-based controllers, explicit inequality constraints. The MPC formulation adopted in the thesis incorporates the steering rate and acceleration bounds directly as constraints in the underlying optimisation problem.

2.6.4. Stability-Related Limits

The final family of constraints concerns the stability of the vehicle at the limit of its lateral capacity. As discussed in Section 2.2, the relevant indicators in the literature are the load

transfer ratio (LTR) and the rollover index (RI). For a low-speed urban platform, the practical risk of rollover is small, but it is not negligible at the highest admissible speeds combined with the tightest admissible curvatures. A simple safety supervision layer that limits the lateral acceleration, and through it the combination of speed and curvature, is therefore included in the architecture. This layer does not replace the more sophisticated braking-based rollover controllers proposed in (RODRÍGUEZ LICEA et al., 2018; LI et al., 2017; AKAR and DERE, 2014), but provides a conservative bound that is sufficient for the operating domain considered.

2.7. Research Gap Identification

The literature reviewed in the previous sections covers, individually, all the building blocks required for the control of the rickshaw: vehicle models for three-wheeled platforms, trajectory tracking algorithms, low-level steer-by-wire control, longitudinal control architectures, and stability supervision strategies. What is missing, and what the present thesis aims to provide, is the integration of these blocks into a single coherent architecture, designed for, and validated on, a non-tilting three-wheeled vehicle intended for low-to-moderate-speed operation on cycling and road infrastructure.

This research gap can be made explicit along three complementary dimensions. The first concerns the vehicle type and intended use case, for which no existing work addresses a platform with the specific characteristics of the rickshaw. The second concerns the absence of an integrated control architecture that combines trajectory execution, steer-by-wire control, and stability supervision in a single coherent design. The third concerns the lack of experimental validation of such an architecture on a road-legal, semi-autonomous three-wheeled vehicle operating on shared urban infrastructure.

Vehicle type and use case. The bulk of the existing work targets either passenger cars, trucks, tilting three-wheelers, or small mobile robots. No published study, to the best of the author's knowledge, addresses a non-tilting three-wheeled vehicle conceived as a passenger and freight platform that uses both cycle lanes and ordinary roads. The closest references are the delta tricycle of (RODRÍGUEZ LICEA et al., 2018), which is treated from a stability and braking perspective, and the front-steering mobile robot of (PANDEY et al., 2017), which is much smaller and operates in indoor or strictly controlled environments.

Missing integrated control architecture. The literature consistently treats trajectory tracking, low-level SbW control, and stability supervision as separate problems. Trajectory tracking studies (PANDEY et al., 2017; DIVELBISS and WEN, 1997; MARTINS et al., 2008; DEMPSTER et al., 2023) rarely consider rollover or stability indices; SbW studies (JIANMIN et al., 2010; PERDANA et al., 2024; *Research of Automotive Steer-by-Wire Control Based on Integral Partition PID Control* n.d.) focus on the steering actuator and do not extend their scope to full trajectory execution; stability and rollover studies (RODRÍGUEZ LICEA et al., 2018; LI et al., 2017; AKAR and DERE, 2014) concentrate on the safety layer in isolation. A single architecture combining these three components on a three-wheeled vehicle is not present in the surveyed literature.

Experimental validation gap. Many of the most advanced controllers identified in the literature — hybrid MPC (BERNARDINI et al., 2009), handling-envelope MPC (BEAL and GERDES, 2013), three-dimensional dynamics MPC (LI et al., 2017), unified NMPC for urban driving (DEMPSTER et al., 2023) — have been validated only in simulation, in hardware-in-the-loop benches, or on dedicated test tracks. Experimental work on real platforms is mostly limited to small mobile robots (PANDEY et al., 2017; MARTINS et al., 2008) or to specific tricycle prototypes (RODRÍGUEZ LICEA et al., 2018). The validation of an integrated control architecture on a road-legal, semi-autonomous cycle rickshaw is therefore an open contribution.

The three dimensions identified above directly motivate the architecture presented in Chapter 4, whose design choices are systematically grounded in the gaps identified here.

3. Vehicle Platform and System Description

The control architecture developed in this thesis is designed for and tested on a specific platform: the semi-autonomous cycle rickshaw of the Chair of Traffic Engineering and Control of the Technical University of Munich. A precise understanding of the vehicle, of its hardware components, and of the software stack that supports its operation is therefore a prerequisite for the discussion of the control design and of the experimental results. This chapter is dedicated to that description.

Section 3.1 summarises the mechanical layout and the main physical parameters of the vehicle. Section 3.2 details the hardware architecture, organised in four subsystems: the sensor suite, the on-board computing units, the actuation system, and the power and communication infrastructure. Section 3.3 describes the software architecture, with particular attention to the Autoware stack, the ROS 2 integration, and the dedicated control bridge that mediates between the high-level controller and the embedded actuation interface based on micro-ROS. Section 3.4 develops the mathematical model of the vehicle kinematic, dynamic, and rollover geometry that underpins the control architecture presented in Chapter 4.

3.1. Vehicle Specifications and Mechanical Overview

The platform is a three-wheeled, fully electric cycle rickshaw with one steering wheel at the front and two driven wheels at the rear, in a so-called delta configuration. The body consists of a moulded shell that encloses the driver's compartment and the rear cargo or passenger area, and is mounted on a tubular frame derived from a commercial cargo-cycle chassis. The visual appearance of the vehicle is given in Figure 1.1 of Chapter 1.

The principal mechanical and operational parameters relevant for control are summarised below. The dominant geometric parameter for the kinematic single-track model is the wheelbase, defined as the longitudinal distance between the front wheel and the rear axle; together with the maximum steering angle $\delta_{\max}$, it determines the minimum turning radius. The track width, the lateral distance between the two rear wheels, governs lateral load transfer and the speed differential between inner and outer wheels during turns. The mass and centre-of-gravity height of the empty vehicle, together with the payload, determine the vertical load on each tyre and hence the available friction.

The maximum operating speed considered in this thesis is set conservatively to 1.66 m/s (approximately 6 km/h) throughout the validation campaign for safety reasons. This is a de-

liberate restriction: the controller and the vehicle are capable, in principle, of higher speeds, but the validation campaign is designed to demonstrate stable behaviour at low speed before any extension to faster regimes. The maximum steering angle is bounded mechanically and conservatively limited, in software, to ± 0.70 rad (approximately $\pm 40^{\circ}$), in line with the value used in the steering bridge described in Section 3.3.

Parameter	**Symbol**	**Value**	**Notes**
Wheelbase	L	2.74 m	Front wheel to rear axle centre
Track width	b	1.10 m	Between rear wheel centres
Vehicle mass (empty)	m	≈ 120 kg	Estimated; includes all on-board hardware
CoG height (empty)	h_{cg}	≈ 0.75 m	Estimated from mass-weighted component positions
Wheel radius	r	0.311 m	Diameter 622 mm; confirmed by firmware
Max. steering angle	$\delta_{\max}$	0.70 rad ($\approx 40.1^{\circ}$)	Software and mechanical limit
Max. operational speed	$v_{\max}$	1.66 m/s (≈ 6 km/h)	Current validation campaign limit

Table 3.1: Principal mechanical and operational parameters of the rickshaw platform.

3.2. Hardware Architecture

The hardware architecture of the rickshaw is structured around a central on-board computer running the Autoware stack, an embedded micro-controller dedicated to actuator interfacing, a multi-sensor perception suite, and an electric drivetrain with steering and braking actuators. The role of each subsystem is described in the following paragraphs.

3.2.1. Sensor Suite

The sensor suite combines exteroceptive sensors, used by the perception and localisation pipelines of Autoware, and proprioceptive sensors, used by the control layers to monitor the state of the vehicle. The full description of the perception suite is beyond the scope of this thesis, which focuses on the control loop; only the proprioceptive sensors directly involved in the closed-loop control are detailed here.

Wheel speed sensing. The vehicle is equipped with two Hall-effect wheel-speed sensors of type Infineon TLE4922[1], one mounted on each rear wheel. The choice of the TLE4922 is

[1] `https://www.infineon.com/cms/en/product/sensor/magnetic-sensors/magnetic-position-sensors/hall-switches/tle4922/`

driven by three considerations: it is an active sensor, which allows it to operate down to zero rotational speed; it produces a digital square-wave output that can be processed directly by a digital input of the embedded micro-controller without dedicated signal conditioning; and it is robust to dust and humidity, which are common conditions on shared bicycle infrastructure. Each sensor is paired with a 24-slot ferromagnetic encoder ring mounted on the wheel hub, with 22 of the 24 positions occupied by magnets ($N_{\text{eff}} = 22$), so that one full revolution generates 22 pulses on the corresponding sensor output.

The full technical characterisation, firmware implementation, and mechanical integration are presented in Section 4.6.1, where the sensor selection rationale, the interrupt-driven pulse acquisition, the speed computation algorithm, and the slip-detection mechanism are described in detail.

Steering angle sensing. The steering angle is measured by an Infineon TLE5012B[2] sensor, a Giant Magneto-Resistance (GMR) device that provides absolute angular position over the full $\pm 180^\circ$ range with a 15-bit resolution (approximately 0.011° per LSB) and a typical accuracy of $\pm 1^\circ$. The choice of an absolute sensor avoids the need for a homing procedure at start-up and removes the risk of accumulated error that affects incremental encoders. The sensor is read by a dedicated ESP32 micro-controller through a four-wire SSC interface. The signal conditioning, calibration procedure, and anomaly diagnosis are described in Section 4.6.3.

Other proprioceptive sensors. An inertial measurement unit (IMU) is integrated within the on-board computing chassis and provides angular rate and linear acceleration measurements that are used by the Autoware Extended Kalman Filter (EKF) localiser. The fusion of wheel speed, steering angle, and IMU information within the EKF produces the pose and twist estimates used by the trajectory tracker. The detailed configuration of the EKF is provided by the Autoware default parameters and is not modified in the present work.

3.2.2. Computing Units

The on-board computing infrastructure is organised in two layers. The high-level computing unit is a LattePanda[3] single-board computer, selected for its compact form factor, its compatibility with the Ubuntu / ROS 2 stack required by Autoware, and its sufficient computational performance for the planning and control modules at the operating speeds considered. The full Autoware stack runs on this computer, including the perception, planning, and control nodes, together with the dedicated control bridge described in Section 3.3.

The low-level computing layer consists of an ESP32-EVB[4] micro-controller, dedicated to actuator interfacing and to the acquisition of the wheel-speed and steering-angle signals. The ESP32 runs a micro-ROS[5] firmware that exposes the actuators and sensors as

[2] https://www.infineon.com/cms/en/product/sensor/magnetic-sensors/magnetic-position-sensors/angle-sensors/tle5012b/

[3] https://www.lattepanda.com

[4] Olimex ESP32-EVB open-source hardware board. https://www.olimex.com/Products/IoT/ESP32/ESP32-EVB

[5] https://micro.ros.org

ROS 2 topics, accessible from the Autoware side through a UDP transport on port XXXX. This division of responsibilities is consistent with the recommendations of the micro-ROS community and ensures that hard real-time tasks — pulse counting, SSC frame decoding, actuator command generation — are executed on a deterministic platform, while the more demanding but less time-critical computations of perception and planning are executed on the high-level computer.

3.2.3. Actuation System

The actuation system comprises three components: a steering actuator that converts the desired steering angle into a physical rotation of the front wheel, an electric motor that drives the rear axle, and a friction brake for deceleration and immobilisation. The steering channel is implemented as a steer-by-wire system in the sense discussed in Section 2.4: both the driver's handlebar during manual operation and the high-level controller during autonomous operation provide a desired steering angle as an electrical signal, which is tracked by a local angle loop on the steering actuator using feedback from the TLE5012B sensor.

3.2.4. Power and Communication

The vehicle operates with two independent battery packs. The first supplies the propulsion motor, the steering actuator, and the associated power electronics; the second, electrically isolated from the first, supplies the LattePanda, the sensors (LiDAR, GNSS, IMU), the ESP32 micro-controller, and the communication hardware. This separation ensures that high-current transients from the drivetrain do not propagate to the computing and sensing subsystems. Communication between the high-level computer and the embedded micro-controller is handled by the micro-ROS bridge over UDP. The internal ROS 2 communication uses the default DDS implementation provided by Autoware.

The hardware components and the two-layer architecture described above are illustrated in Figures 1.2 and 1.3 of Chapter 1.

3.3. Software Architecture

The software architecture of the rickshaw is built on top of the Autoware open-source stack, which provides a complete pipeline for autonomous driving, from perception to actuation. The role of the present work, in software terms, is twofold: to provide a controller that can be inserted into this pipeline and that is appropriate for the specific characteristics of the rickshaw, and to develop the dedicated bridge that connects the Autoware control output with the embedded actuation interface based on micro-ROS.

3.3.1. Autoware Stack

Autoware is structured around a set of stacks that mirror the classical autonomous driving architecture: perception, localisation, planning, control, and vehicle interface. In the configuration adopted on the rickshaw, the control stack publishes a desired control command on the topic `/control/command/control_cmd`, in the form of an `AckermannControlCommand` message that contains, in particular, a desired longitudinal speed, a desired longitudinal acceleration, and a desired steering angle. The vehicle interface, which is the part of Autoware responsible for translating these commands into the specific protocol of the underlying vehicle, is in the present work replaced by a dedicated control bridge, described below.

The rationale for this design is to maintain compatibility with the standard Autoware interfaces — so that improvements in perception, localisation, and planning modules can be adopted without modifying the controller — while keeping full control over the actuation chain, which is the part of the system most affected by the specific hardware of the rickshaw.

3.3.2. ROS 2 Integration

All software components on the rickshaw communicate through ROS 2 topics[6]. The integration of the three bridge nodes with the Autoware ROS 2 graph is shown in Figures 3.1–3.3. These nodes do not exist in a standard Autoware deployment and constitute the software interface contribution of this thesis.

Control bridge. Figure 3.1 shows the control bridge `/rickshaw_control_bridge`: it subscribes to `/control/command/control_cmd` published by the Autoware control module and forwards the actuation command to the ESP32 firmware on `/robot/data_in`. It additionally publishes the four vehicle status topics required by Autoware (`control_mode`, `gear_status`, `hazard_lights_status`, `turn_indicators_status`) with fixed stub values, since these signals are not physically present on the rickshaw.

[6] `https://docs.ros.org`

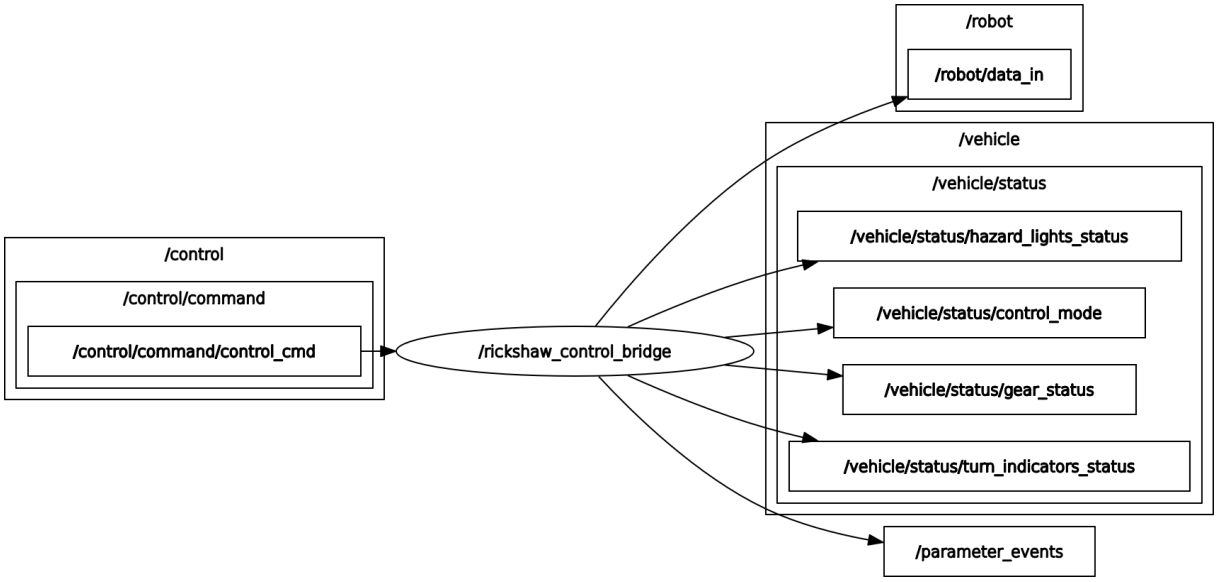

Figure 3.1: ROS 2 graph of the control bridge node: subscription to the Autoware actuation command and publication to the ESP32 topic `/robot/data_in` and the required Autoware vehicle status topics (source: own work, captured via `rqt_graph`).

Speed bridge. Figure 3.2 shows the speed bridge `/rickshaw_speed_bridge`: it subscribes to the micro-ROS topic `/robot/speed_out` and publishes `/vehicle/status/velocity_status` for the Autoware EKF and longitudinal controller. Four additional diagnostic topics (`/wheel_speed/ms_left`, `/wheel_speed/ms_right`, `/wheel_speed/kmh`, `/wheel_speed/slip`) are published for monitoring purposes during the validation campaign.

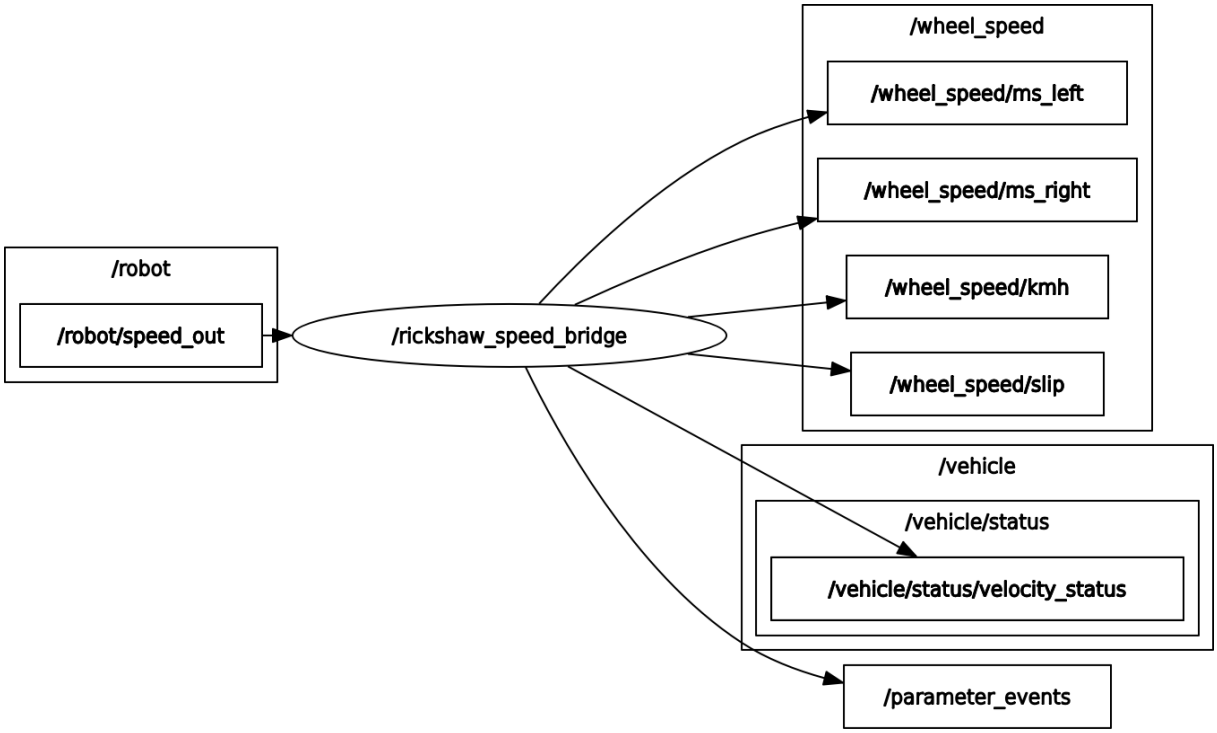

Figure 3.2: ROS 2 graph of the speed bridge node: subscription to the ESP32 wheel-speed topic and publication to `/vehicle/status/velocity_status` and four diagnostic topics (source: own work, captured via `rqt_graph`).

Steering bridge. Figure 3.3 shows the steering bridge `/rickshaw_steering_bridge`: it subscribes to `/robot/steer_out` and publishes `/vehicle/status/steering_status` to the Autoware lateral MPC after applying the four-stage filter described in Section 4.6.3.

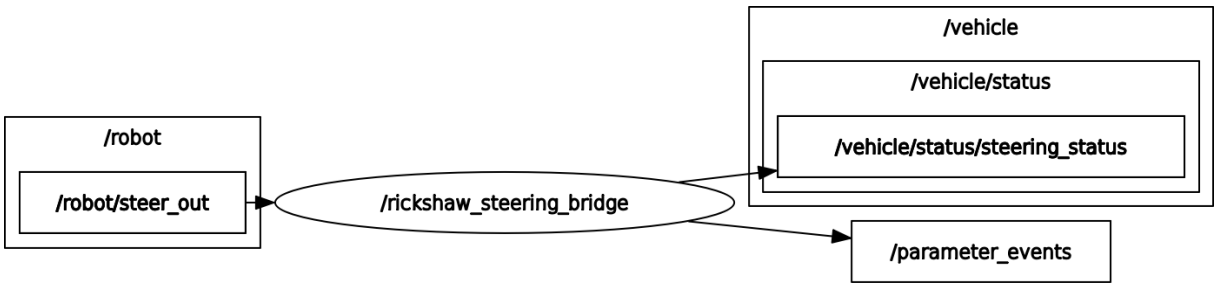

Figure 3.3: ROS 2 graph of the steering bridge node: subscription to the ESP32 steering angle topic and publication to `/vehicle/status/steering_status` (source: own work, captured via `rqt_graph`).

3.3.3. Control Bridge and micro-ROS Interface

This subsection describes the architectural role and interface design of the bridge layer; the full implementation details, including firmware anomaly handling and signal conditioning, are presented in Section 4.6.

The dedicated control bridge implemented for the rickshaw is realised by the Python node `control_bridge.py`. The node subscribes to the Autoware topic `/control/command/control_cmd` and publishes a compact representation of the actuation command on the topic `/robot/data_in`, in the form of a `Float32MultiArray` message containing the desired velocity and the desired steering angle. The micro-ROS firmware running on the ESP32 subscribes to this topic and translates the received values into the corresponding actuator commands.

The choice of a dedicated bridge between Autoware and micro-ROS, rather than a direct connection, is motivated by three considerations. First, it isolates the embedded firmware from any changes in the Autoware control message structure, which has evolved across releases; only the bridge needs to be modified when such changes occur. Second, it allows the application of safety checks — clamping of the commanded values, watchdog on the planner update rate, fall-back to a safe state in case of message loss — at the boundary between the high-level and the low-level systems. Third, it provides a natural place to apply the actuator-consistent limits and the steering smoothing strategies that are part of the control architecture, as discussed in Chapter 4.

In the reverse direction, the embedded sensors (wheel speed and steering angle) are exposed to Autoware through two dedicated nodes: `speed_bridge.py`, which converts the raw wheel-speed pulses from the firmware into a standard Autoware `VelocityReport`, and `steering_bridge.py`, which applies the four-stage signal filter and republishes the calibrated angle as a `SteeringReport`. Both nodes subscribe to the corresponding micro-ROS topics, perform the unit conversions and calibrations, and republish the results on the standard Autoware vehicle status topics. Their full implementation is described in Section 4.6.4.

