FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