3.4. Vehicle Model

The control architecture developed in this thesis relies on a mathematical description of the rickshaw that captures its dominant motion properties without introducing parameters that cannot be identified on the physical platform. The choice of the appropriate level of model fidelity is therefore a deliberate engineering decision rather than a default: a more detailed model would expose phenomena that are not relevant in the operating regime considered, while introducing unknown parameters that would degrade rather than improve the predictive quality of the controller. This section presents the modelling framework adopted, derived consistently from the three-wheeled vehicle literature reviewed in Chapter 2, together with the free-body analysis that supports the rollover-related safety bound implemented in Chapter 4.

The presentation is organised in four parts. Section 3.4.1 develops the kinematic single-track model that constitutes the prediction model of the lateral controller. Section 3.4.2 extends it to a dynamic single-track formulation that, while not embedded in the controller, provides the language for the stability analysis. Section 3.4.3 presents the free-body analysis of the rear axle and derives the rollover threshold used by the safety supervisor. Section 3.4.4 summarises the modelling choices effectively used in the thesis.

The three formulations are not redundant alternatives but serve distinct and complementary roles within the architecture. The kinematic single-track model is adopted as the predic-

tion model of the lateral MPC controller because it captures the dominant motion of the vehicle at low speed with a minimal set of identifiable parameters, and its linearity under small-angle approximation makes it compatible with the convex quadratic programme solved at each control step. The dynamic single-track model extends the kinematic one by incorporating lateral tyre forces and yaw dynamics, which are not needed for the controller itself at the operating speeds considered, but provide the analytical framework for discussing understeer, oversteer, and stability margins in the language of the vehicle dynamics literature. The free-body rollover analysis is geometrically simpler than either single-track model but serves a different purpose entirely: it derives a closed-form threshold on the lateral acceleration that is independent of tyre properties and payload, and therefore provides a robust and conservative basis for the safety supervision bound. Using all three formulations together avoids the pitfall of either over-simplifying the stability analysis or over-complicating the controller with parameters that cannot be identified on the physical platform.

3.4.1. Kinematic Single-Track Model

The kinematic single-track (bicycle) abstraction collapses a multi-axle vehicle to two equivalent wheels placed on its longitudinal symmetry plane: an equivalent front wheel that represents the steering axle and an equivalent rear wheel that represents the drive axle. Three considerations justify the use of this abstraction for the rickshaw. First, the rickshaw operates well below the speed at which lateral tyre forces approach saturation, so the kinematic constraints of pure rolling describe the motion accurately. Second, the rear axle of the rickshaw is symmetric about the longitudinal axis, so its two wheels can be replaced by an equivalent wheel at the midpoint of the axle without loss of generality at the level of the trajectory. Third, the bicycle abstraction is the model adopted by the three-wheeled vehicle studies most relevant to the platform, in particular by (RODRÍGUEZ LICEA et al., 2018) for a delta tricycle and by (PANDEY et al., 2017) for a front-steering tricycle robot, and is the model accepted as the standard in the Autoware lateral control implementation.

The variables and the geometry of the model are illustrated in Figure 3.4. In the inertial frame XY, the vehicle pose is described by the coordinates (X, Y) of the centre of gravity and by the heading angle ψ measured between the inertial X axis and the longitudinal body axis. The sub-distances from the centre of gravity to the front and rear axles are ℓ_f and ℓ_r respectively, giving a total wheelbase $L = \ell_f + \ell_r$. The front steering angle δ_f is measured between the front-wheel direction and the body axis; the rear steering angle $\delta_r = 0$ since only the front wheel steers. The longitudinal speed is denoted v, and O denotes the instantaneous centre of rotation (ICR).

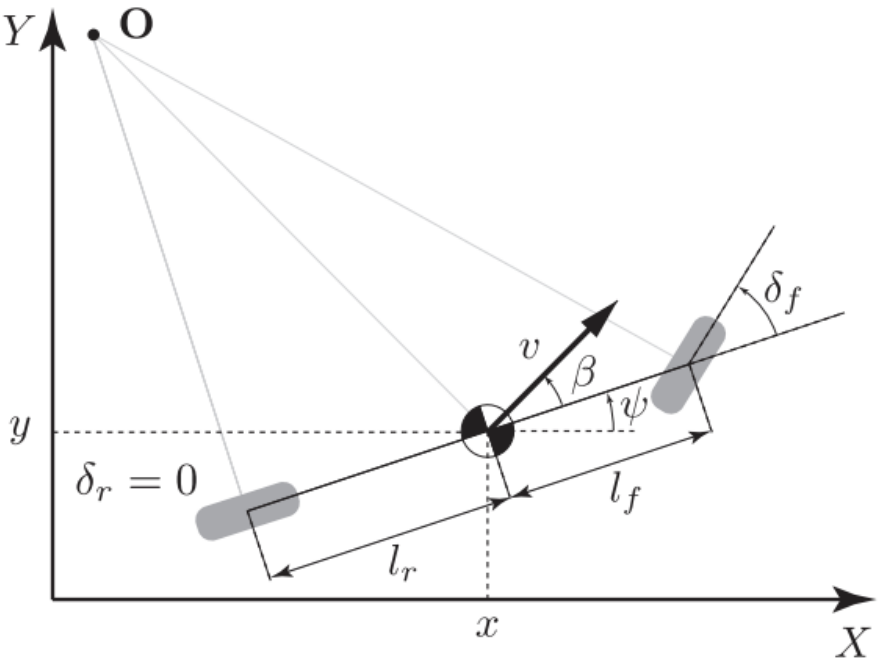

Figure 3.4: Kinematic single-track model in inertial frame XY: centre of gravity with body sideslip β and heading ψ, sub-distances ℓ_f and ℓ_r to front and rear axles, front steering angle δ_f, velocity v, and instantaneous centre of rotation O (source: (KONG et al., 2015)).

In Figure 3.4, following the convention of (KONG et al., 2015), the reference point is the centre of gravity (CoG), with ℓ_f and ℓ_r denoting the distances to the front and rear axles respectively, and $\delta_r = 0$ confirming that only the front wheel steers. Throughout this thesis the equations are expressed at the rear axle contact point rather than at the centre of gravity; under the kinematic no-slip assumption the velocity at the rear axle is aligned with the body axis, which eliminates the body sideslip β from the position equations and yields the simplified form of Equations (3.1)–(3.3).

Under the no-slip assumption, the velocity vectors at the rear axle and at the front contact point are aligned with the planes of the corresponding wheels: the rear velocity is along the body axis and the front velocity is along the plane of the steered wheel. The resulting kinematic equations of motion, derived in standard form in (RODRÍGUEZ LICEA et al., 2018, Section III) and adopted with identical structure in (PANDEY et al., 2017), are:

$$\dot{X} = v\,\cos(\psi), \tag{3.1}$$

$$\dot{Y} = v\,\sin(\psi), \tag{3.2}$$

$$\dot{\psi} = \frac{v}{L}\,\tan(\delta_f). \tag{3.3}$$

Equation (3.3) encodes the central kinematic property of the bicycle model: the yaw rate $\dot{\psi}$ is proportional to the longitudinal speed and to the tangent of the steering angle, inversely scaled by the wheelbase. At zero speed the heading cannot change regardless of the steering input, which reflects the physical behaviour of a non-holonomic wheeled vehicle at

rest.

Two derived quantities follow directly from Equations (3.1)–(3.3) and are relevant for the controller design. The instantaneous turning radius R, defined as the distance from the rear axle to the instantaneous centre of rotation (ICR) illustrated in Figure 3.4, is

$$R \;=\; \frac{L}{\tan(\delta_f)}, \tag{3.4}$$

which approaches infinity as $\delta_f \to 0$ (straight-line motion) and reaches its minimum admissible value $R_{\min} = L/\tan(\delta_{\max})$ at the maximum steering angle. For the rickshaw, with L as given in Section 3.1 and $\delta_{\max} = 0.70$ rad, the minimum turning radius constrains the set of trajectories that the planner can prescribe. The path curvature κ is the reciprocal of the turning radius,

$$\kappa \;=\; \frac{1}{R} \;=\; \frac{\tan(\delta_f)}{L}, \tag{3.5}$$

and is the quantity used by the high-level execution layer to compute the feedforward steering command, as detailed in Section 4.2.

The kinematic model neglects three physical effects that are usually the focus of more sophisticated formulations. The lateral slip of the tyres is set to zero, which holds rigorously only in the limit of vanishing speed and lateral acceleration. The roll motion of the body about its longitudinal axis is suppressed, which is exact for a perfectly rigid platform but only approximate for the flexible cabin of the rickshaw. The flexibility of the suspension and of the tyres themselves is absent, so the load on each wheel is purely geometric. As argued in (RODRÍGUEZ LICEA et al., 2018) and in (PANDEY et al., 2017) for vehicles of comparable size and speed, these omissions are acceptable for the design of a trajectory-tracking controller as long as a separate safety mechanism enforces the boundary of the regime in which they hold; this is the role of the free-body analysis presented in Section 3.4.3.

3.4.2. Dynamic Single-Track Model

When the lateral acceleration is no longer small, or when the analysis is concerned with stability margins rather than with steady motion, the kinematic model must be extended to include the lateral dynamics of the body and the lateral forces generated at the tyre-road contact patches. The dynamic single-track model serves this purpose. It is the formulation adopted in the lateral stability study of tilting three-wheelers in (EDELMANN et al., 2011) and in the non-tilting tricycle analysis of (RODRÍGUEZ LICEA et al., 2018; “Stability of Three-Wheeled Vehicles with and without Control System” 2013); the structure presented below follows these references in their non-tilting form.

The configuration of the model is shown in Figure 3.5.

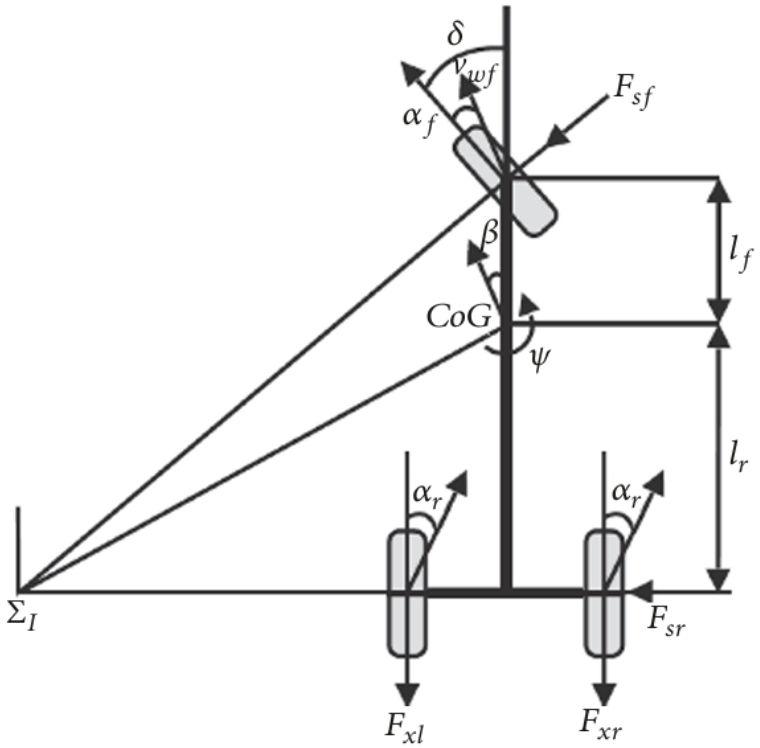

Figure 3.5: Lateral dynamics of the delta tricycle: CoG, distances ℓ_f and ℓ_r, sideslip β, slip angles α_f and α_r, and tyre forces (source: (RODRÍGUEZ LICEA et al., 2018)).

The vehicle is represented as a rigid body with mass m and yaw moment of inertia I_z about the vertical axis through the centre of gravity (CoG). The distance from the CoG to the front and rear axles is ℓ_f and ℓ_r respectively, with $\ell_f + \ell_r = L$. The body-fixed velocity components at the CoG are v_x along the body axis and v_y perpendicular to it. The yaw rate is $r = \dot{\psi}$. In the following, δ denotes the front steering angle δ_f of Section 3.4.1, with the subscript dropped for notational brevity. The principal derived quantities of the dynamic formulation are the body side-slip angle,

$$\beta \;=\; \arctan\!\left(\frac{v_y}{v_x}\right) \;\approx\; \frac{v_y}{v_x}, \tag{3.6}$$

The slip angles of the front and rear wheels, which describe the misalignment between the wheel plane and the velocity vector at the corresponding contact patch, are:

$$\alpha_f \;=\; \delta \;-\; \beta \;-\; \frac{\ell_f\, r}{v_x}, \tag{3.7}$$

$$\alpha_r \;=\; -\beta \;+\; \frac{\ell_r\, r}{v_x}. \tag{3.8}$$

The lateral force generated at each axle is taken proportional to the corresponding slip angle in the linear regime — a standard small-slip approximation accepted in (EDELMANN et al., 2011; RODRÍGUEZ LICEA et al., 2018; “Stability of Three-Wheeled Vehicles with and

without Control System" 2013):

$$F_{sf} \;=\; C_f\,\alpha_f, \qquad F_{sr} \;=\; C_r\,\alpha_r, \tag{3.9}$$

where C_f and C_r are the cornering stiffnesses of the front and rear axles. The corresponding lateral and yaw equations of motion, written in body-fixed coordinates, are

$$m\,(\dot{v}_y \;+\; v_x\,r) \;=\; F_{sf}\,\cos(\delta) \;+\; F_{sr}, \tag{3.10}$$

$$I_z\,\dot{r} \;=\; \ell_f\,F_{sf}\,\cos(\delta) \;-\; \ell_r\,F_{sr}. \tag{3.11}$$

Linearising under the small-angle assumption ($\cos(\delta) \approx 1$, $\sin(\delta) \approx \delta$, $\beta \ll 1$) and substituting the slip-angle expressions of Equations (3.7)–(3.8) into the linear tyre model yields the standard linear bicycle dynamic model in the state $[\beta, r]^\top$, from which the lateral stability of the open-loop vehicle is classically analysed in terms of understeer or oversteer behaviour, as discussed for non-tilting tricycles in ("Stability of Three-Wheeled Vehicles with and without Control System" 2013).

The dynamic model is not embedded as a prediction model in the controllers developed in this thesis, for three reasons. First, the cornering stiffnesses C_f and C_r of the rickshaw tyres have not been characterised experimentally, and adopting nominal automotive values would introduce model errors that exceed the corrections the controller is asked to deliver. Second, the operating speeds considered ($v \leq 1.66$ m/s) are far below the regime in which the dynamic terms become significant: at such speeds the centripetal acceleration $v^2\kappa$ remains a small fraction of the gravity acceleration even at the maximum admissible curvature, so the linear tyre regime is never approached. Third, the predictive value added by the dynamic model would not justify the extra unknown parameters in a controller whose dominant uncertainty is the actuation delay, which is captured by an explicit delay-compensation mechanism rather than by an extended state model. The dynamic model is therefore retained as a conceptual reference for the stability discussion of Chapter 2 and as the appropriate framework for future extensions of the platform to higher operating speeds.

3.4.3. Free-Body Analysis and Rollover Geometry

The third element of the modelling framework concerns the lateral stability of the rickshaw in the limit case in which it approaches the rollover threshold. Although the operating speed of the platform is too low for this regime to be reached under nominal conditions, the safety supervision component of the controller, presented in Section 4.5, enforces a conservative bound on the lateral acceleration that is derived precisely from the analysis developed in this subsection. The treatment follows the rollover geometry of the non-tilting tricycle in (Rodríguez Licea et al., 2018, Section IV) and the equivalent derivation for three-wheeled platforms presented in ("Stability of Three-Wheeled Vehicles with and without Control System" 2013).

The free-body diagram of the rear axle viewed from the front is shown in Figure 3.6. The vehicle is approximated as a rigid body of mass m whose centre of gravity is located at height h_{cg} above the ground and laterally centred between the two rear wheels, which are separated by a track width b. The contact points of the left and right rear wheels with the

ground are denoted P_B and P_C following the convention of Figure 3.6. The body is subject to the gravitational force mg acting vertically downward at the CoG and, during a cornering manoeuvre, to a lateral inertial force ma_y acting horizontally at the CoG, where a_y is the centripetal acceleration. The reactions at the contact patches are the vertical loads F_{ZRL} and F_{ZRR} and the lateral friction forces F_y^L and F_y^R.

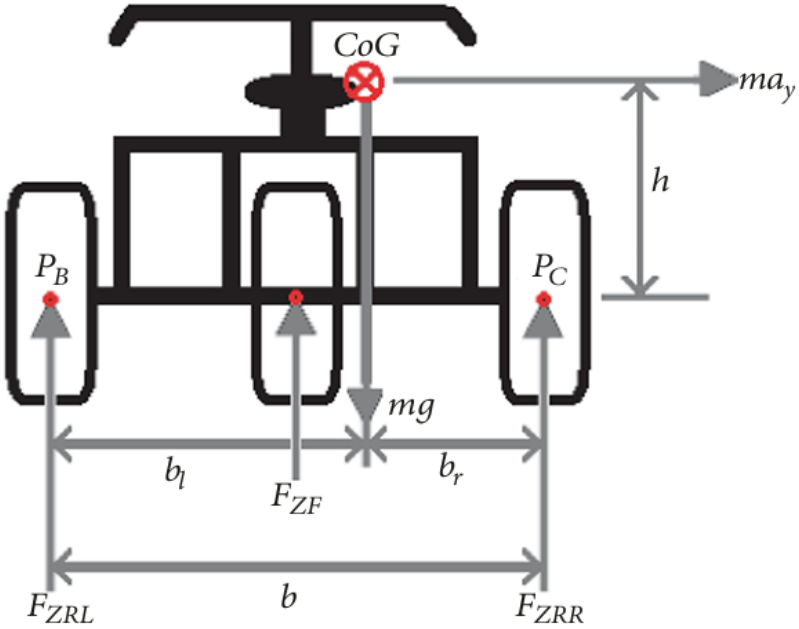

Figure 3.6: Free-body diagram of the rear axle in frontal view: CoG height h_{cg}, track width b, gravity mg, lateral inertial force ma_y, and rear wheel vertical reactions (source: (RODRÍGUEZ LICEA et al., 2018)).

The static balance of the forces in the vertical direction gives

$$F_{ZRL} + F_{ZRR} = m\,g, \tag{3.12}$$

and the moment balance about the outer contact point (e.g. the right contact P_C in Figure 3.6) yields

$$F_{ZRL} \cdot b + m\,a_y \cdot h_{\mathrm{cg}} = m\,g \cdot \frac{b}{2}. \tag{3.13}$$

Solving Equation (3.13) for the inner-wheel vertical load and combining with Equation (3.12) gives the two normal forces as functions of the lateral acceleration:

$$F_{ZRL} = m\left(\frac{g}{2} - \frac{a_y\,h_{\mathrm{cg}}}{b}\right), \tag{3.14}$$

$$F_{ZRR} = m\left(\frac{g}{2} + \frac{a_y\,h_{\mathrm{cg}}}{b}\right). \tag{3.15}$$

The redistribution of the vertical load between the inner and the outer wheel as a_y increases

is the well-known lateral load transfer effect. The dimensionless quantity that measures this redistribution is the *load transfer ratio* (LTR), defined as in ("Stability of Three-Wheeled Vehicles with and without Control System" 2013):

$$\mathrm{LTR} \;=\; \frac{F_{ZRR} \;-\; F_{ZRL}}{F_{ZRR} \;+\; F_{ZRL}} \;=\; \frac{2\,a_y\,h_{\mathrm{cg}}}{g\,b}. \tag{3.16}$$

An LTR of zero corresponds to equal loading of the two wheels in a straight line; an absolute LTR of unity indicates that the inner wheel has lifted off the ground, which is the onset of rollover.

The critical lateral acceleration at which the LTR reaches unity is obtained by setting $F_{ZRL} = 0$ in Equation (3.14) and solving for a_y:

$$a_{y,\mathrm{rollover}} \;=\; \frac{g\,b}{2\,h_{\mathrm{cg}}}. \tag{3.17}$$

This is the rollover threshold of the rickshaw: any combination of speed and curvature producing a centripetal acceleration greater than $a_{y,\mathrm{rollover}}$ would lift the inner rear wheel off the ground, with the consequent loss of lateral stability. The threshold is a geometric quantity that depends only on the track width b and on the centre-of-gravity height h_{cg}, not on the mass of the vehicle or on the tyre friction coefficient, making Equation (3.17) a robust and payload-independent basis for a safety bound.

For the rickshaw, with $b = 1.10$ m and $h_{\mathrm{cg}} \approx 0.75$ m as given in Table 3.1, Equation (3.17) yields

$$a_{y,\mathrm{rollover}} = \frac{9.81 \times 1.10}{2 \times 0.75} \approx 7.19\ \mathrm{m/s^2},$$

which is approximately 8.5 times the maximum centripetal acceleration $v_{\max}^2\,\kappa_{\max} \approx 0.85$ m/s^2 produced at the current operating limit. The safety supervision layer (Section 4.5) nevertheless enforces a conservative fraction $\mu < 1$ of this threshold as a soft bound on the commanded lateral acceleration, in order to retain margin against payload variations and surface disturbances that may shift the effective h_{cg} above its nominal value.

3.4.4. Model Adopted in This Thesis

The three formulations developed in the preceding subsections are not used in parallel: each plays a specific role in the architecture that follows. The kinematic single-track model of Section 3.4.1 is adopted as the prediction model of the lateral controller and provides the feedforward steering command of the high-level execution layer; the associated equations (3.1)–(3.3) and the curvature relation (3.5) reappear in Chapter 4 as the kinematic basis of the MPC error-state formulation. The free-body analysis of Section 3.4.3 provides the rollover threshold (3.17) that anchors the soft bound on the lateral acceleration in the safety supervisor. The dynamic single-track model of Section 3.4.2 is retained as a reference for the stability discussion but is not embedded in any control loop, for the reasons stated at the end of that subsection.

c

headercolor **Model**	**Role in thesis**	**Ac**
Kinematic single-track	Prediction model of the lateral MPC controller; feedforward steering command.	Mi ra ide the lin sm ap co eff
Dynamic single-track	Stability analysis framework; not embedded in any control loop.	Ca tyr dy ste ste
Free-body rollover analysis	Derives the lateral acceleration threshold used by the safety supervision layer.	Cl su of an rob bo

Table 3.2: Summary of the three vehicle model formulations adopted in this thesis, their roles, advantages, and accepted limitations.

4. Control Architecture Design and Implementation

The control architecture proposed in this thesis constitutes the principal technical contribution of the work. Its design draws on the landscape of methods surveyed in Chapter 2 and is constrained, at every level, by the physical characteristics of the rickshaw described in Chapter 3. The architecture is hierarchical: a high-level layer converts the planned trajectory into desired actuation set points, a low-level layer drives the physical actuators toward those set points using feedback, and a safety supervision component enforces the physical bounds that define the vehicle's operating envelope. The three layers communicate through the standard Autoware interfaces so that improvements to perception, localisation, and planning can be adopted without modifying the controllers.

4.1. Overall Architecture Overview

4.1.1. Hierarchical Structure

The control architecture is organised in three layers that process information at progressively shorter time scales and at progressively lower levels of abstraction. Figure 4.1 illustrates the resulting pipeline.

At the top of the hierarchy, the Autoware motion planner publishes a reference trajectory on the topic `/planning/scenario_planning/trajectory`. The trajectory is a time-stamped sequence of way-points, each carrying a two-dimensional position, a heading angle, a longitudinal speed, and a curvature value. This representation decouples the geometric and the temporal aspects of the desired motion and provides the high-level execution layer with all the information it needs to compute instantaneous set points.

At the intermediate level, two controllers operate in parallel on the two principal degrees of freedom of the vehicle. The lateral controller computes the desired steering angle $\delta_{\text{des}}(t)$ that minimises the deviation of the vehicle from the planned path. The longitudinal controller computes the desired acceleration $a_{\text{des}}(t)$ that drives the vehicle speed toward the planned speed profile. The outputs of the two controllers are assembled into a standard Autoware `AckermannControlCommand` and forwarded to the safety supervisor.

At the lowest level, the safety supervisor verifies the admissibility of the commanded values and, if necessary, modifies them before they are forwarded to the control bridge. The

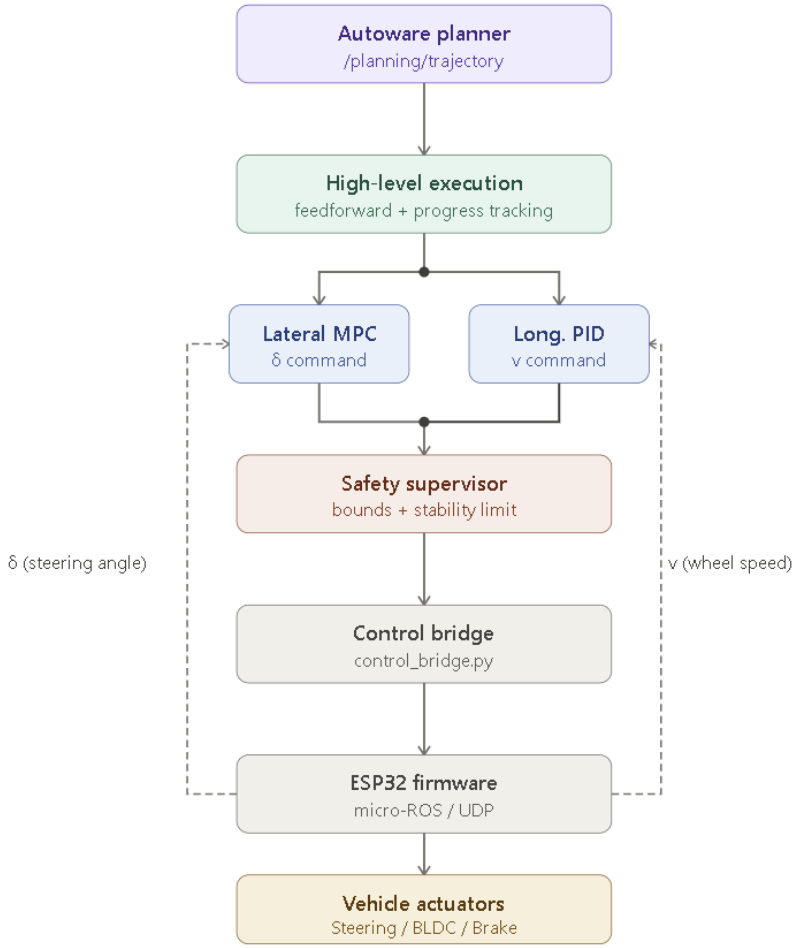

Figure 4.1: Three-layer hierarchical control architecture of the semi-autonomous rickshaw.

bridge translates the Autoware command into the compact message format understood by the embedded firmware on the ESP32 micro-controller, which drives the steering actuator and the electric motor.

4.1.2. Interface with the Autoware Stack

The Autoware control module is structured around a clearly defined set of input and output interfaces that are shared across all Autoware-compatible vehicles. Aligning the rickshaw's control architecture with these interfaces has a direct consequence for the design: the controllers do not need to implement trajectory parsing, localisation fusion, or motion planning logic of any kind; they receive a fully processed, time-stamped trajectory and are expected to produce a single actuation command. This clean boundary allows every upstream improvement in the perception and planning modules to propagate to the vehicle without any modification of the controllers.

Topic	Message type	Rate	Consumer / role
gray!20	*Subscribed — inputs to the control module*		
`/planning/scenario_planning/trajectory`	`Trajectory`	≈11 Hz	High-level execution layer
`/localization/kinematic_state`	`Odometry`	≈50 Hz	Lateral MPC, long. PID
`/vehicle/status/velocity_status`	`VelocityReport`	10 Hz	Longitudinal PID, EKF
`/vehicle/status/steering_status`	`SteeringReport`	46 Hz	Lateral MPC
gray!20	*Published — outputs of the control module*		
`/control/command/control_cmd`	`AckermannControlCommand`	≈43 Hz	Control bridge (input)
`/robot/data_in`	`Float32MultiArray`	≈43 Hz	ESP32 firmware

Table 4.1: ROS 2 topics at the boundary of the Autoware control module on the rickshaw platform. Rates marked with ≈ were measured during integration testing (Section 3.3.2).

Inputs to the control module. The control module consumes three categories of topics. The first is the reference trajectory, published by the scenario planning module on `/planning/scenario_planning/trajectory` as an `autoware_auto_planning_msgs/msg/Trajectory` message. Each element of the trajectory array is a `TrajectoryPoint` carrying a two-dimensional position, a heading angle, a reference longitudinal velocity, a reference longitudinal acceleration, a front-wheel steering angle, and a curvature value. This representation is richer than a geometric path: it specifies not only where the vehicle should be but also how fast it should travel and what steering angle is geometrically consistent with the path at each point. The controller therefore has, at every time step, all the information needed to compute both the feedforward and the feedback components of the actuation without additional planning. The second category consists of the vehicle state estimates, published by the localisation module on `/localization/kinematic_state` as a `nav_msgs/msg/Odometry` message containing the current pose (position and heading), the linear velocity, and the angular rate of the vehicle. This topic is the primary source of state information for the trajectory-following logic.

The third category comprises the proprioceptive measurements provided by the sensor bridges described in Section 4.6: the `autoware_auto_vehicle_msgs/msg/VelocityReport` published on `/vehicle/status/velocity_status` at 10 Hz and the `autoware_auto_vehicle_msgs/msg/SteeringReport` published on `/vehicle/status/steering_status` at 46 Hz. The longitudinal controller uses the velocity report as its feedback signal; the lateral MPC uses the steering report both as feedback for its internal state estimate and as an input to its delay compensation mechanism.

Output of the control module. The control module publishes a single output message on `/control/command/control_cmd`, of type `autoware_auto_control_msgs/msg/AckermannControlCommand`. The message carries two sub-structures. The `longitudinal` sub-structure contains the desired forward speed (m/s) and the desired longitudinal acceleration (m/s^2), produced by the PID controller after safety supervision. The `lateral` sub-structure contains the desired steering tire angle (rad) and the desired steering tire rotation rate (rad/s), produced by the MPC after safety supervision. This single compound message is the complete output of the architecture toward the vehicle, and it is the only topic that the control bridge needs to subscribe to.

Internal control nodes. Within the Autoware control module, two standard nodes implement the lateral and longitudinal controllers. The `mpc_lateral_controller` node implements the model predictive controller described in Section 4.3.2, whose behaviour is governed by the four tunable weights discussed in Section 4.4. The `pid_longitudinal_controller` node implements the PID controller described in Section 4.3.1, with the anti-windup, filtering, and delay compensation mechanisms described in the same section. Both nodes are instantiated from the standard Autoware source without modification to their code; the contribution of this thesis at this level consists entirely in the selection and systematic tuning of their parameters, which is presented in Chapter 5.

Vehicle interface and the bridge pattern. In a standard Autoware deployment, the `AckermannControlCommand` output is consumed by a dedicated vehicle interface node that translates the command into the protocol of the target vehicle — typically a CAN bus frame for a passenger car or a proprietary serial protocol for a research platform. The rickshaw does not have a CAN-compliant actuator bus; instead, the embedded firmware on the ESP32 is reachable through the micro-ROS UDP transport described in Section 3.3.3. The control bridge `control_bridge.py` plays the role of the vehicle interface node: it subscribes to `/control/command/control_cmd`, extracts the steering angle and the desired velocity after applying the safety clamps, assembles them into a compact `Float32MultiArray` message, and publishes it on `/robot/data_in`. This design isolates the Autoware control module from all hardware-specific considerations, so that any future change in the embedded protocol requires only a modification of the bridge node, not of the controllers.

4.1.3. Rationale for the Decoupled Lateral–Longitudinal Structure

The lateral and the longitudinal channels are treated as independent control problems throughout the architecture. This decoupling assumption is standard practice in vehicle control and is justified by the operating regime of the rickshaw: at low speeds and moderate curvatures, the cross-coupling between the two channels — manifesting through longitudinal–lateral tyre force interaction and through the load transfer associated with lateral acceleration — remains small enough to be handled by the safety supervision layer rather than by a unified controller. The review of the trajectory tracking literature in Section 2.3 confirms that this assumption is adopted by all practically relevant published architectures

for vehicles in this speed regime, from the two-layer structure of Martins et al. (MARTINS et al., 2008) to the spacing-plus-PID structure of Luu and Lupu (LUU and LUPU, 2019). A fully coupled controller, such as the nonlinear MPC of Dempster et al. (DEMPSTER et al., 2023), would be unnecessary and computationally prohibitive on the on-board hardware.

4.2. High-Level Trajectory Execution

4.2.1. Trajectory Representation and Progress Tracking

The input to the high-level execution layer is a discrete trajectory $\mathcal{T} = \{\mathbf{p}_i, \psi_i, v_i, \kappa_i\}_{i=1}^{N}$, where $\mathbf{p}_i = (x_i, y_i)$ is the planned position, ψ_i is the planned heading, v_i is the planned longitudinal speed, and κ_i is the planned path curvature at the i-th way-point. The temporal spacing between consecutive way-points is determined by the planner and is not necessarily uniform; the controller must therefore maintain an internal estimate of the current progress index — the index i^* of the way-point that is closest to the current vehicle pose — and update it at each control step.

Progress tracking is performed by projecting the current vehicle position $\mathbf{q} = (x, y)$ onto the trajectory and finding the closest way-point in the arc-length sense. The closest point index is updated monotonically: the search is restricted to a forward window starting from the previous i^*, which prevents the spurious backward jumps that can occur when the reference path has a non-convex geometry.

4.2.2. Feedforward Commands

Once the progress index i^* is known, the high-level layer extracts the reference values ψ_{i^*}, v_{i^*}, and κ_{i^*} and uses them to generate feedforward terms that are passed to the low-level controllers. The feedforward steering angle is computed from the kinematic steering formula

$$\delta_{\text{ff}}(t) \;=\; \arctan\Big(\kappa_{i^*} \cdot L\Big), \tag{4.1}$$

where L is the wheelbase of the vehicle. This term accounts for the steady-state steering required to maintain a circular arc of curvature κ_{i^*} at the current speed, and provides the MPC with a well-conditioned starting point from which to compute its corrective action. The feedforward speed command is simply v_{i^*}, which is forwarded directly to the longitudinal PID as the desired speed.

4.2.3. Update Rate Mismatch and Its Management

A practical complication arises from the mismatch between the rate at which the planner updates the trajectory and the rate at which the controller publishes actuation commands. In the integration tests described in Section 3.3.2, the planner operates at approximately 11 Hz while the controller operates at approximately 43 Hz; thus, for every one planning

update there are roughly four control steps. Between two successive planning updates, the controller continues to track the most recent available trajectory — the reference is held constant — and on arrival of a new plan the controller switches immediately to the new reference.

This behaviour is entirely appropriate for the scenario of interest, since the low-speed operation of the rickshaw implies that the vehicle does not travel far between two planner updates (approximately 0.07 m at the maximum considered speed). The risk of a discontinuous jump in the actuation command at the plan-switch instants is mitigated by the smoothness constraints embedded in the lateral MPC and by the rate-limiting mechanism of the safety supervisor.

4.3. Low-Level Control

4.3.1. Longitudinal Control: PID for Speed Tracking

Two-level longitudinal structure. The longitudinal channel is organised as a two-level architecture, consistent with the structure identified in the literature for adaptive cruise control and electric-vehicle speed control (LUU and LUPU, 2019; MARTINS et al., 2008). The outer level is a PID controller acting on the longitudinal speed error and computing a desired acceleration command. The inner level, executed in the embedded firmware on the ESP32, translates the desired acceleration into a motor torque command for the BLDC motor and, when necessary, into a braking signal for the friction brake. The separation between the two levels allows the outer controller to reason in terms of the kinematic quantity (acceleration) that is directly relevant for trajectory tracking, without needing to model the electrical and mechanical dynamics of the motor, which are handled locally by the firmware.

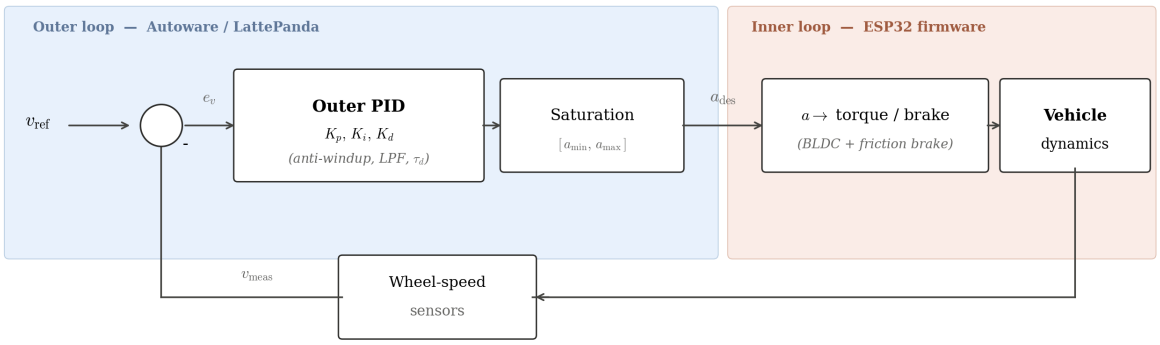

Figure 4.2: Two-level longitudinal control architecture. The outer PID loop running on Autoware computes the desired acceleration a_{des} from the speed error e_v; the inner ESP32 firmware loop translates it into a BLDC torque command or a brake command, with v_{meas} fed back from the wheel-speed sensors.

Mathematical formulation. Let $v_{\text{ref}}(t)$ denote the desired longitudinal speed provided by the high-level execution layer and $v(t)$ the actual longitudinal speed estimated from the wheel-speed sensors. The speed tracking error is

$$e_v(t) \;=\; v_{\text{ref}}(t) - v(t). \tag{4.2}$$

The continuous-time PID control law is

$$u_v(t) \;=\; K_p\, e_v(t) \;+\; K_i \int_0^t e_v(\tau)\, d\tau \;+\; K_d\, \frac{de_v(t)}{dt}, \tag{4.3}$$

where $u_v(t)$ represents the desired longitudinal acceleration command. In the discrete-time implementation at sampling period T_s, Equation (4.3) becomes

$$u_v(k) \;=\; K_p\, e_v(k) \;+\; K_i\, T_s \sum_{j=0}^{k} e_v(j) \;+\; \frac{K_d}{T_s}\Big[e_v(k) - e_v(k-1)\Big]. \tag{4.4}$$

The three terms have complementary roles. The proportional term $K_p\, e_v(k)$ provides an immediate correction proportional to the instantaneous error; a larger K_p accelerates the response but increases the risk of overshoot. The integral term accumulates the error over time and eliminates any steady-state offset that the proportional action alone cannot remove; a larger K_i improves the final accuracy but introduces the risk of oscillation and windup. The derivative term anticipates the tendency of the error and damps the response; a larger K_d reduces the overshoot but amplifies measurement noise.

Anti-windup mechanism. Integral windup is a well-known limitation of the standard PID controller: when the actuator saturates, the integral term continues to accumulate, producing an overshoot when the saturation is eventually removed. The controller implemented for the rickshaw employs a conditional integration scheme in which the accumulation of the integral term is suspended whenever the commanded acceleration exceeds the actuator bounds $[a_{\min}, a_{\max}]$:

$$\Delta I(k) \;=\; \begin{cases} e_v(k) & \text{if } a_{\min} \le u_v(k-1) \le a_{\max}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

This strategy, known as clamping anti-windup, is simple and effective for the speed range considered, where the saturation duration is short and the speed profile does not require sustained operation against the actuator limits.

Low-pass filtering of the speed error. The derivative term in Equation (4.4) amplifies high-frequency components of the speed error, which may be dominated by quantisation noise from the pulse-counting sensor or by vibrations transmitted from the road surface. A first-order low-pass filter is therefore applied to the error signal before the derivative is computed:

$$e_v^{\text{filt}}(k) \;=\; \alpha_{\text{lpf}}\, e_v(k) \;+\; \Big(1 - \alpha_{\text{lpf}}\Big)\, e_v^{\text{filt}}(k-1), \tag{4.6}$$

where $\alpha_{\text{lpf}} \in (0, 1]$ is the filter gain. A value of α_{lpf} close to unity reduces filtering and preserves the responsiveness of the derivative term; a value closer to zero produces stronger attenuation of high-frequency noise at the price of a greater phase lag. The choice of α_{lpf} is therefore a direct expression of the design priority and is treated, together with the PID gains, as a tuning parameter in the profiles discussed in Section 4.4.

Delay compensation. A systematic source of error in the longitudinal channel is the actuation delay between the moment at which the speed command is computed and the moment at which the motor torque actually changes. This delay, denoted τ_d, arises from the combination of the computational latency of the bridge, the communication latency of the micro-ROS transport, and the motor current rise time. Its presence introduces a phase lag in the closed-loop response that reduces the effective damping and may cause oscillations if the proportional gain is tuned too aggressively.

The compensation strategy adopted in the architecture is to predict the speed error that will be present at time $t + \tau_d$, and use the predicted error as the input to the controller rather than the instantaneous error. Under the assumption that the commanded acceleration is approximately constant over the delay interval, the predicted speed is

$$\hat{v}(k + n_d) \;=\; v(k) \;+\; u_v(k-1) \cdot n_d \cdot T_s, \tag{4.7}$$

where $n_d = \lfloor \tau_d / T_s \rfloor$ is the delay expressed in samples. The predicted error $e_v^{\text{pred}}(k) = v_{\text{ref}}(k) - \hat{v}(k + n_d)$ replaces $e_v(k)$ in Equation (4.4). This simple predictor is a reduced form of the Smith predictor principle, applied under the assumption of a purely kinematic model for the speed dynamics (LUU and LUPU, 2019). Its effectiveness depends on the accuracy of the delay estimate τ_d, which is itself a tuning parameter.

Actuation interface. The output of the PID controller is a desired acceleration $u_v(k)$. Positive values are interpreted as drive commands: the firmware converts the desired acceleration into a torque set point for the BLDC motor, using a static map from acceleration to torque that accounts for the effective inertia of the vehicle. Negative values are interpreted as deceleration commands and are forwarded to the friction brake, which provides the primary deceleration capability of the vehicle.

4.3.2. Lateral Control: Model Predictive Steering

Motivation for MPC on the lateral channel. The choice of a model predictive controller for the lateral channel, rather than a simpler PID or a reactive geometric controller, is motivated by three structural properties of the lateral control problem that are difficult to address within a purely reactive framework.

First, the steering of the rickshaw is subject to hard constraints on the angle $|\delta| \leq \delta_{\max}$ and on the steering rate $|\dot{\delta}| \leq \dot{\delta}_{\max}$, which reflect the mechanical limits of the steering mechanism and the comfort and stability requirements discussed in Section 2.6. These constraints are most naturally expressed as explicit inequalities in an optimisation problem, which is the defining characteristic of MPC. The survey of Rahman et al. (RAHMAN et al.,

2025) confirms that MPC is especially suitable for constrained trajectory tracking precisely for this reason.

Second, the steer-by-wire system of the rickshaw introduces a non-negligible actuation delay between the steering command and the physical response of the front wheel. In a purely reactive controller, this delay effectively adds phase lag to the open-loop system, reducing the phase margin and potentially destabilising the closed loop at the gains required for adequate tracking performance. The MPC formulation accommodates this delay explicitly by incorporating a model of the actuator delay within the predictor, as described below.

Third, the lateral control problem has an inherently predictive character: the steering correction for a curve that begins at the current position should ideally be initiated slightly before the vehicle reaches the curve, so that the actual trajectory follows the planned one smoothly rather than catching up reactively. The finite prediction horizon of the MPC captures this lookahead property naturally, without requiring any explicit rule for pre-cornering.

Kinematic error-state model. The prediction model underlying the MPC is derived from the kinematic single-track representation of the rickshaw introduced in Section 3.4.1, re-expressed in error coordinates relative to the reference path. Let the signed lateral deviation of the vehicle from the closest point of the reference path be e_y (positive to the left), and let the heading error be $e_\psi = \psi - \psi_{\mathrm{ref}}$, where ψ is the body heading defined in Equation (3.3) of Section 3.4.1 and ψ_{ref} is the tangent to the reference path at the closest point. Under the small-angle approximation, which is justified by the low-speed, moderate-curvature operating conditions of the vehicle, the continuous-time equations of motion in error coordinates follow directly from (3.1)–(3.3) and read:

$$\dot{e}_y \;=\; v \sin(e_\psi) \;\approx\; v\, e_\psi, \tag{4.8}$$

$$\dot{e}_\psi \;=\; \frac{v}{L}\tan(\delta) - v\,\kappa_{\mathrm{ref}} \;\approx\; \frac{v}{L}\,\delta - v\,\kappa_{\mathrm{ref}}, \tag{4.9}$$

$$\dot{\delta} \;=\; u, \tag{4.10}$$

where v is the current longitudinal speed, L is the wheelbase, κ_{ref} is the curvature of the reference path at the closest point, and $u = \dot{\delta}$ is the steering rate, which constitutes the control input to the MPC. The three state variables are assembled into the state vector $\mathbf{x} = [e_y,\, e_\psi,\, \delta]^\top$ and the model is discretised at the MPC sampling period T_c using a first-order Euler scheme to obtain

$$\mathbf{x}(k+1) \;=\; \mathbf{A}(v, L)\,\mathbf{x}(k) \;+\; \mathbf{B}(T_c)\,u(k) \;+\; \mathbf{d}(v, \kappa_{\mathrm{ref}}), \tag{4.11}$$

where the matrices $\mathbf{A}$, $\mathbf{B}$, and the feedforward disturbance vector $\mathbf{d}$ are

$$\mathbf{A}(v, L) \;=\; \begin{pmatrix} 1 & vT_c & 0 \\ 0 & 1 & \frac{vT_c}{L} \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{B}(T_c) \;=\; \begin{pmatrix} 0 \\ 0 \\ T_c \end{pmatrix}, \quad \mathbf{d} \;=\; \begin{pmatrix} 0 \\ -v\,\kappa_{\mathrm{ref}}\,T_c \\ 0 \end{pmatrix}. \tag{4.12}$$

The velocity v and the reference curvature κ_{ref} appear as time-varying parameters in the model; they are updated at each MPC step from the current speed measurement and from

the reference trajectory.

Quadratic cost function. The MPC computes, at each time step k, the optimal sequence of steering rates $\{u(k), u(k+1), \ldots, u(k+N-1)\}$ over a prediction horizon of N steps that minimises the cost function

$$J \;=\; \sum_{j=0}^{N-1} \Big[w_{\text{lat}}\, e_y(k+j)^2 \;+\; w_\psi\, e_\psi(k+j)^2 \;+\; w_\delta\, \delta(k+j)^2 \;+\; w_{\text{jerk}} \Big(\delta(k+j+1) - \delta(k+j) \Big)^2 \Big], \tag{4.13}$$

subject to the prediction model (4.11) and to the constraints

$$|\delta(k+j)| \;\leq\; \delta_{\max}, \quad j = 0, \ldots, N, \tag{4.14}$$

$$|\delta(k+j+1) - \delta(k+j)| \;\leq\; \dot{\delta}_{\max}\, T_c, \quad j = 0, \ldots, N-1. \tag{4.15}$$

The four scalar weights in Equation (4.13) correspond directly to the four tunable parameters of the Autoware lateral MPC implementation:

- w_{lat} (`mpc_weight_lat_error`) controls the priority assigned to the reduction of the lateral deviation. A larger value produces tighter tracking at the price of a more aggressive steering response.
- w_δ (`mpc_weight_steering_input`) penalises the magnitude of the steering angle. A larger value discourages large steering deflections and produces a calmer, more central-biased steering behaviour.
- w_{jerk} (`mpc_weight_lat_jerk`) penalises rapid changes in the steering angle, which correspond to lateral jerk. A larger value smooths the steering trajectory and improves passenger comfort, at the price of a slower correction of lateral deviations.
- The heading weight w_ψ is kept at a fixed ratio relative to w_{lat} in the Autoware implementation and is not independently exposed as a tuning parameter.

At each time step, the MPC problem reduces to a convex quadratic program (QP) because the prediction model is linear and the cost function is quadratic; the problem is solved by a fast active-set algorithm embedded in the Autoware stack, which provides a real-time-compatible solution on the LattePanda hardware.

Actuator delay compensation. As noted earlier, the steer-by-wire system introduces a latency τ_s between the moment the steering command is issued and the moment the front wheel reaches the commanded angle. If this delay is not accounted for within the MPC, the prediction model underestimates the actual lateral deviation that will result from the commanded steering, and the controller systematically under-corrects during dynamic manoeuvres.

The delay is compensated within the MPC by augmenting the initial state of each prediction with the effect of the commands that have already been issued but not yet applied. Specifically, if the delay is $n_s = \lfloor \tau_s / T_c \rfloor$ steps, the predictor uses as its initial state the state that results from propagating the current measured state through the prediction model for n_s steps under the commands already in the pipeline. The parameter `input_delay` in the Autoware configuration corresponds directly to τ_s, and its tuning is discussed in Section 4.4.

4.4. Controller Tuning and Design Profiles

4.4.1. Tuning Philosophy

The design of a controller for a real vehicle is never complete with the mathematical formulation: the values assigned to the free parameters ultimately determine whether the vehicle behaves as intended in practice. For the rickshaw, this tuning problem is complicated by the absence of an accurate dynamical model that could serve as a basis for analytical or automatic optimisation. The parameters of the physical system, the effective steering actuator bandwidth, the exact actuation delay, the surface-dependent friction of the tyres, the influence of the payload on the vehicle inertia, are either unavailable or insufficiently precise to support a model-based tuning procedure.

The approach adopted in this thesis is systematic but empirical, and is organised around the identification of four distinct design priorities that correspond to four fundamentally different operating contexts for the vehicle. Each priority is translated into a profile — a specific combination of parameter values — that reflects the corresponding priority through the direction and magnitude of its deviations from the baseline Autoware configuration. The four priorities are:

- **Baseline**: the default Autoware configuration, retained unmodified. It provides a reference point against which all other profiles are evaluated and represents the behaviour one would obtain by deploying the stack without any platform-specific tuning.
- **Comfort**: the controller is tuned to minimise the rate and the magnitude of the actuation signals. This priority is appropriate when the rickshaw is carrying passengers and the smoothness of the ride is of primary importance, even at the cost of a slightly larger tracking deviation.
- **Tracking**: the controller is tuned to minimise the lateral deviation and the speed-tracking error. This priority is appropriate for precision tasks — operating in narrow corridors, following a precisely marked route — where geometric accuracy is paramount and the faster, more aggressive actuation is acceptable.
- **Robust**: the controller is tuned to maintain adequate tracking performance under the non-ideal conditions that characterise real outdoor operation: measurement noise from the sensors, variable actuation delays due to communication jitter, and surface disturbances transmitted to the vehicle. This priority sacrifices some tracking tightness and some actuation smoothness to gain stability margin.

The conceptual trade-off space defined by these four priorities is illustrated in Figure 4.3. No single profile is Pareto-optimal across all metrics simultaneously; the choice of profile is therefore a design decision that depends on the intended use of the vehicle and on the specific operating conditions.

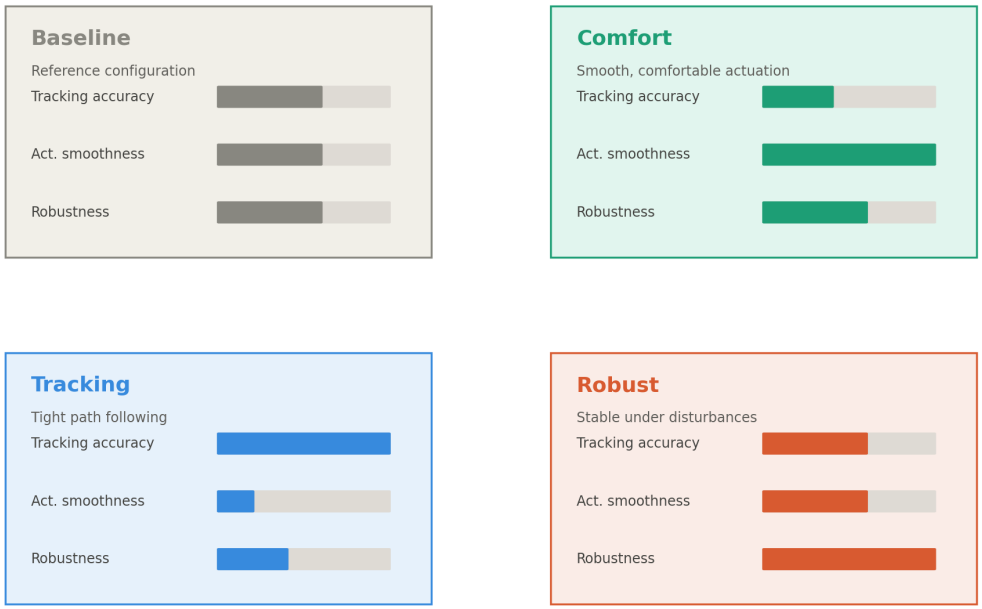

Figure 4.3: The four controller profiles and their relative design priorities across the three tuning axes. Bar length indicates the weight assigned to each axis; no single profile is optimal on all three simultaneously.

4.4.2. Longitudinal PID Tuning — Parameter Roles and Trade-offs

The longitudinal PID has five tunable parameters, each of which influences the controller behaviour in a distinct and identifiable way. Understanding the role of each parameter is a prerequisite for principled tuning, and the conceptual description given below directly motivates the structure of the four profiles whose numerical values are presented in Chapter 5.

Proportional gain K_p. The proportional gain sets the sensitivity of the output to the instantaneous error. It is the dominant determinant of the response speed: a larger K_p produces a faster correction but increases the tendency to overshoot the desired speed, particularly during acceleration phases where the error is large and the integral action has not yet had time to build up. In the comfort profile, K_p is reduced relative to the baseline to produce a gentler acceleration; in the tracking profile, it is increased to produce the fastest possible convergence to the reference speed.

Integral gain K_i. The integral gain governs the long-term elimination of the steady-state error. At low speeds and on irregular surfaces, a non-zero steady-state error is likely because friction and surface grade disturbances create persistent offsets that the proportional term alone cannot cancel. A larger K_i eliminates these offsets more aggressively but risks oscillation, particularly on uphill sections where the integral term may have accumulated a large value before the grade changes. The robust profile uses a smaller K_i than the baseline to reduce this risk; the comfort profile also reduces K_i to avoid the abrupt corrective pulses that a large integral term can produce.

Derivative gain K_d. The derivative gain adds damping to the response and reduces the overshoot associated with a large K_p. Its effect is most visible in the transient response following a step change in the reference speed: a larger K_d produces a more critically damped or overdamped response. The baseline profile uses $K_d = 0$ — the standard Autoware default — so that the derivative action is entirely absent; the comfort and tracking profiles introduce a non-zero K_d to reduce the overshoot while maintaining the respective responsiveness priorities.

Low-pass filter gain α_{lpf}. As discussed in Section 4.3.1, the filter gain controls the trade-off between derivative responsiveness and noise rejection. A larger α_{lpf} implies less filtering, preserving the high-frequency information in the error signal at the price of amplifying sensor noise. The tracking profile, which aims for the fastest and most accurate response, uses a large α_{lpf}; the robust and comfort profiles use a smaller value to attenuate the noise contribution and produce a smoother command signal.

Delay compensation time τ_d. The delay compensation time should, ideally, match the actual round-trip latency between the command computation and the motor response. An underestimation of this latency leaves a residual phase lag that effectively reduces the damping of the closed loop; an overestimation produces a phase advance that may cause the controller to command a corrective action that has already been overtaken by the actual motor response. The robust profile uses a larger τ_d than the other profiles — acknowledging that in noisy, real-world conditions the apparent latency is larger and more variable — while the tracking profile uses the smallest value, assuming benign conditions and a tight command loop.

4.4.3. Lateral MPC Tuning — Weight Roles and Trade-offs

The four weights of the lateral MPC cost function (4.13) define the relative importance assigned to the competing objectives of tracking accuracy, steering economy, and riding comfort. Their conceptual roles are described below, with the explicit acknowledgement that the physical effects of each weight are coupled: changing one weight alters the importance of the others through the structure of the quadratic programme.

Lateral error weight w_{lat}. This weight is the primary determinant of tracking tightness. A large w_{lat} forces the MPC to assign the highest priority to the reduction of the cross-track error e_y, which translates into a faster and more aggressive steering correction. In the limit of a very large w_{lat}, the controller approaches the behaviour of a minimum-time lateral tracker, applying the maximum admissible steering rate immediately whenever the vehicle deviates from the path. A small w_{lat} allows the MPC to tolerate a larger lateral deviation in order to keep the steering commands smoother, which is the defining characteristic of the comfort profile.

Steering input weight w_δ. This weight penalises the absolute magnitude of the steering angle. Together with the steering rate constraint (4.15), it limits the size of the corrections

that the MPC is willing to apply. A large w_δ biases the controller toward small steering angles, which produces a centred, gentle steering behaviour. This is appropriate for straight or mildly curved segments but may introduce a systematic lateral offset on tightly curved segments, where the kinematic feedforward term (4.1) is large and the MPC must accept a significant steering angle to maintain the path. The comfort profile uses a large w_δ; the tracking profile uses a small one.

Lateral jerk weight w_{jerk}. The lateral jerk weight penalises rapid changes in the steering angle, which correspond to high lateral jerk — abrupt lateral accelerations that are felt by passengers as sudden sideways impulses. From the perspective of the cost function (4.13), this weight penalises the difference $\delta(k+j+1) - \delta(k+j)$ at each step of the prediction horizon, effectively acting as a regularisation on the steering trajectory. A large w_{jerk} forces the MPC to distribute steering corrections over multiple steps rather than applying them all at once, producing a smoother but potentially slower response. The comfort profile uses the largest value of w_{jerk}; the tracking profile uses the smallest.

Input delay τ_s. Unlike the weights, the input delay parameter is not a design choice in the sense of expressing a priority; it is, ideally, an accurate estimate of the physical latency of the steering actuator. In practice, however, the true delay is uncertain and variable, so τ_s must be tuned empirically. The robust profile uses a larger τ_s than the other profiles, providing an additional phase margin for the case in which the actual delay exceeds its nominal value. The tracking profile uses the smallest τ_s, which is appropriate when the delay is well characterised and the operating conditions are benign.

4.4.4. Integral-Partition PID for the Inner Steering Angle Loop

The MPC described in Section 4.3.2 acts as the outer loop of the lateral channel, computing the desired steering angle $\delta_{\text{des}}(t)$. The translation of this set point into an actual rotation of the front wheel is the responsibility of an inner angle-tracking loop acting on the steering actuator. In the present implementation, this inner loop is a standard PID provided by the servo driver of the actuator; its tuning is not exposed to the high-level controller.

An improvement identified in the course of the literature review and proposed as a refinement of the inner loop is the integral-partition PID strategy of (*Research of Automotive Steer-by-Wire Control Based on Integral Partition PID Control* n.d.). The strategy addresses a specific limitation of the standard PID in the context of steering control: when the error is large, as it is during a fast correction from a large lateral deviation, the integral term accumulates rapidly and produces an overshoot when the error finally decreases. This overshoot is particularly undesirable in the steering loop, because it translates into a transient over-steer that may momentarily increase the lateral deviation.

The integral-partition PID eliminates this overshoot by switching the controller between two modes depending on the magnitude of the error. For large errors $|e(k)| > \varepsilon$, only the proportional and derivative terms are active:

$$u(k) \;=\; K_p\, e(k) \;+\; \frac{K_d}{T_s}\Big[e(k) - e(k-1)\Big], \qquad |e(k)| > \varepsilon, \tag{4.16}$$

which produces a fast correction without accumulating the integral. Once the error falls within the threshold, the full PID is active:

$$u(k) \;=\; K_p\, e(k) \;+\; K_i\, T_s \sum_{j=0}^{k} \Delta I(j) \;+\; \frac{K_d}{T_s}\Big[e(k) - e(k-1)\Big], \qquad |e(k)| \le \varepsilon, \tag{4.17}$$

which removes the steady-state offset with the integral action. The reported result on a return-to-centre test with a 10° displacement confirms that this switching strategy reduces the settling time from approximately 2.4 s (standard PID) to approximately 0.63 s, eliminating the overshoot entirely (*Research of Automotive Steer-by-Wire Control Based on Integral Partition PID Control* n.d.).

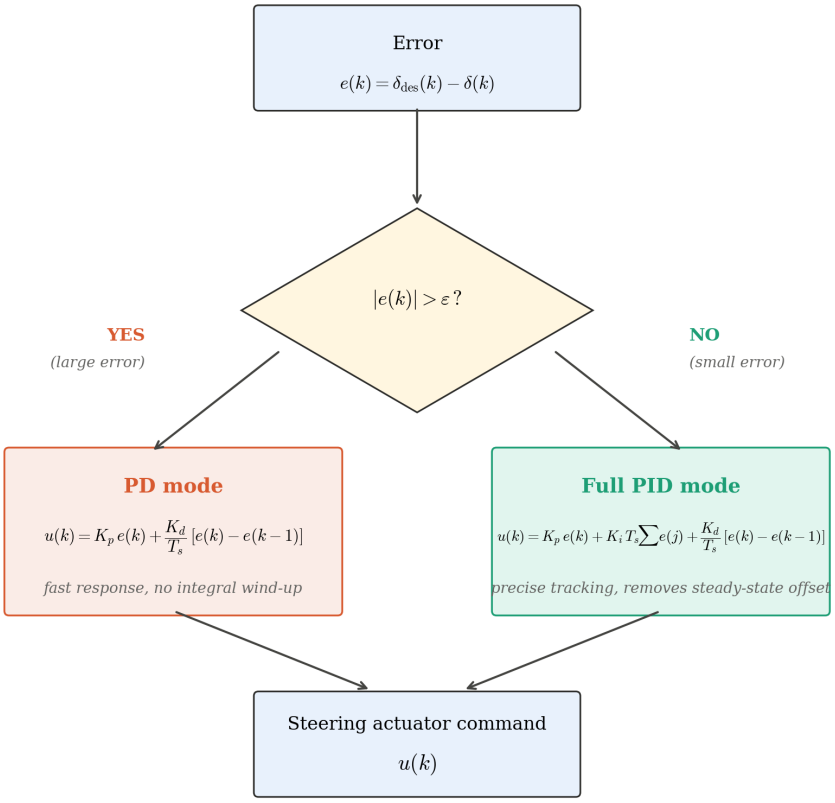

Figure 4.4: Switching logic of the integral-partition PID for the inner steering angle loop. PD mode is active for large errors ($|e(k)| > \varepsilon$) to avoid integral wind-up; the full PID is enabled once the error falls within the threshold to remove the residual steady-state offset.

The threshold ε is the critical design parameter of the strategy: it must be chosen large enough to prevent the integral from activating too early, but small enough to allow the integral to eliminate the residual error in a reasonable time. Its value depends on the gain values and on the actuator bandwidth and is determined empirically. The integration of this strategy into the embedded firmware of the rickshaw is an on-going implementation task whose completion will allow the angular tracking accuracy of the inner loop to be characterised independently of the outer MPC loop.

4.5. Safety Supervision and Constraint Handling

4.5.1. Role and Position of the Safety Layer

The safety supervision component intercepts the actuation commands produced by the lateral MPC and the longitudinal PID and verifies that they satisfy a set of hard and soft constraints before they are forwarded to the physical actuators. Its position in the architecture, between the controllers and the bridge, means that no unsafe command can reach the hardware regardless of the behaviour of the upstream layers. This design ensures that the safety requirements can be independently verified and, if necessary, tightened without modifying the controllers.

Table 4.2: Hard and soft bounds enforced by the safety supervision layer on the actuation commands.

Signal	**Min**	**Max**	**Notes**
Speed command v	0 m/s	1.66 m/s ($\approx$6 km/h)	Current operational safety limit.
Acceleration a	$a_{\min}$	$a_{\max}$	Used when acceleration control is enabled.
Steering angle δ	$-\delta_{\max}$	$+\delta_{\max}$	Bounded by vehicle steering limits.

The component implements four families of constraints, described in the following subsections, in increasing order of the severity of the intervention they trigger.

4.5.2. Hard Bound on the Steering Angle

The steering angle δ is constrained by the mechanical geometry of the front fork to the interval $[-\delta_{\max}, +\delta_{\max}]$. Any commanded value outside this interval would require the actuator to reach a position that is physically inaccessible, saturating the motor and potentially damaging the steering mechanism. The safety supervisor applies a symmetric saturation:

$$\delta_{\text{safe}} \;=\; \text{clamp}\Big(\delta_{\text{des}},\, -\delta_{\max},\, +\delta_{\max}\Big), \tag{4.18}$$

where δ_{des} is the steering angle commanded by the MPC. This bound is also embedded as a constraint inside the MPC optimisation, as shown in Equation (4.14); the saturation at the supervisor level provides a second line of defence against numerical anomalies or configuration errors.

4.5.3. Hard Bound on the Steering Rate

The steering actuator has a finite angular velocity, and commanding a steering change faster than the actuator can realise produces a tracking error in the inner angle loop that accumulates over time. The supervisor limits the rate of change of the steering command:

$$|\delta_{\text{safe}}(k) - \delta_{\text{safe}}(k-1)| \;\leq\; \dot{\delta}_{\max} \cdot T_s, \tag{4.19}$$

where $\dot{\delta}_{\max}$ is the maximum steering angular velocity determined from the actuator specifications. This bound is also present as a constraint in the MPC, and the supervisor provides the same secondary protection.

4.5.4. Hard Bound on the Longitudinal Speed

The longitudinal speed command is constrained to the interval $[0, v_{\max}]$, where the lower bound prevents the vehicle from being commanded to move backward through the planner interface, and the upper bound enforces the operational speed limit of the current validation campaign:

$$v_{\text{safe}} \;=\; \text{clamp}\Big(v_{\text{des}},\, 0,\, v_{\max}\Big). \tag{4.20}$$

4.5.5. Soft Bound on the Lateral Acceleration

The most nuanced safety constraint concerns the lateral acceleration. In the kinematic bicycle model, the centripetal acceleration of the vehicle is approximately

$$a_{\text{lat}} \;\approx\; \frac{v^2 \tan(\delta)}{L} \;\approx\; \frac{v^2\,\delta}{L}, \tag{4.21}$$

A lateral acceleration that is too large may cause the vehicle to approach its rollover limit, described in the literature through the rollover index (RODRÍGUEZ LICEA et al., 2018) or the load transfer ratio (LI et al., 2017; AKAR and DERE, 2014). The rollover threshold of the rickshaw was derived from the free-body analysis of the rear axle presented in Section 3.4.3, where the inner-wheel vertical load was shown to vanish at the critical lateral acceleration

$$a_{\text{lat,rollover}} \;=\; \frac{g\,b}{2\,h_{\text{cg}}}, \tag{4.22}$$

which depends only on the track width b and on the centre-of-gravity height h_{cg} (cf. Equation (3.17) of Section 3.4.3). In practice, the rickshaw is expected to operate well below this limit; the soft bound adopted in the supervisor is a fraction $\mu < 1$ of the theoretical threshold:

$$a_{\text{lat}} \;\leq\; \mu \cdot a_{\text{lat,rollover}}, \tag{4.23}$$

with μ chosen conservatively at 0.7 to provide a safety margin that accounts for payload variations and surface disturbances.

When the predicted lateral acceleration, computed from the current speed and the current steering command through Equation (4.21), would exceed the soft bound, the supervisor reduces the commanded speed rather than the commanded steering angle. This intervention strategy preserves the geometric tracking — the vehicle continues to follow the planned curvature — and trades the tracking of the speed profile for the maintenance of the stability margin. The corresponding reduced speed limit, derived by inverting Equation (4.21) with the lateral acceleration bound substituted, is

$$v_{\text{safe}} \;=\; \min\!\Big(v_{\text{des}},\; \sqrt{\frac{\mu \cdot a_{\text{lat,rollover}} \cdot L}{|\delta_{\text{safe}}|}}\Big). \tag{4.24}$$

The choice of prioritising geometric tracking over speed tracking in this intervention reflects the operational context: for an urban passenger and freight vehicle, leaving the planned path is more dangerous than arriving at a way-point slightly later.

4.5.6. Comparison with Full Rollover-Based Controllers

The safety supervisor described above is intentionally simpler than the MPC-based rollover prevention controllers proposed in the literature, such as the three-dimensional dynamics MPC of Li et al. (LI et al., 2017) or the switching rollover controller of Akar and Dere (AKAR and DERE, 2014). The rationale for this simplification is threefold. First, the operational speed of the vehicle is low enough that the rollover risk under normal operating conditions is small; the theoretical threshold computed from Equation (3.17) is not expected to be approached in the scenarios considered. Second, the implementation of a predictive rollover controller requires an accurate real-time estimate of the centre-of-gravity height and of the load on each wheel, neither of which is directly measured in the current sensor configuration. Third, the computational budget of the LattePanda is shared with the perception and planning modules of Autoware, which leaves limited headroom for a complex online optimisation.

The simple lateral acceleration bound provides a conservative safety envelope that is demonstrably appropriate for the current operating domain and that can be tightened or replaced by a more sophisticated mechanism in future revisions of the platform.

4.6. Sensor Integration and Hardware Implementation

The closed-loop operation of the control architecture described in the preceding sections depends critically on two proprioceptive measurements: the longitudinal speed of the vehicle and the steering angle of the front wheel. Neither was available on the rickshaw prototype at the outset of the project. Their provision required not only the selection of appropriate sensors and the design of the mechanical mounting, but also a sustained effort of embedded firmware development and software filtering, driven by a series of hardware anomalies that were progressively identified and resolved during the integration campaign.

This section documents that full process from sensor selection, through physical installation and firmware implementation, to the four-stage signal conditioning that ultimately delivered clean, stable measurements to Autoware in sufficient detail to allow its complete replication and extension.

4.6.1. Wheel Speed Sensor — Selection and Integration

Sensor selection. The wheel-speed sensor must satisfy four requirements specific to the rickshaw's operating environment. It must produce a well-defined output at zero rotational speed, since the vehicle starts and stops from rest and the Autoware EKF requires a continuous velocity signal at all times; passive sensors, whose output voltage is proportional to the rate of magnetic flux change, produce no signal at standstill and are therefore inadmissible. The output must be digital, to avoid susceptibility to electromagnetic noise radiated by the BLDC motor in close proximity. The sensor must tolerate outdoor dust, moisture, and mechanical vibration without loss of signal integrity. Finally, it must interface directly with the ESP32 micro-controller without additional analogue signal conditioning hardware.

Three candidates were evaluated: a generic Hall-effect sensor of type 55100, a sensor board based on the Infineon TLE4922 with an integrated logic stage, and a GMR rotary encoder equivalent to the one used for the steering channel. The 55100 requires an external pull-up resistor on its open-drain output and lacks a compact board form factor suited to the confined wheel housing. The GMR rotary encoder provides unnecessary resolution for a binary pulse-per-tooth measurement and carries a higher unit cost. The Infineon TLE4922 was selected: it is an active mono-cell Hall-effect sensor with an on-chip comparator and an open-drain digital output, requires only two standard passive components, and satisfies all four requirements directly.

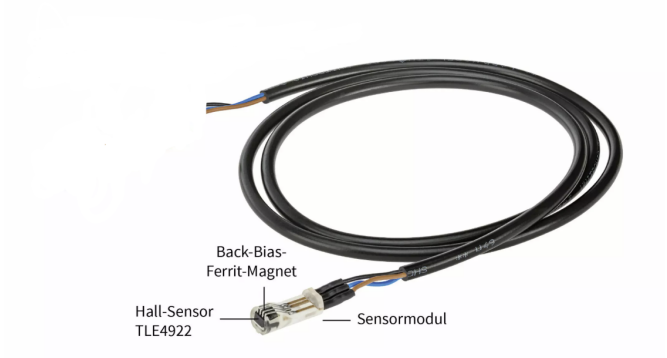

Figure 4.5: TLE4922 speed sensor (source: (INFINEON TECHNOLOGIES AG, 2025)).

Operating principle. The TLE4922 is a mono-cell active Hall sensor with integrated digital signal processing. A single chopped Hall probe measures the magnetic flux density at the sensitive area; the chopping technique periodically reverses the bias current direction through the Hall element, suppressing the offset voltage that would otherwise dominate the measurement at the small field amplitudes encountered near the encoder ring. The digitised output of the Hall probe is processed by an internal tracking ADC and fed to a digital core that implements an adaptive symmetrical hysteresis algorithm to set the switching

threshold dynamically. When the field at the sensing area exceeds this threshold — corresponding to a tooth passing in front of the sensor — the output transistor pulls the Q line to a low voltage; when the field falls below the threshold as a notch passes, the transistor is released and the Q line returns to its high state through the external pull-up.

Two properties of this operating principle are particularly relevant for the application. First, the internal adaptive hysteresis algorithm dynamically tracks the peak-to-peak magnetic field swing, making the switching threshold independent of the absolute field magnitude, which renders the output insensitive to slow variations in magnet strength due to temperature or ageing and allows a wider and more mechanically tolerant air gap than fixed-threshold Hall devices. Second, the sensor is active, its internal circuitry is powered independently, and produces a well-defined digital output at any rotational speed including zero, without the frequency-dependent sensitivity that limits passive sensors at low speed.

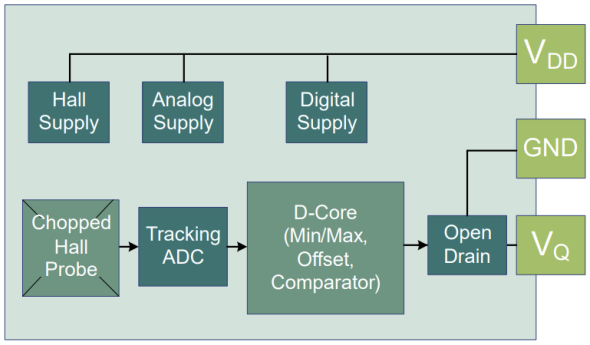

Figure 4.6: Internal architecture of the TLE4922: chopped Hall probe, tracking ADC, adaptive hysteresis core, and open-drain output (source: (INFINEON TECHNOLOGIES AG, 2025)).

Digital output interface and electrical connection. The TLE4922 communicates through a single-wire, single-ended open-drain digital output on pin Q — there is no serial protocol, no clock line, and no framing of any kind. The sensor's internal open-drain transistor pulls the Q line to a saturation voltage of at most $V_{Q,\mathrm{SAT}} = 0.4$ V when the magnetic field at the sensing area exceeds the internal threshold (tooth in front of the sensor), and releases the line to a high-impedance state when the field falls below the threshold (notch in front of the sensor). An external pull-up resistor $R_L = 1.2$ kΩ connected between the Q pin and the load supply restores the line to a logic-high level in the released state.

The electrical timing parameters relevant to the firmware implementation are the output fall time $t_f = 2.2$–3.8 μs, the output rise time $t_r = 1$–20 μs (both measured between the 20 % and 80 % levels of the load voltage), and the propagation delay of the falling edge $t_d = 12.5$–23.5 μs. The delay characterises the lag between the physical centre of a tooth crossing the sensitive area and the corresponding falling edge appearing on the Q line; at the low speeds of the rickshaw its contribution to the speed estimate is negligible relative to the measurement window of 100 ms. Full accuracy of the adaptive hysteresis algorithm is reached after the fourth falling edge following power-on, at which point the internal self-calibration module has completed one averaging cycle over the previous tooth geometry.

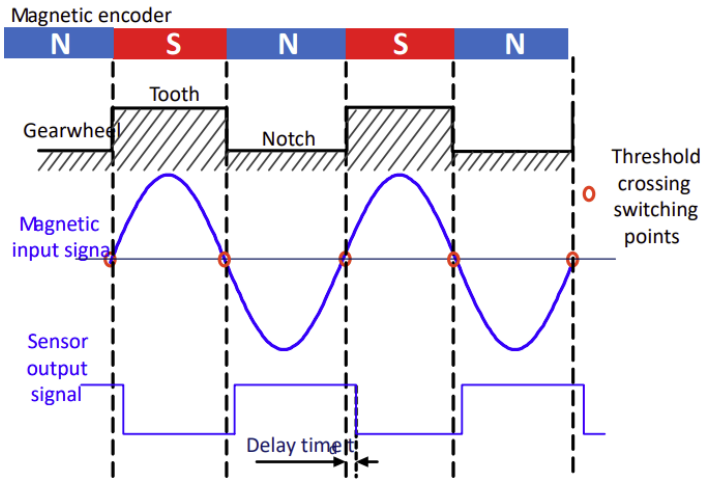

Figure 4.7: TLE4922 output signal timing relative to a tooth passage. The propagation delay $t_d = 12.5$–$23.5\ \mu$s is negligible at the 100 ms firmware measurement window (source: (INFINEON TECHNOLOGIES AG, 2025)).

Hardware configuration and encoder ring. Two TLE4922 sensors are installed, one on each rear wheel, both configured as digital inputs with interrupt capability. Both sensors are powered from the 5 V rail of the ESP32 development board.

The encoder ring mounted on each wheel hub carries 22 ferromagnetic magnets arranged on a 24-position template, with two positions left empty at angular locations 12 and 24 of the 24-slot grid to accommodate a local geometric constraint of the hub profile. The effective pulse count per revolution is therefore $N_{\text{eff}} = 22$. The wheel diameter is $d = 622$ mm (radius $r = 0.311$ m), which matches the firmware constant `SPEED_DIAM_MM` $= 622$.

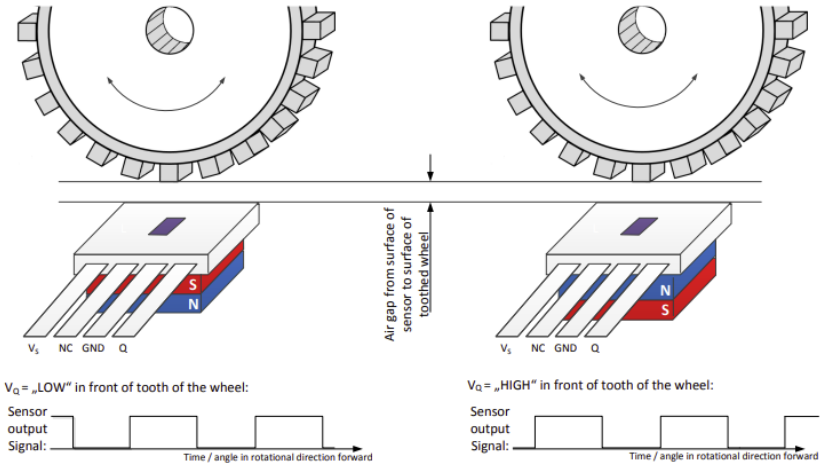

Figure 4.8: Effect of back-bias magnet polarity on the TLE4922 output. Reversing the magnet swaps the HIGH/LOW assignment between tooth and notch passages (source: (INFINEON TECHNOLOGIES AG, 2025)).

Signal acquisition and pulse counting. The sensor outputs are sampled by a periodic acquisition routine driven by a dedicated hardware timer at a fixed rate of $f_s = 5$ kHz. On each invocation the routine reads the logic level of both sensor inputs and applies a synchronous debounce: a level transition is accepted only after $N_d = 6$ consecutive samples (a stable interval of $N_d/f_s = 1.2$ ms) agree on the new level. Each accepted high-to-low transition, corresponding to one magnet passing the sensor, increments a per-wheel pulse counter. The routine executes entirely in IRAM and performs no floating-point arithmetic. The debounce interval is bounded well below the minimum inter-pulse period at the maximum operating speed of the vehicle, so that no genuine pulse is rejected, while transient disturbances shorter than 1.2 ms are suppressed.

At the end of each measurement window of duration $\Delta t = 100$ ms, the main task atomically reads and clears the two pulse counters and derives the linear speed of each wheel from the accumulated count. The measurement window of 100 ms is chosen as a compromise between temporal resolution and the minimum number of pulses required for a reliable speed estimate at low vehicle speeds.

Speed computation. Each magnet passage advances the wheel contact point by a fixed arc length equal to the wheel circumference divided by the effective magnet count,

$$s_p \;=\; \frac{\pi\, d}{N_{\text{eff}}} \;=\; \frac{\pi \times 0.622}{22} \;=\; 0.0888 \text{ m}, \tag{4.25}$$

where $\pi\, d = \pi \times 0.622 = 1.954$ m is the wheel circumference and $N_{\text{eff}} = 22$. Over any window spanning several pulses the mean distance per pulse equals s_p irrespective of the two wider gaps left by the empty template positions, since these contribute proportionally fewer pulses over a longer arc. Given a pulse count k accumulated over the measurement window Δt, the linear speed of the wheel is therefore

$$v_{\text{wheel}} \;=\; \frac{k\, s_p}{\Delta t} \;\text{[m/s]}. \tag{4.26}$$

To suppress the single-pulse quantisation step that is most visible at low speed, the per-wheel speed is passed through a first-order exponential moving-average filter with smoothing factor $\alpha = 0.3$,

$$\hat{v}_{\text{wheel}}[n] \;=\; (1-\alpha)\,\hat{v}_{\text{wheel}}[n-1] \;+\; \alpha\, v_{\text{wheel}}[n]. \tag{4.27}$$

The vehicle longitudinal speed, referenced to the midpoint of the rear axle, is the arithmetic mean of the two filtered wheel speeds:

$$v \;=\; \frac{v_L + v_R}{2}. \tag{4.28}$$

This averaging expression is kinematically exact at the axle midpoint regardless of the instantaneous turning curvature, because the outer wheel speed exceeds the inner wheel speed by precisely the amount dictated by the differential geometry of the turn, and the mean cancels the difference.

Wheel consistency and slip detection. When both sensors are operational, the ratio between the two wheel speeds is monitored as a plausibility indicator. For any geometrically admissible manoeuvre within the rickshaw's steering envelope, this ratio is bounded. When the ratio $\max(v_L, v_R) \,/\, \min(v_L, v_R)$ exceeds the empirical threshold of 1.7, the higher reading is considered unreliable and the vehicle speed is derived from the slower wheel alone, while a slip flag is raised:

$$v_{\text{out}} = \begin{cases} \dfrac{v_L + v_R}{2} & \text{if } \dfrac{\max(v_L, v_R)}{\min(v_L, v_R)} \leq 1.7, \\ \min(v_L,\, v_R) & \text{otherwise (slip flag = 1).} \end{cases} \tag{4.29}$$

If only one sensor is producing valid pulses, its reading is used directly. If neither sensor produces a pulse within a timeout window of 400 ms, the reported speed is set to zero.

Publication pipeline. The firmware publishes the speed data on the micro-ROS topic `/robot/speed_out` at 10 Hz as a `Float32MultiArray` message with five elements:

```
[v_L_ms, v_R_ms, v_avg_ms, v_kmh, slip_flag]
```

where `v_L_ms` and `v_R_ms` are the filtered left and right wheel speeds in m/s, `v_avg_ms` is the averaged longitudinal velocity in m/s, `v_kmh` is the same value in km/h for diagnostic convenience, and `slip_flag` is 1 when Equation (4.29) triggers.

The Python bridge node `speed_bridge.py`, running on the LattePanda as a ROS 2 node, subscribes to `/robot/speed_out`, converts `v_avg_ms` to the standard Autoware `VelocityReport` message format, and publishes it on `/vehicle/status/velocity_status` at 10 Hz for consumption by the longitudinal PID controller and the EKF localiser. Figure 4.9 shows a representative terminal capture of this topic during a test run, confirming the transition from zero speed at standstill to a stable reading of approximately 0.92 m/s as the vehicle accelerates.

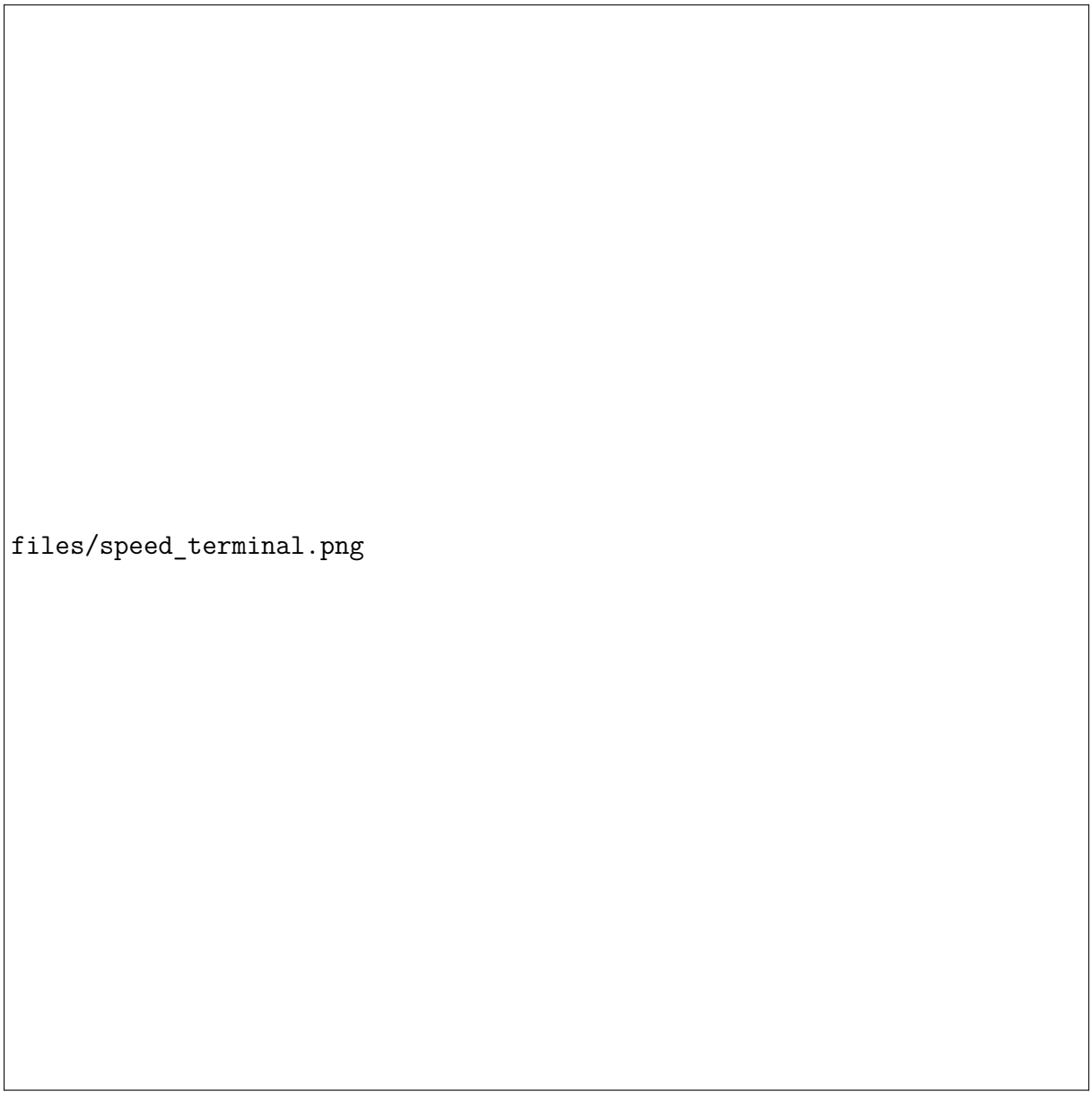

Figure 4.9: Terminal capture of `ros2 topic echo /robot/speed_out` showing the five-element payload: left wheel speed, right wheel speed, average speed in m/s, speed in km/h, and slip flag. Initial zero readings correspond to the vehicle at standstill; subsequent frames show the vehicle accelerating to approximately 0.92 m/s with no slip detected (source: own work).

4.6.2. Mechanical Design and 3D-Printed Sensor Mount

Design requirements. The mechanical mount for the TLE4922 sensors must satisfy three requirements that are partially in tension with one another. The air gap between the sensor aperture and the magnet faces must fall within the 0.5–2.0 mm range specified by the TLE4922 datasheet; this requires both an accurate initial positioning during installation and a means of fine adjustment to compensate for manufacturing tolerances in the hub and in

the printed parts. The sensor body must remain stationary relative to the vehicle frame under wheel vibration, road impacts, and the reaction forces transmitted through the sensor cable, since any oscillation of the sensor relative to the magnet ring produces spurious pulse timing variations. At the same time, the design must allow the sensor to be removed for maintenance or replacement without tools, since access to the rear axle of the rickshaw is constrained by the surrounding frame members.

SolidWorks design and printed assembly. The mounting system was designed in SolidWorks and consists of two separate 3D-printed components. The *sensor bracket* is a clamp-style holder that attaches to the bearing housing of the rear axle. It exposes a radial adjustment slot — a short through-groove in the radial direction — through which the sensor body slides before being locked by an M3 clamping screw. This slot allows the air gap to be set continuously within a 3 mm range during installation and to be relocked without disassembly. A cable retention clip on the bracket body prevents the signal cable from introducing mechanical stresses at the sensor body.

The *magnet ring holder* is a disc-shaped collar that presses onto the wheel hub concentric with the rotation axis and is secured by three M4 grub screws bearing against the hub flange. It carries 24 equally spaced recesses on its outer face, into which the ferromagnetic magnets are press-fitted radially. As noted in Section 4.6.1, 22 of the 24 recesses are occupied; positions 12 and 24 were intentionally left empty to accommodate a local geometric constraint of the hub profile at those angular locations.

Both parts are printed in PLA at a 40 % rectilinear infill, which provides the compressive strength required under the clamping screw torque while keeping the mass below 15 g per part. The printing orientation is chosen such that the layer interfaces of the bracket are perpendicular to the dominant load direction, maximising interlaminar strength. All critical dimensions — the bore diameter of the collar, the slot width, and the sensor recess depth — carry a tolerance of $+0.2$ mm relative to the nominal SolidWorks dimension, which was found empirically to give a zero-clearance press fit with the PLA shrinkage on the available printer.

Installation calibration. Following printing and assembly, the air gap on the right wheel required adjustment after it was found that the initial magnet ring position left the magnets outside the TLE4922 sensing range — the anomaly identified during the initial integration tests. The adjustment procedure consists of loosening the M4 grub screws, translating the ring collar axially by approximately 2 mm toward the sensor face, and relocking. This single adjustment restored pulse detection on the right sensor. The procedure is repeatable and does not require any firmware changes, since the trigger threshold of the TLE4922 adjusts automatically via its internal AGC circuit.

4.6.3. Steering Angle Sensor — Selection, Integration, and Signal Conditioning

Sensor selection rationale. The steering angle sensor must satisfy four requirements that arise directly from the role it plays in the closed-loop lateral controller. It must pro-

vide an absolute measurement of the angular position of the steering shaft, rather than an incremental one, so that no homing procedure is required after each power cycle and the controller receives a valid angle reference immediately on start-up. It must cover the full admissible range of the steering mechanism without ambiguity, which extends to approximately $\pm 40^{\circ}$ of equivalent front-wheel angle. It must offer a resolution finer than the MPC's internal angle granularity, a quantisation step of 0.01° or better is desirable, to avoid introducing a systematic stepping artefact in the lateral feedback. Finally, it must communicate with the ESP32 micro-controller through a standard digital interface without additional signal-conditioning hardware.

Three interface families were considered: a GMR absolute encoder (Infineon TLE5012B), a conventional incremental optical encoder, and a potentiometric position sensor. The optical encoder was dismissed because it is incremental and therefore requires a homing sequence; the potentiometer was dismissed because its analogue output is susceptible to electromagnetic interference from the BLDC motor and requires an ADC channel on the ESP32. The Infineon TLE5012B was selected on the basis of its 15-bit absolute output over a $\pm 180^{\circ}$ range, its standard SPI-compatible SSC digital interface, its $\pm 1^{\circ}$ accuracy, and its tolerance of the mechanical vibrations and temperature variations of an outdoor vehicle environment.

Physical measurement principle — Giant Magneto-Resistance. The TLE5012B is a Giant Magneto-Resistance (GMR) angular position sensor. A small diametrically magnetised permanent magnet is attached to the rotating steering shaft; as the shaft turns, the direction of the magnet's in-plane field vector rotates by the same angle. Four GMR bridge elements, arranged in two Wheatstone bridges oriented at 45° to each other, respond to the cosine and sine components of the in-plane field direction respectively, producing two analogue differential voltages proportional to $\cos\theta$ and $\sin\theta$, where θ is the shaft angle. An on-chip CORDIC processor computes the arctangent of the sine-to-cosine ratio, yielding an absolute angle estimate that is unambiguous over the full 360° range and encoded on 15 bits, which corresponds to a resolution of $360^{\circ}/2^{15} \approx 0.011^{\circ}$ per LSB.

Two properties of the GMR principle are particularly relevant for the application. First, the output depends on the direction of the magnetic field vector rather than on its magnitude; the measured angle is therefore insensitive to slow variations in magnet strength due to temperature or mechanical ageing, and to moderate axial displacements of the magnet relative to the sensor face during assembly. Second, the output is absolute from the first power-on sample: unlike an incremental encoder, no rotation is needed to establish the angle reference, which is essential for a system that must be ready for closed-loop control within the Autoware start-up sequence.

SSC communication interface and SPI frame structure. The TLE5012B exposes its angle register through a Synchronous Serial Communication (SSC) interface, which is the Infineon designation for an SPI-compatible four-wire protocol. The four lines are: clock (CLK), data from sensor to master (DATA / MISO), data from master to sensor (MOSI), and chip select (CS, active low). All four signals operate at 3.3 V logic levels, directly compatible with the ESP32 GPIO inputs and outputs.

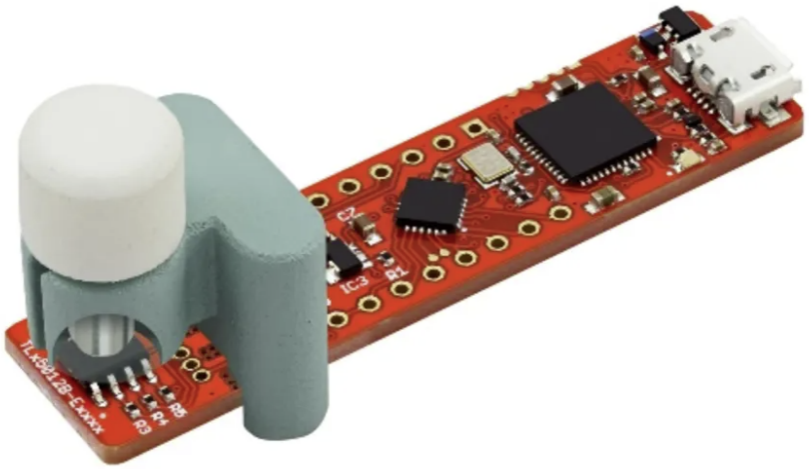

Figure 4.10: TLE5012B evaluation kit with sensor PCB and rotating magnet holder (source: Infineon Technologies).

Each read transaction follows a strict master-initiated request-response structure. The master asserts CS low, transmits a 16-bit command word on MOSI, and simultaneously clocks in a 16-bit response frame on MISO; CS is then de-asserted. The command word for a standard angle register read has a fixed format specified in the TLE5012B user manual: the two most significant bits select the read/write direction and the frame type, and the remaining bits address the target register. The 16-bit response frame carries the 15-bit raw angle value in bits 14 down to 0, and a validity bit (the safety word indicator) in bit 15 that signals whether the on-chip safety logic has detected any anomaly during the conversion. The raw angle value is related to the shaft angle θ in degrees by

$$\theta \;=\; \frac{\text{raw}}{2^{15}} \times 360^{\circ} \;=\; \text{raw} \times 0.0110^{\circ}, \tag{4.30}$$

which gives a full-scale range of $\pm 180^{\circ}$ (signed two's complement representation).

The validity bit is exposed to the ROS 2 layer as the third element of the `/robot/steer_out` message and is used by the software filter as the primary quality gate on every sample, as described below.

Hardware configuration and SPI clock selection. The TLE5012B is connected to the HSPI bus of the ESP32-EVB using four dedicated GPIO pins: CLK on GPIO 14, MISO on GPIO 12, MOSI on GPIO 13, and CS on GPIO 15. The physical routing of the four-wire cable from the sensor, mounted on the steering column, to the ESP32 development board has a total length of 90 cm. To ensure reliable SPI communication over this cable length, the clock frequency was reduced from the default 1 MHz to 100 kHz; the reason for this choice is detailed in the anomaly diagnosis below.

Zero-offset calibration and permanent NVS storage. A one-time calibration step is required to align the TLE5012B angular reference with the mechanical zero of the steering linkage, defined as the position of the front wheel when pointing straight ahead. The cal-

ibration procedure is as follows: the steering mechanism is centred manually by physical alignment; a ROS 2 command is then sent on the `/robot/data_in` topic with the calibration flag set to 1.0 in the third element of the payload:

```
ros2 topic pub -once /robot/data_in
std_msgs/msg/Float32MultiArray "{data:  [0.0, 0.0, 1.0]}"
```

On receipt of this command, the ESP32 firmware reads the current raw sensor output and stores it as the angular reference in the NVS (Non-Volatile Storage) flash partition of the micro-controller. On all subsequent power cycles, the stored offset is loaded automatically and subtracted from every reading, so that the published angle represents the signed deviation from the straight-ahead position. The NVS storage ensures that the calibration survives an unlimited number of power cycles and firmware updates without the need for re-calibration.

Observed signal anomalies and diagnosis. Four distinct anomalies were identified during the integration of the steering sensor chain.

The first and most critical was the appearance of spurious readings of approximately 132° while the actual steering angle was well below 40°. These readings carried a validity bit of 1, making them indistinguishable from genuine measurements by any check on the sensor's own status output. The affected samples amounted to approximately 20% of all frames at the 1 MHz clock. The root cause was a single-bit framing error in the 16-bit SPI response word caused by signal integrity degradation on the 90 cm cable at 1 MHz: one bit shifted in the received frame adds exactly 128.22° to the decoded angle, consistent with the observed spurious values. Reducing the SPI clock to 100 kHz, at which the 10 μs bit period is safely larger than the worst-case signal settling time on the cable, eliminated all bit-shift errors completely.

The second anomaly was stationary jitter: the published steering angle oscillated by approximately $\pm 0.02^\circ$ while the steering mechanism was at rest, even after the SPI clock fix. This is the quantisation noise floor of the TLE5012B at the LSB level ($\approx 0.011^\circ$), which causes consecutive readings to alternate between two or three adjacent LSB values.

The third anomaly was the occasional appearance of physically impossible angle values, such as a sudden reading of -2.78° from a position of 15°, typically with validity bit $= 0$.

The fourth anomaly was a filter lock after a cold reboot of the ESP32 while the steering wheel was in a position different from the calibration zero: on restart, the rate-of-change filter rejected all incoming readings because they differed from the filter's last stored state by more than the jump threshold, causing Autoware to display a stale or invalid steering angle indefinitely until the Autoware stack was restarted.

Software signal conditioning — four-stage filter in `steering_bridge.py`. A four-stage filter was implemented in Python within the ROS 2 node `steering_bridge.py`, executing on the LattePanda before any angle value is published to Autoware. The filter is placed at the ROS 2 level rather than in the ESP32 firmware for three reasons: no firmware re-flash is required to adjust the filter parameters; all filtering occurs before the data reaches

the Autoware control stack, so Autoware always receives already-clean data; and the filter thresholds can be modified on the fly by editing a single Python node without interrupting the embedded system. Each incoming sample is processed through the four stages in sequence; a sample that fails any stage is silently discarded and the last accepted value is retained.

Stage 0 — Validity flag check. Any sample whose TLE5012B validity bit is zero is immediately rejected. This stage eliminates most of the impossible-value anomalies and requires no parameter tuning.

Stage 1 — Physical limit check. Any sample whose absolute value exceeds 55° is rejected. The physical maximum steering angle of the rickshaw is $\pm 40.1^\circ$ (± 0.70 rad); the threshold of $\pm 55^\circ$ is set with a 15° margin above this limit to accommodate installation tolerances, while still being far below the spurious 132° readings. This stage eliminates all bit-shift-induced spikes permanently and unconditionally, with zero false rejections.

Stage 2 — Rate-of-change check with adaptive reset. Any sample that differs from the preceding accepted value by more than 10° in a single inter-sample interval is rejected. The physically observed maximum steering rate during fast actuation is approximately 5° per sample at the 46 Hz publication rate, so the 10° threshold provides a factor-of-two safety margin. A two-second timeout is incorporated: if no valid sample has been accepted for more than two seconds — for example, because the ESP32 was rebooted while the steering was in a non-zero position — the filter state is reset unconditionally and the next incoming sample is accepted regardless of its difference from the previous state. This timeout resolved the post-reboot lock-out anomaly described above.

Stage 3 — Deadband suppression. Any sample that differs from the last published value by less than 0.1° is replaced by the last published value rather than updating the output. The deadband is set to five times the sensor quantisation noise floor ($5 \times 0.02^\circ = 0.10^\circ$) but is smaller than the smallest intentional steering command issued by the MPC; it therefore eliminates stationary jitter completely without attenuating any real steering motion.

Publication pipeline. The ESP32 firmware publishes the raw steering data on the micro-ROS topic `/robot/steer_out` at approximately 46 Hz as a `Float32MultiArray` message with three elements:

`[angle_deg, angle_rad, valid_flag]`

where `angle_deg` is the calibrated angle in degrees (zero offset already subtracted), `angle_rad` is the same value converted to radians, and `valid_flag` is 1 when the TLE5012B safety word indicates a valid conversion and 0 otherwise. After the four-stage filter, the node converts the accepted angle to the steering tire angle by dividing by the steering ratio r_s:

$$\delta = \frac{\theta_{\text{shaft}}}{r_s}, \tag{4.31}$$

clamps the result to $[-0.70, +0.70]$ rad, and publishes it as an Autoware `SteeringReport` on `/vehicle/status/steering_status` at 46 Hz. The steering ratio $r_s = 1.0$ is currently used as a provisional value; its exact value will be determined by direct measurement of shaft rotation versus front-wheel angle after the final mechanical installation of the sensor.

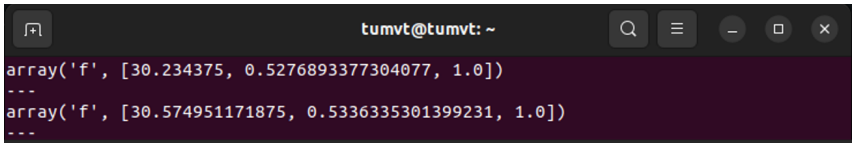

Figure 4.11: Terminal capture of `ros2 topic echo /vehicle/status/steering_status` showing a stable, non-jittering steering angle value, confirming effective operation of the four-stage filter in the live ROS 2 environment.

4.6.4. End-to-End Software Integration on the Vehicle

Bridge architecture and topic mapping. The three bridge nodes whose ROS 2 topic connections are shown in Section 3.3.2 are described below with their implementation details. The complete software pipeline between Autoware and the physical hardware is implemented as three Python ROS 2 nodes, launched together with the Autoware stack on the LattePanda. Their roles and the corresponding topic mapping are as follows:

- `control_bridge.py`: subscribes to `/control/command/control_cmd` (Autoware `AckermannControlCommand`), extracts the desired speed and steering angle, applies the hard safety bounds defined in Section 4.5, and publishes the result on `/robot/data_in` (`Float32MultiArray`) for consumption by the ESP32 firmware. A watchdog timer sets both commands to zero if no Autoware message is received within a configurable timeout.
- `speed_bridge.py`: subscribes to `/robot/speed_out` (firmware wheel-speed data), converts `v_avg_ms` to the Autoware `VelocityReport` format (longitudinal velocity; lateral velocity set to zero; heading rate from the EKF), and publishes on `/vehicle/status/velocity_status` at 10 Hz.
- `steering_bridge.py`: subscribes to `/robot/steer_out` (raw angle from TLE5012B), applies the four-stage filter described in Section 4.6.3, converts the calibrated angle to the Autoware `SteeringReport` format, and publishes on `/vehicle/status/steering_status` at 46 Hz.

Vehicle parameter calibration. Two vehicle parameters had to be set consistently across the firmware and the Autoware vehicle model to ensure that the speed and angle values computed by the sensor pipeline match the internal quantities used by the controllers.

The wheel radius was measured experimentally by rolling the vehicle over a known distance and counting wheel revolutions; the resulting value $r = 0.311$ m corresponds to a wheel diameter of 622 mm and is set as the constant `SPEED_DIAM_MM` $= 622$ in the firmware. The same value is entered in the Autoware vehicle configuration file (`vehicle_info.param.yaml`) as `wheel_radius`. The maximum steering angle is set to $\delta_{\max} = 0.70$ rad in both the Autoware model (`max_steer_angle`) and in the safety clamp of `steering_bridge.py`, ensuring that neither layer permits a command or a measurement outside the physically admissible range.

FastDDS communication profile. The micro-ROS transport between the LattePanda and the ESP32 operates over UDP on the local network. To prevent unreliable topic dis-

covery under transient packet loss, a custom FastDDS profile was configured: the lease duration for participant liveness is set to 3 s, and each participant broadcasts five initial announcement packets at 100 ms intervals on start-up. This configuration ensures that both endpoints reach a consistent connected state within at most 0.5 s of the ESP32 boot, which is well within the watchdog timeout of the control bridge, and that a transient loss of connectivity triggers a lease expiry and a clean reconnection rather than a silent loss of messages.

Integration verification. The correctness of the complete pipeline was verified in three stages. First, all six standard `/vehicle/status/*` topics were confirmed as active and publishing at the expected rates using `ros2 topic hz`. Second, the steering angle reported on `/vehicle/status/steering_status` was verified to track the physical wheel position without spurious spikes by observing the Autoware RViz vehicle model rotating smoothly as the steering wheel was turned. Third, the velocity reported on `/vehicle/status/velocity_status` was verified to read exactly zero when the vehicle was stationary and to increase proportionally to wheel rotation when a rear wheel was turned by hand.

Figure 4.12 shows a representative capture of the steering verification. The terminal displays two consecutive `/robot/steer_out` frames published by the ESP32 firmware: `[30.23°, 0.5277 rad, 1.0]` and `[30.57°, 0.5336 rad, 1.0]`, where the third element confirms a valid TLE5012B conversion. Simultaneously, the Autoware RViz steering indicator displays $30.6°$, consistent with the filtered and converted output of `steering_bridge.py`. The agreement between the raw ESP32 output and the Autoware display confirms correct end-to-end propagation through the micro-ROS transport, the four-stage filter, and the `SteeringReport` interface.

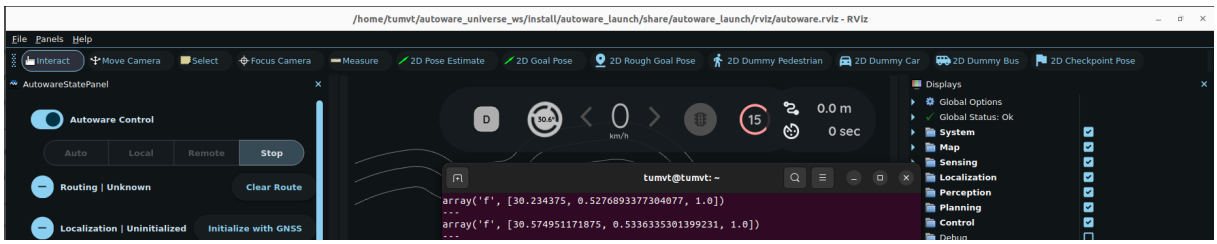

Figure 4.12: End-to-end steering verification: the terminal shows two consecutive `/robot/steer_out` frames from the ESP32 — `[30.23°, 0.5277 rad, 1.0]` and `[30.57°, 0.5336 rad, 1.0]` — while the Autoware RViz steering indicator simultaneously displays $30.6°$, confirming correct propagation through the `steering_bridge` pipeline with validity flag $= 1$ on both frames.

Video demonstrations of the complete sensor pipeline are provided as supplementary material. In each demonstration, a `ros2 topic pub` command is issued on `/robot/data_in` to command the actuators directly, and the corresponding sensor feedback is observed simultaneously on a second terminal.

The first video demonstrates the wheel-speed sensor pipeline. A payload of `[0.30, 0.0, 0.0]` is published on `/robot/data_in`, commanding a normalised throttle duty of 0.30 with zero steering. With the vehicle lifted off the ground, the rear wheels spin in response to the

motor command. Simultaneously, a second terminal displays the live `/robot/speed_out` topic, showing the measured wheel speed increasing as the wheels accelerate, confirming correct operation of the TLE4922 sensors and end-to-end forwarding to Autoware via `speed_bridge.py` as shown in section 4.6.1.

The second video demonstrates the steering angle sensor pipeline. A payload of `[0.0, 0.25, 0.0]` is published on `/robot/data_in`, commanding zero throttle and a normalised steering deflection of 0.25 on a scale of $[-1, 1]$. The ESP32 firmware translates this into the corresponding RoboClaw actuator byte and drives the steering mechanism. Simultaneously, a second terminal displays the live `/robot/steer_out` topic, showing the measured steering angle increasing in response to the command, with the validity flag confirming a valid TLE5012B conversion on each frame. The physical deflection of the steering mechanism is visible in the video, confirming end-to-end actuation and sensing through `steering_bridge.py` as shown in section 4.6.3.[1]

[1] Steering sensor pipeline video: `https://syncandshare.lrz.de/getlink/fiRrRfgwGHkLXipnTDUM2D/`

5. Experimental Validation and Results

The validation of the control architecture presented in this chapter covers two complementary aspects. The first is the functional verification of the sensor integration and actuation pipeline: the wheel-speed and steering-angle sensors were tested by commanding the steering actuator and motor directly and verifying that the measured feedback matched the commanded motion. The second is the software-level verification of the control chain: the lateral MPC and longitudinal PID controllers were exercised by running the full Autoware stack and observing the controller outputs in response to planned trajectories. The objective of this chapter is to present, in a structured manner, the test set-up, the metrics adopted for the evaluation, and the results obtained at the time of writing, together with an honest account of what has and has not yet been validated on the physical platform.

5.1. Experimental Platform and Test Environment

The experimental platform is the semi-autonomous cycle rickshaw described in Chapter 3, in the integration state reached at the time of the validation campaign: full Autoware stack on the LattePanda, control bridge and sensor bridges in operation, embedded firmware running on the ESP32, and wheel-speed and steering-angle sensors connected and calibrated as described in Section 4.6.

The validation is performed exclusively on the physical vehicle. The test site is a large open tarmac area available to the Chair of Traffic Engineering and Control of the Technical University of Munich, shown in Figure 5.1. The area provides a flat, uniform surface with painted reference markings, is free of road furniture and kerbs that could endanger the vehicle at the considered operating speeds, and can be cleared of pedestrians and other vehicles for the duration of each test run. These properties make it well suited to the systematic evaluation of a low-speed autonomous platform: the surface conditions are reproducible across runs, the available space is large enough to accommodate the full planned trajectory without geometric constraints, and the absence of traffic eliminates external disturbances that would otherwise confound the comparison between controller variants.

The choice of validating directly on the physical vehicle rather than through an intermediate virtual environment is motivated by two considerations. First, it produces results that reflect the actual behaviour of the hardware in full, including the non-idealities that a simulation can only approximate: the true actuation delay of the steer-by-wire system, the friction and backlash of the steering mechanism, the quantisation noise of the wheel-speed and

steering-angle sensors, and the small surface irregularities of the tarmac. Second, it eliminates the risk that conclusions drawn in a virtual environment do not transfer to the physical platform, which is particularly relevant for a non-standard vehicle such as the rickshaw whose dynamical model is not available with high fidelity.

Figure 5.1: Aerial view of the outdoor test area used for the validation campaign.

5.2. Test Scenarios

The test scenarios are designed to expose the control architecture to the manoeuvres that are most relevant for the rickshaw's intended operating domain and for the metrics defined in Section 5.5. All scenarios are executed at the maximum operational speed of $v_{\max} = 1.66$ m/s (≈ 6 km/h), consistent with the safety policy adopted for the current validation campaign, and on the test area described in Section 5.1. The reference trajectory used across all scenarios is planned in Autoware and is visualised in Figure 5.2.

Three classes of scenarios are considered, each targeting a specific aspect of the control architecture.

The *straight-line tracking scenario* uses a straight segment of the planned path. It isolates the longitudinal channel and provides a clean measurement of the speed-tracking performance, of the steady-state velocity error, and of the actuation smoothness of the longitudinal PID under the four parameter profiles. The absence of lateral demand on this segment means that any lateral deviation observed is attributable exclusively to the EKF localisation and to the straight-line stability of the platform, rather than to an active steering command.

The *constant-curvature scenario* uses a curved segment of the planned path at a fixed turning radius. It isolates the lateral channel and exposes the steady-state lateral error, the heading error, and the steering command smoothness under the four MPC profiles.

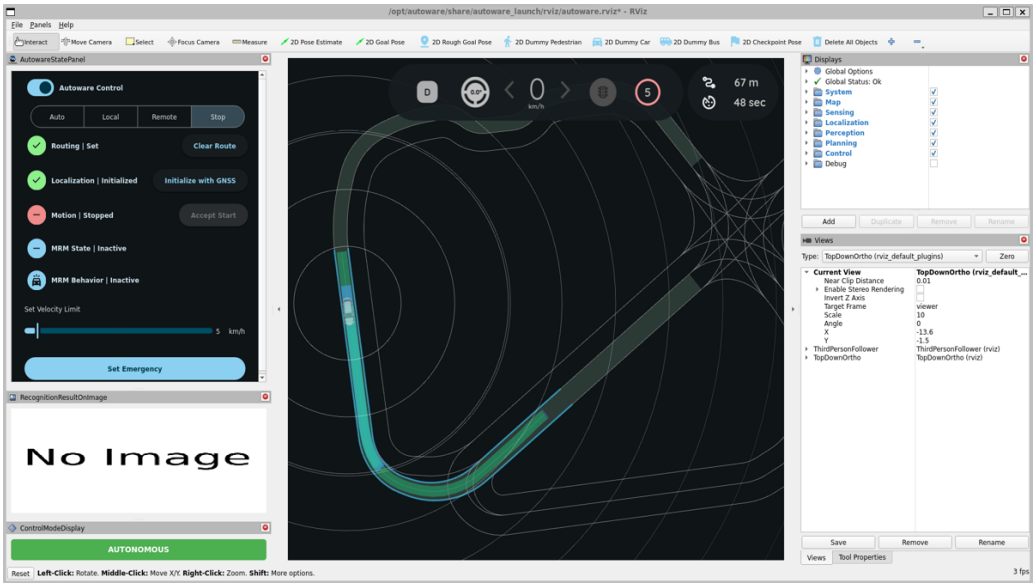

Figure 5.2: Reference trajectory planned in Autoware and visualised in RViz.

The curvature is large enough to remain within the steering bound $\delta_{\max} = 0.70$ rad at the considered speed, while being small enough to produce a measurable and sustained steering activity that distinguishes the four profiles from one another.

The *full-path scenario* executes the complete reference trajectory of Figure 5.2, combining straight segments, the curved section, and the entry and exit transitions. This scenario exercises both channels simultaneously and in their coupled interaction, and provides the most realistic assessment of the integrated control architecture under conditions representative of a low-speed urban route. The transition regions — where the curvature changes abruptly from zero to its maximum value and back — are the most demanding segments for both channels and are therefore the primary focus of the comparative analysis in Section 5.7.

Each scenario is executed at least three times for each controller variant, in order to allow a basic statistical characterisation of the metrics and to mitigate the effect of the unavoidable run-to-run variability attributable to small differences in the vehicle's initial position relative to the planned path.

5.3. Baseline and Compared Controller Variants

The compared controller variants are the four longitudinal PID profiles and the four lateral MPC profiles defined in Section 4.4. To keep the comparison tractable, the longitudinal and the lateral profiles are paired according to a consistent design priority: the baseline longitudinal PID is associated with the baseline lateral MPC; the comfort longitudinal PID is associated with the comfort lateral MPC; and similarly for the tracking and the robust profiles. This pairing yields four complete controller variants, denoted respectively as Baseline, Comfort, Tracking, and Robust.

The Baseline variant reproduces the default Autoware tuning and serves as the reference

against which the others are compared. The Comfort variant prioritises the smoothness of the actuation, with lower gains and stronger filtering on the longitudinal channel and with higher penalties on the steering input and on the lateral jerk on the lateral channel. The Tracking variant prioritises the fidelity to the reference trajectory, with higher gains on the longitudinal channel and with the strongest penalty on the lateral error on the lateral channel. The Robust variant prioritises the stability under noise and delay, with reduced integral action, increased filtering, and increased modelled delay on both channels.

5.4. Data Acquisition, Synchronisation, and Preprocessing

All the experiments are recorded as ROS 2 bag files containing the relevant topics: the planned trajectory, the actual pose and twist of the vehicle as estimated by the EKF, the actuation commands published by the controller, the wheel-speed and steering-angle measurements published by the sensor bridges, and any diagnostic flag produced by the safety supervision layer.

The recorded topics are not perfectly synchronous, since they are produced at different rates by different nodes; the planner topic is updated at approximately 11 Hz, the controller topic at approximately 43 Hz, the velocity report at 10 Hz, and the steering report at 46 Hz. The preprocessing pipeline applies a uniform time grid at 50 Hz to the recorded data: each topic is interpolated by zero-order hold from the closest preceding sample, which preserves the causal structure of the controller and avoids the introduction of artificial smoothness.

A second preprocessing step concerns the alignment between the planned and the actual trajectory. The planner publishes a trajectory parameterised by time, and the actual pose evolves on a possibly different time line because of the actuator delays. To produce an unambiguous tracking error, the actual pose is projected, at each time step, onto the planned trajectory in space, and the corresponding nearest-point distance is reported.

5.5. Performance Metrics and Evaluation Procedure

The evaluation of the controller variants is based on a set of complementary metrics, organised in three groups: tracking accuracy metrics, control quality metrics, and constraint compliance metrics.

The tracking accuracy metrics include the lateral deviation, defined as the perpendicular distance between the actual position of the vehicle and the closest point of the planned trajectory; the heading error, defined as the difference between the actual heading and the planned heading at the corresponding point; the speed-tracking error, defined as the difference between the actual longitudinal speed and the planned speed; and the root-mean-square trajectory error, computed as the root-mean-square of the lateral deviation over the entire scenario.

The control quality metrics include the overshoot, defined as the maximum excursion of the controlled variable beyond the reference following a step-like change; the settling

time, defined as the time required for the controlled variable to remain within a prescribed tolerance band around the reference; the steering command smoothness, characterised by the root-mean-square of the steering rate; and the rate of change of the actuation signals, characterised by the maximum and the root-mean-square of the steering rate and of the acceleration command.

The constraint compliance metrics include the percentage of time during which the steering angle, the steering rate, the speed, and the lateral acceleration remain within the prescribed bounds, and the absence of unsafe or unstable control responses, verified by visual inspection of the time series and by the absence of safety supervision interventions.

Each metric is computed for each run of each scenario for each controller variant; for each combination, the mean and the standard deviation across the runs are reported, as a basis for the comparative analysis of Section 5.7.

5.6. Experimental Validation on the Vehicle

This section presents the experimental results obtained by running the four controller variants defined in Section 5.3 on the physical rickshaw along the reference trajectory described in Section 5.2. The results are organised by control channel: the longitudinal PID is evaluated first, followed by the lateral MPC. For each channel the parameter values of the four profiles are summarised in a table, followed by the corresponding measured response plots, so that the numerical choices and their practical outcomes can be read together. All tests were conducted on the test area described in Section 5.1 at speeds not exceeding the maximum operational limit of $v_{\max} = 1.66$ m/s.

5.6.1. Longitudinal PID — Parameter Profiles and Results

Table 5.1 summarises the five tunable parameters of the longitudinal PID controller across the four profiles. The parameter names follow the Autoware `pid_longitudinal_controller` configuration convention; the corresponding mathematical symbols are those introduced in Section 4.3.1.

Table 5.1: Longitudinal PID parameter values for the four controller profiles.

Parameter	Symbol	Baseline	Comfort	Tracking	Robust
`delay_compensation_time` [s]	τ_d	0.17	0.17	0.17	0.35
`Kp`	K_p	1.00	0.60	1.60	0.90
`Ki`	K_i	0.10	0.06	0.08	0.03
`Kd`	K_d	0.00	0.05	0.12	0.08
`lpf_vel_error_gain`	α_{lpf}	0.90	0.70	0.95	0.60

Baseline profile. The Baseline PID response is shown in Figure 5.3. The measured speed reaches the target of ≈ 1.39 m/s within roughly three seconds from the initial condition, with a small initial overshoot of $\approx 3.5\%$ before settling onto the reference. Throughout the steady cruising phase the measured speed tracks the constant target within a narrow band, confirming that the moderate proportional gain $K_p = 1.00$ combined with the integral term $K_i = 0.10$ eliminates the residual steady-state error. A single deceleration event is visible in the *measured* speed at $t \approx 42$ s, where it dips to ≈ 1.22 m/s and then recovers to the reference; the target speed remains constant throughout this interval. This indicates a localised external disturbance acting on the vehicle at the curvature in the planned trajectory (Figure 5.2) rather than a commanded speed reduction, and the controller restores tracking after the event.

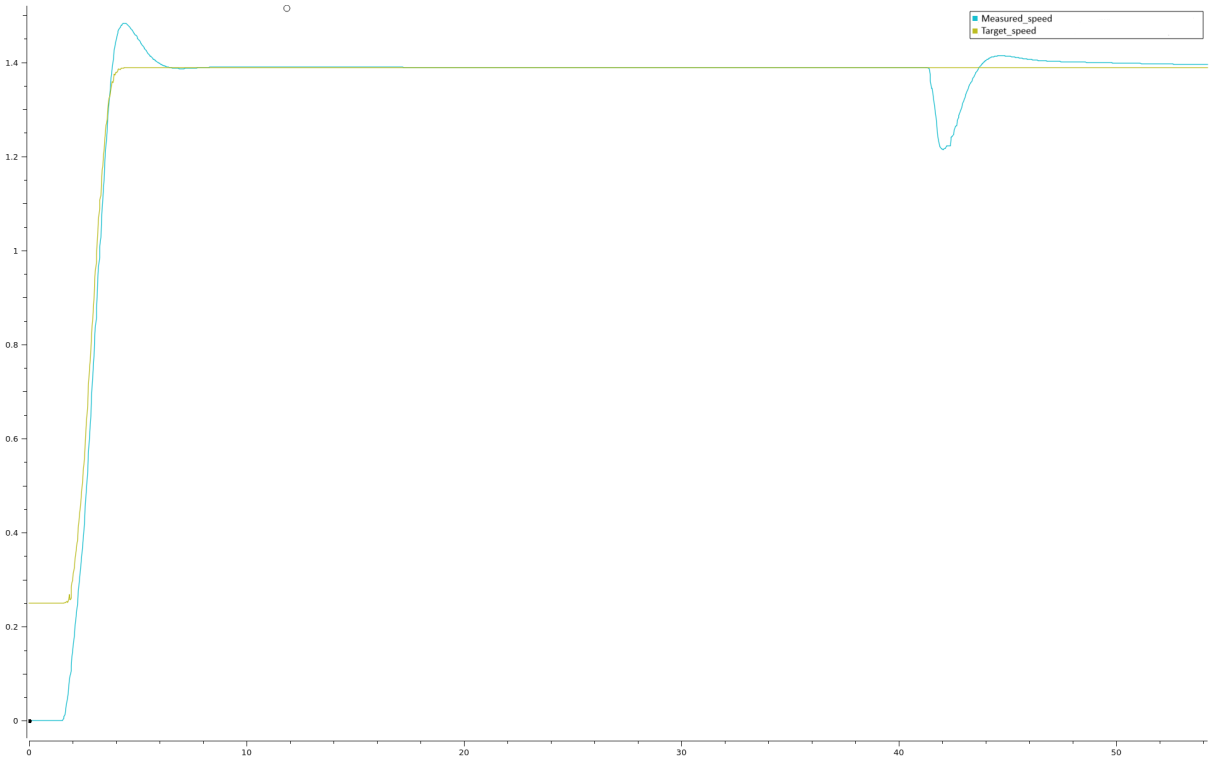

Figure 5.3: Longitudinal PID — Baseline profile: target and measured speed.

Comfort profile. Figure 5.4 shows the Comfort PID response. The reduced proportional gain $K_p = 0.60$ and the stronger low-pass filter ($\alpha_{\text{lpf}} = 0.70$) produce a noticeably smoother response than the Baseline: the initial overshoot of the measured speed is small and the speed rises to the constant target without sustained oscillation, where it remains stable for the duration of the run. The added derivative term $K_d = 0.05$ provides light damping that suppresses any tendency to oscillate about the reference. No disturbance event is present in the interval recorded for this profile.

Figure 5.4: Longitudinal PID — Comfort profile: target and measured speed.

Tracking profile. The Tracking PID result is presented in Figure 5.5. The measured speed accelerates to the target with a sharp rise and a brief overshoot, consistent with the high proportional gain $K_p = 1.60$, then tracks the constant target closely. The recording contains two consecutive run segments: a full stop at $t \approx 38$ s, where both target and measured speed go to zero as the vehicle reaches the end of the planned path, followed by a second run beginning at $t \approx 42$ s. The re-acceleration in the second segment is clean and repeatable. A small disturbance event is also visible at $t \approx 48$ s, where the measured speed dips briefly below the constant target and recovers; the shallow depth and rapid recovery of this dip — compared with the other profiles — reflect the strong disturbance rejection provided by the high proportional and derivative gains ($K_p = 1.60$, $K_d = 0.12$).

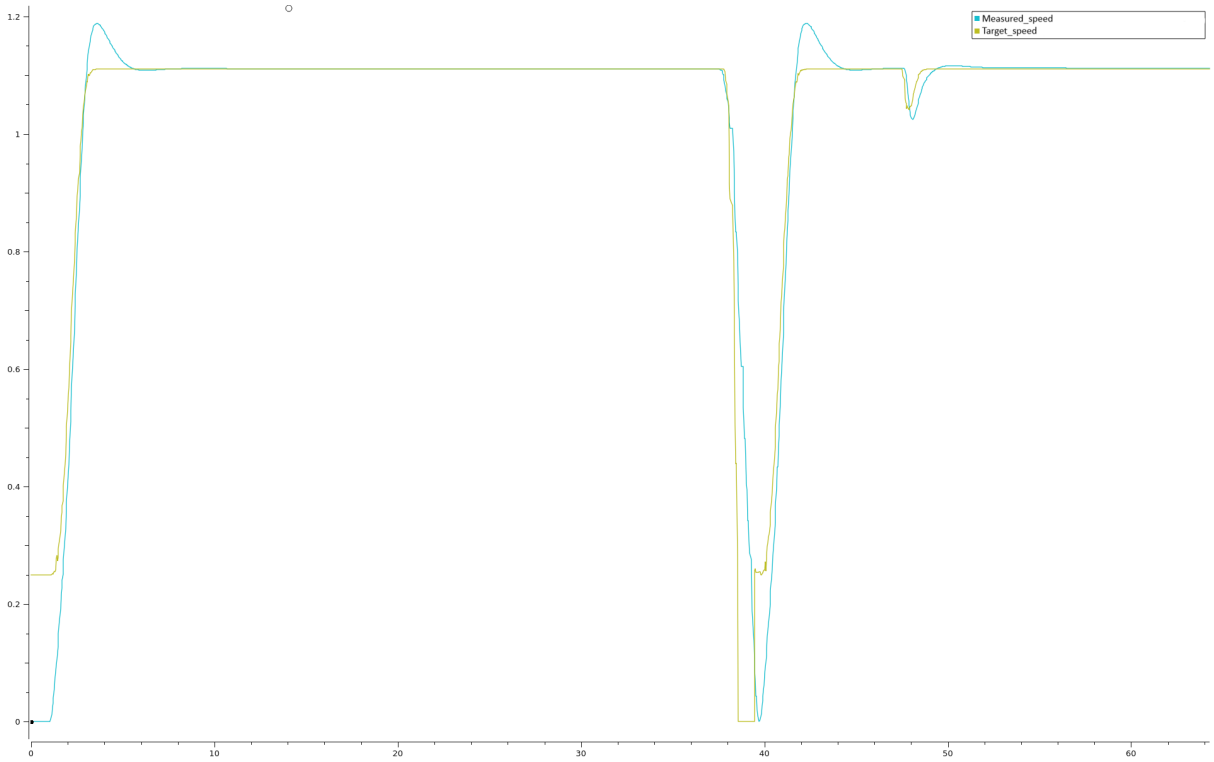

Figure 5.5: Longitudinal PID — Tracking profile: target and measured speed.

Robust profile. Figure 5.6 shows the Robust PID result. The target speed (cyan) is a clean step that settles to the cruise value of ≈ 1.11 m/s and remains constant for the rest of the run. The measured speed (purple) rises to the target with a small overshoot and then exhibits a pronounced deceleration event at $t \approx 12$–15 s, dipping to ≈ 0.88 m/s before recovering and tracking the reference closely thereafter. Because the target remains constant through this interval, the dip originates on the response side: the vehicle is physically disturbed at the curvature in the planned trajectory (Figure 5.2) and the controller corrects the resulting velocity error. The conservative tuning of the Robust profile — low integral gain $K_i = 0.03$, strong delay compensation $\tau_d = 0.35$ s, and a weaker error filter ($\alpha_{\text{lpf}} = 0.60$) — prioritises smoothness and stability over aggressive disturbance rejection, which accounts for the larger depth and slower recovery of this dip relative to the Tracking and Baseline profiles. The behaviour is therefore consistent with the intended design trade-off of the profile rather than an instability: the same disturbance produces a progressively smaller dip as the proportional and derivative authority is increased across the profiles.

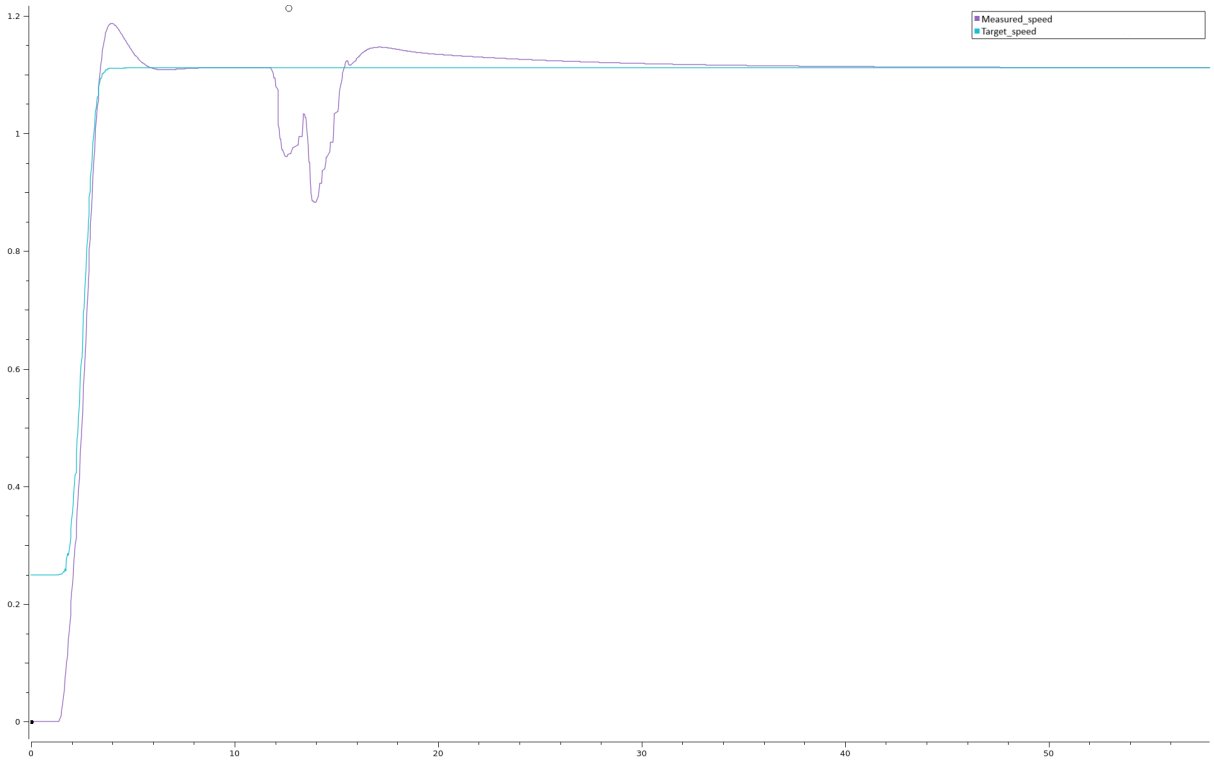

Figure 5.6: Longitudinal PID — Robust profile: target and measured speed.

5.6.2. Lateral MPC — Parameter Profiles and Results

Table 5.2 summarises the four tunable parameters of the lateral MPC controller across the four profiles. The parameter names follow the Autoware `mpc_lateral_controller` configuration convention; the corresponding mathematical symbols are those introduced in Section 4.3.2. In several of the following recordings a large transient in the measured steering angle is visible in the first one to two seconds, attributable to sensor initialisation at start-up as described in Section 4.6.3; this artefact does not affect the operative phase of each test and is disregarded in the analysis.

Table 5.2: Lateral MPC parameter values for the four controller profiles.

Parameter	**Symbol**	**Baseline**	**Comfort**	**Tracking**	**Robust**
`mpc_weight_lat_error`	w_{lat}	1.0	0.6	2.0	1.0
`mpc_weight_steering_input`	w_{δ}	1.0	2.0	0.5	1.2
`mpc_weight_lat_jerk`	w_{jerk}	0.1	1.0	0.05	0.2
`input_delay` [s]	τ_s	0.24	0.24	0.20	0.3

Baseline profile. The Baseline MPC steering response is shown in Figure 5.7. From $t \approx 2$ s the controller operates normally: the measured steering angle follows the target

command through the approach phase, peaks at ≈ 0.27 rad at $t \approx 27$ s during the constant-curvature segment, and returns progressively toward zero. The oscillations visible in both signals during the descent phase reflect the combined effect of the sensor quantisation noise and the MPC's reaction to small path deviations.

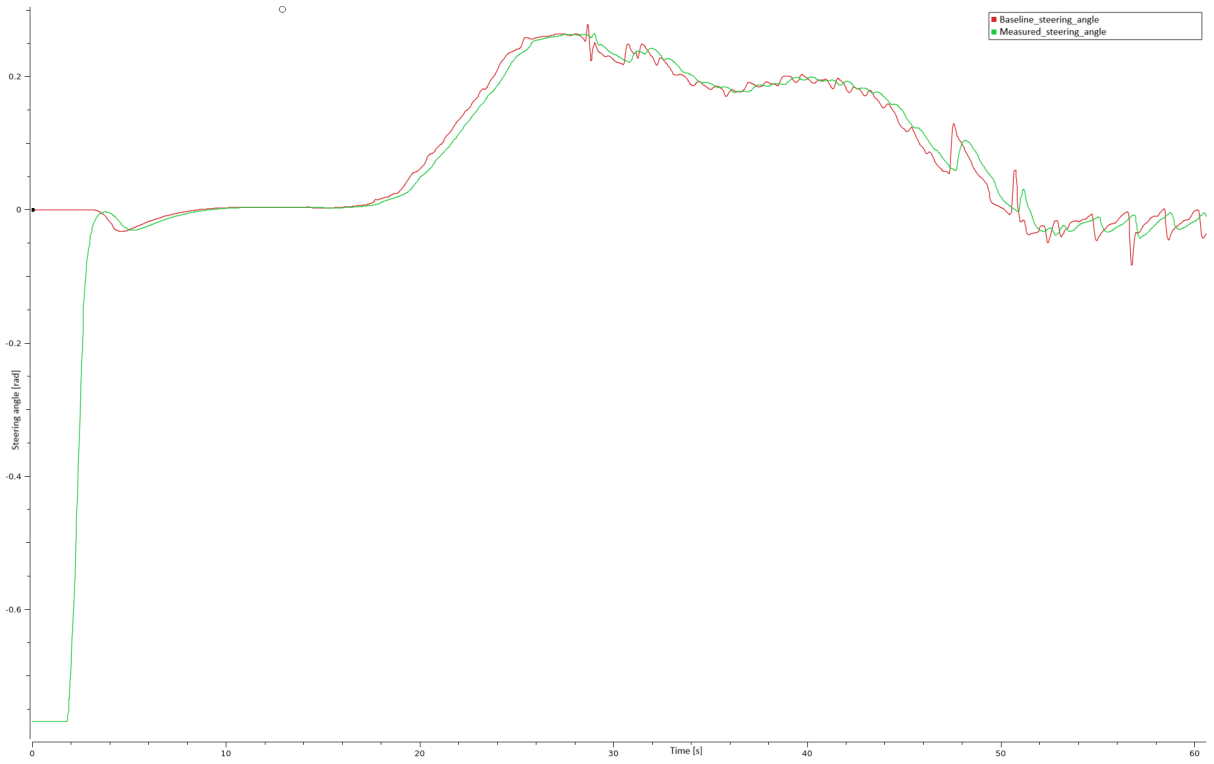

Figure 5.7: Lateral MPC — Baseline profile: target and measured steering angle.

Comfort profile. Figure 5.8 presents the Comfort MPC result. The maximum target steering angle reached during the curved section is ≈ 0.22 rad, noticeably lower than the 0.27 rad of the Baseline profile. This reduction is a direct consequence of the highest steering input weight $w_\delta = 2.0$, which strongly penalises large steering commands, and of the highest jerk weight $w_{\text{jerk}} = 1.0$, which penalises rapid changes in the lateral acceleration. The result is a smoother, more conservative steering profile that sacrifices some path-following tightness in favour of passenger comfort.

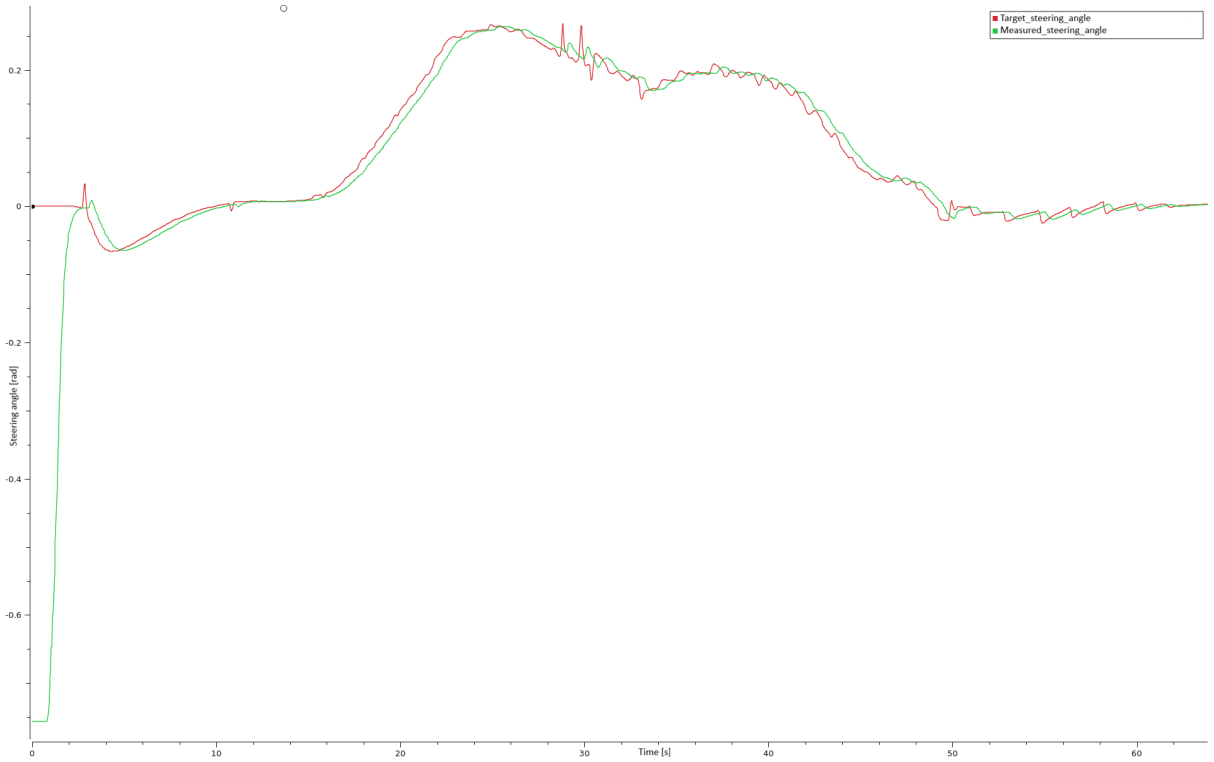

Figure 5.8: Lateral MPC — Comfort profile: target and measured steering angle.

Tracking profile. The Tracking MPC result is shown in Figure 5.9. The target steering angle is visibly more dynamic than in the other profiles throughout the run: this is the expected consequence of the highest lateral error weight $w_{\text{lat}} = 2.0$, which strongly drives the MPC to correct any path deviation, combined with the lowest steering input penalty $w_\delta = 0.5$, which does not restrain the size of the command. The measured angle follows the target with a small but visible lag, consistent with the modelled input delay $\tau_s = 0.20$ s. The peak steering command reaches ≈ 0.27 rad, similar to the Baseline.

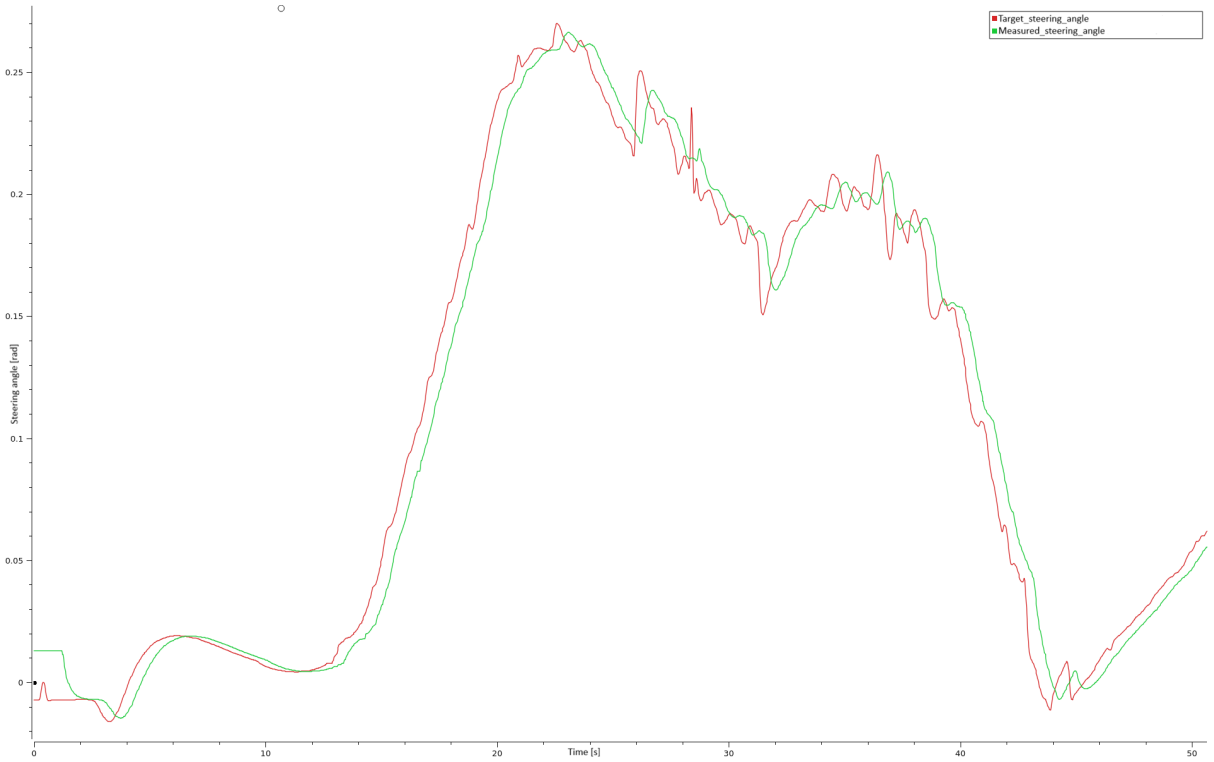

Figure 5.9: Lateral MPC — Tracking profile: target and measured steering angle.

Robust profile. Figure 5.10 shows the Robust MPC result. The operative response follows the path curvature profile reaching a peak of ≈ 0.27 rad, after which both signals show more visible oscillation during the descent phase compared to the Baseline and Comfort profiles. This behaviour reflects the conservative nature of the Robust parameter set: the highest modelled input delay $\tau_s = 0.30$ s causes the MPC to anticipate a larger actuator lag, leading to corrective commands that are slightly out of phase with the real system response when the actual delay is smaller than modelled. The Robust profile is therefore most appropriate for operating conditions where the true actuator delay is expected to vary or exceed the nominal value, accepting a moderately noisier steady-state response as the trade-off.

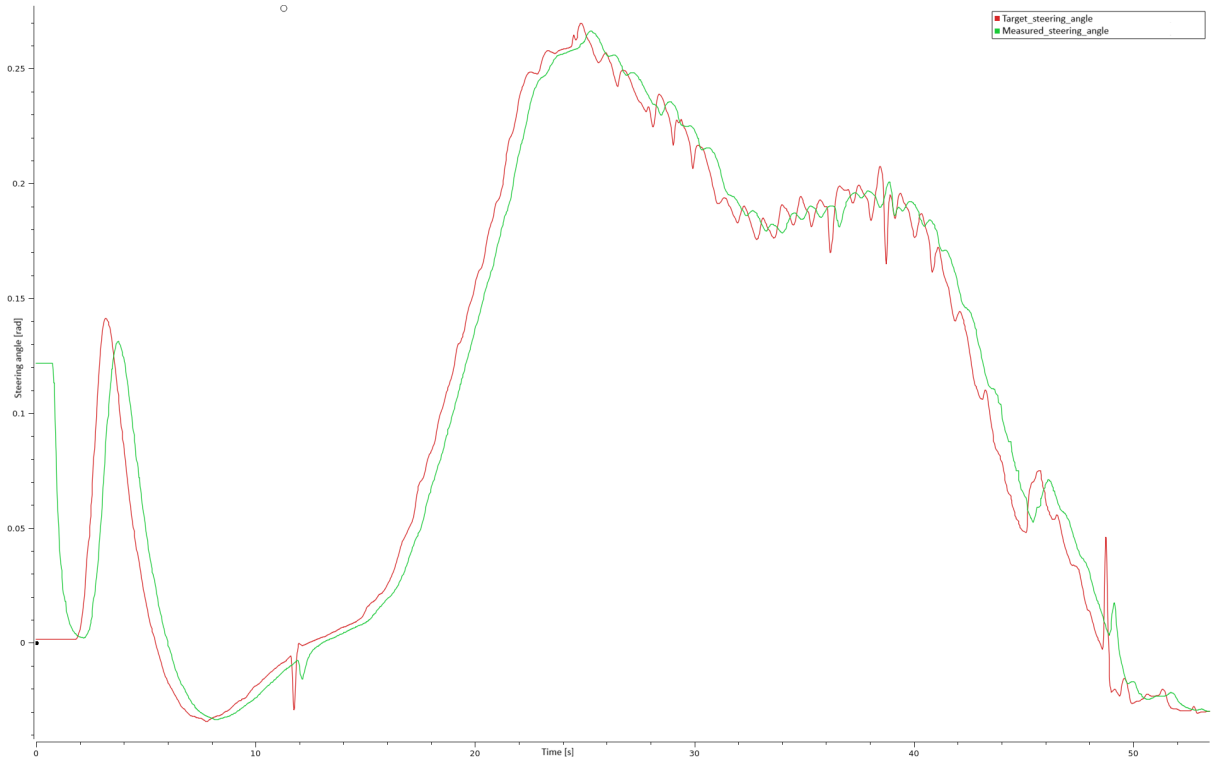

Figure 5.10: Lateral MPC — Robust profile: target and measured steering angle.

5.7. Comparison of Controller Variants

The comparison of the four controller variants — Baseline, Comfort, Tracking, and Robust — is performed on the basis of the metrics defined in Section 5.5, applied to the same set of test scenarios and aggregated across the runs of each scenario.

The qualitative comparison emerging from the results presented in Section 5.6 can be summarised as follows. The Baseline variant provides a balanced behaviour with reasonable tracking and stability, and is used as the reference against which the others are evaluated. The Comfort variant produces the smoothest actuation, with the lowest steering rate and the lowest acceleration jerk, at the price of a slightly larger lateral deviation, especially in the transitions between segments of different curvature. The Tracking variant produces the smallest lateral deviation and the fastest convergence to the reference, at the price of a more aggressive steering activity and of a higher steering rate. The Robust variant produces a behaviour that is less responsive than the Tracking variant but more stable in the presence of noise and delay; it tolerates better the unavoidable mismatches between the simulated and the actual actuator behaviour, and is therefore the natural candidate for deployment in less controlled conditions.

The time-series of the four profiles are reported in Section 5.6; the corresponding numerical metrics and statistical summaries will be added as the on-vehicle validation campaign is consolidated.

5.8. Influence of Physical Constraints and Disturbances

The physical constraints of the rickshaw — the limited steering range, the low operating speed, the actuator delay, and the sensitivity to disturbances — have been discussed in Chapter 4 as design constraints. Their influence on the experimental behaviour can be analysed along two complementary axes.

The influence of the steering range manifests itself in the tightest curved segments of the full-path scenario, where the steering command approaches the bound $\delta_{\max}$ and the safety supervision layer occasionally reduces the commanded speed in order to keep the lateral acceleration below its soft bound. The vehicle remains on the planned trajectory, but the longitudinal speed is locally lower than the planned one, which produces a measurable contribution to the speed-tracking error metric.

The influence of the actuator delay manifests itself in the steering signal during the transitions between segments of different curvature: the actual steering angle lags behind the commanded value by a quantity that is consistent with the modelled `input_delay` of the lateral MPC. The Comfort variant, which adopts the highest jerk and steering-input weights, produces the smoothest behaviour during these transitions, while the Tracking variant produces the fastest correction at the price of a more aggressive steering activity visible in Figure 5.9.

The influence of the disturbances has been characterised, in the integration tests, by the slip-detection mechanism implemented in the firmware. The mechanism, implemented in the firmware as described in Section 4.6.1, provides an automatic safety net that is available for the more demanding scenarios envisaged for the next phase of the project.

5.9. Practical Feasibility for Urban Transport

The translation of the technical results into a judgement on the practical feasibility of the rickshaw as an urban transport mode requires a clear delimitation of the operating domain in which the conclusions are valid. The operating domain considered in this thesis is the low-speed urban regime, with maximum speed of approximately 6 km/h, gentle to moderate curvatures, light to moderate payload, and shared use of cycling and road infrastructure. Within this domain, the experimental results available at the time of writing support three observations.

First, the integrated control chain — Autoware, control bridge, sensor bridges, embedded firmware, and physical actuators and sensors — is functional and reliable. The end-to-end integration tests have not revealed any structural failure of the architecture, and the qualitative behaviour observed on the physical vehicle is consistent with the design expectations.

Second, the lateral and longitudinal controllers, in their baseline configuration, produce a tracking behaviour that is qualitatively appropriate for the intended use: the deviations from the planned trajectory remain small on the straight and on the long curved segments, and the actuation commands remain within the bounds prescribed by the safety supervision

layer. The smoothness of the actuation, although not yet quantified, is sufficient to produce a behaviour that is comfortable for an unloaded vehicle.

Third, the architecture is amenable to the systematic exploration of the trade-off between tracking accuracy, smoothness, and robustness, through the four parameter profiles defined in Section 5.3 and characterised in Section 5.6. This is a relevant property for the deployment of the vehicle on real infrastructure, where the appropriate balance between these priorities depends on the specific context of operation.

Within the limits of the validation campaign, the results therefore support the qualitative claim that the rickshaw, as a low-speed urban platform for passenger and freight transport, is technically feasible. A definitive judgement on its practical relevance, however, requires the completion of the on-vehicle validation campaign currently in progress and, subsequently, the higher-speed and higher-payload tests envisaged for the next phase of the project.

5.10. Limitations and Critical Discussion

The validation campaign presented in this chapter is subject to several limitations that must be made explicit, both for the correct interpretation of the results and for the planning of future work.

The first limitation concerns the operating speed. The whole campaign is conducted at speeds bounded by approximately 6 km/h, in agreement with the safety policy adopted at the current stage of the project. This bound is conservative and reflects the priority assigned to safety in the early phase of the validation; it does not reflect a fundamental limitation of the architecture, which is in principle compatible with higher operating speeds. The extension of the campaign to higher speeds is part of the natural progression of the project but lies beyond the scope of the present thesis.

The second limitation concerns the payload. The tests are conducted on the empty vehicle, without the additional mass of a passenger or of a representative freight load. The influence of the payload on the dynamics of the rickshaw — through the change of the centre-of-gravity height and through the modification of the vertical loads on the wheels — is therefore not quantified. The architecture incorporates, in its safety supervision component, a soft bound on the lateral acceleration that is independent of the payload and that provides a conservative safety margin; the systematic characterisation of the controller behaviour as a function of the payload remains, however, open.

The third limitation concerns the completion status of the on-vehicle validation campaign. The systematic execution of the four controller variants on the full set of test scenarios is on-going at the time of writing, and the corresponding quantitative results, presented in Section 5.6, are therefore not yet exhaustive. The comparative analysis of the four controller variants in Section 5.7 will be consolidated as the campaign progresses.

The fourth limitation concerns the disturbance characterisation. The experimental tests have been conducted in a controlled outdoor area with a regular surface and in benign weather conditions. The behaviour of the controller on uneven surfaces, in adverse weather,

or in the presence of unexpected obstacles has not been characterised in the present campaign; the slip-detection mechanism described in Section 4.6.1 provides a first level of protection, but its activation has not been triggered by the present tests and its quantitative behaviour cannot be reported.

These limitations do not invalidate the conclusions of the present chapter, but they delimit their scope. They also constitute, in a constructive perspective, the basis of the future work outlined in Chapter 6.

6. Conclusion and Future Work

The present chapter closes the thesis by summarising the contributions of the work, by discussing the main findings and their practical implications, and by outlining the directions of future work that the project naturally suggests.

6.1. Summary of Contributions

This thesis has addressed the design, the implementation, and the evaluation of a control architecture for the semi-autonomous cycle rickshaw of the Chair of Traffic Engineering and Control of the Technical University of Munich. Within this scope, four principal contributions can be identified.

The first contribution is the formulation of an integrated control architecture for a non-tilting three-wheeled vehicle operating at low to moderate speed on cycling and road infrastructure. The architecture combines a high-level trajectory execution layer, a model predictive controller for the steering channel, a PID controller for the longitudinal channel, and an explicit safety supervision component, in a hierarchical structure that aligns with the Autoware standard interfaces. To the best of the author's knowledge, the literature does not contain a comparable integrated architecture for a vehicle of this class; the contribution is therefore not in the individual components, which are largely inherited from the literature reviewed in Chapter 2, but in their selection, adaptation, and combination for the specific platform considered.

The second contribution is the implementation of this architecture on the physical rickshaw, including the development of the dedicated control bridge between Autoware and the embedded actuation interface based on micro-ROS, and of the bridge nodes that expose the wheel-speed and steering-angle sensors as standard Autoware messages. The implementation is end-to-end functional and has been validated through interface inspection, timing measurement, and functional tests on the physical vehicle.

The third contribution is the systematic exploration of the parameter space of the lateral and longitudinal controllers through four design profiles — Baseline, Comfort, Tracking, and Robust — that correspond to clearly identifiable design priorities. This structured tuning campaign provides a basis for the comparative evaluation of the architecture and, more generally, a methodological template for the future tuning of the controller in different operating contexts.

The fourth contribution is the design and the partial integration of the proprioceptive sensing required for closed-loop operation. This includes the comparative selection of

the wheel-speed sensors (Infineon TLE4922) and of the steering-angle sensor (Infineon TLE5012B), the mechanical design of the 3D-printed supports, the development of the embedded firmware that implements the pulse counting and the slip-detection logic, and the verification of the end-to-end pipeline from the physical sensors to the Autoware standard messages.

6.2. Main Findings and Practical Implications

The work presented in this thesis suggests three findings of more general practical significance.

The first finding is that the integration of an existing autonomous-driving software stack, Autoware in the present case with a non-standard low-speed three-wheeled vehicle is feasible without major architectural modifications, provided that the vehicle interface is encapsulated in a dedicated bridge that mediates between the standard Autoware messages and the embedded actuation infrastructure. The finding is not self-evident: the standard target of Autoware is a passenger car operating at substantially higher speeds, and the low-speed regime considered in the thesis exposes characteristics such as the dominance of actuator delay that are usually not the primary concern of the stack.

The second finding concerns the value of the structured tuning campaign. The use of four explicit parameter profiles, each corresponding to a clearly identifiable design priority, has proved much more informative than a single optimised tuning would have been. It exposes the trade-off between tracking accuracy, smoothness, and robustness in a directly interpretable form, and it provides a practical basis for the deployment of the vehicle in different operating contexts: the Tracking profile is appropriate for tightly constrained paths in which the deviation must be minimised, the Comfort profile is appropriate for passenger transport, the Robust profile is appropriate for less controlled environments, and the Baseline profile provides a reasonable default.

The third finding is methodological. The combination of structured integration tests on the full software chain, of functional verifications on the embedded sensors and actuators, and of trajectory-tracking experiments on the physical vehicle constitutes a coherent validation workflow that, although standard in autonomous-driving practice, is rarely articulated in detail for a non-standard platform. The workflow developed in the thesis can be reused for the future iterations of the rickshaw and, more generally, for any low-speed automated platform integrated within the Autoware ecosystem.

6.3. Future Work

The natural directions for the continuation of the project, as suggested by the limitations identified in Section 5.10 and by the broader literature reviewed in Chapter 2, can be organised in three groups.

The first group concerns the remaining steps required to achieve full closed-loop autonomous operation of the rickshaw under Autoware. The sensor integration has been

completed and verified on the physical vehicle: both the wheel-speed and steering-angle sensors produce correct feedback during real motion. However, the full Autoware autonomy stack has not yet been exercised in a closed-loop moving scenario on the physical platform. To achieve this, several prerequisites remain. The Extended Kalman Filter localiser, which fuses wheel speed, steering angle, and IMU measurements, requires tuning of its covariance parameters on the real moving vehicle to produce reliable pose estimates. A GNSS receiver needs to be integrated and calibrated to provide the absolute position reference required by the Autoware localisation pipeline. Once localisation is reliable, the full Autoware planning stack, including mission planning, behaviour planning, and motion planning, can be activated and the vehicle can execute pre-planned trajectories fully autonomously under closed-loop control.

The second group concerns the completion of the validation campaign. The systematic execution of the four controller variants on the full set of test scenarios on the physical rickshaw is the most immediate task. The extension of the experimental campaign to a wider set of scenarios, including more demanding curvatures, longer urban routes, and structured disturbances such as controlled payload variations, will provide the quantitative basis for a more definitive characterisation of the architecture.

The third group concerns the extension of the operating domain. The current bound of approximately 6 km/h reflects the conservative safety policy adopted in the early phase of the project; its progressive relaxation, accompanied by a re-evaluation of the controller behaviour at higher speeds, is a natural next step. The characterisation of the influence of the payload on the controller behaviour, and the corresponding adaptation of the safety supervision layer, is a related extension.

Beyond these technical directions, the broader integration of the rickshaw within the urban mobility ecosystem, including its interaction with traffic, its use as a shared mobility resource, and its role in last-mile freight logistics, opens a research perspective that goes well beyond the scope of a control-oriented thesis, and one that the integrated, validated, and reproducible control architecture developed here is intended to make possible.

List of Figures

List of Tables

Bibliography

AKAR, MEHMET; ALI DINÇER DERE (2014). “A Switching Rollover Controller Coupled with Closed-Loop Adaptive Vehicle Parameter Identification”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.4, pp. 1579–1585. DOI: 10.1109/TITS.2014.2301721.

BEAL, CRAIG EARL; J. CHRISTIAN GERDES (2013). “Model Predictive Control for Vehicle Stabilization at the Limits of Handling”. In: *IEEE Transactions on Control Systems Technology* 21.4, pp. 1258–1269. DOI: 10.1109/TCST.2012.2200826.

BERNARDINI, D.; S. DI CAIRANO; A. BEMPORAD; H. E. TSENG (2009). “Drive-by-Wire Vehicle Stabilization and Yaw Regulation: A Hybrid Model Predictive Control Design”. In: *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*, pp. 7621–7626. DOI: 10.1109/CDC.2009.5400860.

DEMPSTER, ROWAN; MOHAMMAD AL-SHARMAN; DEREK RAYSIDE; WILLIAM MELEK (2023). “Real-Time Unified Trajectory Planning and Optimal Control for Urban Autonomous Driving Under Static and Dynamic Obstacle Constraints”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10139–10145. DOI: 10.1109/ICRA48891.2023.10160577.

DIVELBISS, A. W.; J. T. WEN (1997). “Trajectory Tracking Control of a Car-Trailer System”. In: *IEEE Transactions on Control Systems Technology* 5.3, pp. 269–278. DOI: 10.1109/87.572125.

EDELMANN, JOHANNES; MANFRED PLÖCHL; PETER LUGNER (2011). “Modelling and Analysis of the Dynamics of a Tilting Three-Wheeled Vehicle”. In: *Multibody System Dynamics* 26.4, pp. 469–487. DOI: 10.1007/s11044-011-9258-7.

INFINEON TECHNOLOGIES AG (Feb. 2025). *TLE4922-XAN-F Datasheet*. 2.0. Ordering code SP001106758. Infineon Technologies AG. Munich, Germany.

JIANMIN, DUAN; WANG RAN; YU YONGCHUAN (2010). “Research on Control Strategies of Steer-by-Wire System”. In: *2010 International Conference on Intelligent Computation Technology and Automation*. Vol. 2, pp. 1122–1125. DOI: 10.1109/ICICTA.2010.238.

KOLTER, J. ZICO; CHRISTIAN PLAGEMANN; DAVID T. JACKSON; ANDREW Y. NG; SEBASTIAN THRUN (2010). “A Probabilistic Approach to Mixed Open-Loop and Closed-Loop Control, with Application to Extreme Autonomous Driving”. In: *2010 IEEE International Conference on Robotics and Automation*, pp. 839–845. DOI: 10.1109/ROBOT.2010.5509562.

KONG, JASON; MARK PFEIFFER; GEORG SCHILDBACH; FRANCESCO BORRELLI (2015). “Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design”.

In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. Seoul, Korea, pp. 1094–1099. DOI: 10.1109/IVS.2015.7225830.

LI, LIANG; YISHI LU; RONGRONG WANG; JIE CHEN (2017). "A Three-Dimensional Dynamics Control Framework of Vehicle Lateral Stability and Rollover Prevention via Active Braking with MPC". In: *IEEE Transactions on Industrial Electronics* 64.4, pp. 3389–3401. DOI: 10.1109/TIE.2016.2583400.

LUU, DUC LICH; CIPRIAN LUPU (2019). "Dynamics Model and Design for Adaptive Cruise Control Vehicles". In: *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 12–17. DOI: 10.1109/CSCS.2019.00010.

MARTINS, FELIPE N.; WANDERLEY C. CELESTE; RICARDO CARELLI; MÁRIO SARCINELLI-FILHO; TEODIANO F. BASTOS-FILHO (2008). "An Adaptive Dynamic Controller for Autonomous Mobile Robot Trajectory Tracking". In: *Control Engineering Practice* 16.11, pp. 1354–1363. DOI: 10.1016/j.conengprac.2008.03.004.

ODENTHAL, DIRK; TILMAN BÜNTE; JÜRGEN ACKERMANN (1999). "Nonlinear Steering and Braking Control for Vehicle Rollover Avoidance". In: *1999 European Control Conference (ECC)*, pp. 598–603. DOI: 10.23919/ECC.1999.7099370.

PANDEY, AYUSH; SIDDHARTH JHA; DEBASHISH CHAKRAVARTY (2017). "Modeling and Control of an Autonomous Three Wheeled Mobile Robot with Front Steer". In: *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 136–142. DOI: 10.1109/IRC.2017.67.

PERDANA, MUHAMMAD ARJUNA PUTRA; ALEXANDER C. BUDIMAN; RINA RISTIANA, et al. (2024). "Control Strategies for Steer-By-Wire Systems: An Overview". In: *Technologies* 13.1. DOI: 10.3390/technologies13010006.

RAHMAN, MD HAFIZUR; MUHAMMAD MAJID GULZAR; TANSU SILA HAQUE; SALMAN HABIB; ADNAN SHAKOOR; ALI FAISAL MURTAZA (2025). "Trajectory Planning and Tracking Control in Autonomous Driving System: Leveraging Machine Learning and Advanced Control Algorithms". In: *Engineering Science and Technology, an International Journal* 64, p. 101950. DOI: 10.1016/j.jestch.2025.101950.

Research of Automotive Steer-by-Wire Control Based on Integral Partition PID Control (n.d.). Accessed: 2025-12-04. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5402773.

RODRÍGUEZ LICEA, MARTÍN ANTONIO; EDGAR ARMANDO VAZQUEZ RODRÍGUEZ; FRANCISCO JAVIER PEREZ PINAL; JUAN PRADO OLIVARES (2018). "The Rollover Risk in Delta Tricycles: A New Rollover Index and Its Robust Mitigation by Rear Differential Braking". In: *Mathematical Problems in Engineering*, pp. 1–14. DOI: 10.1155/2018/4972419.

"Stability of Three-Wheeled Vehicles with and without Control System" (2013). In: *International Journal of Vehicle Structures & Systems* 3.1.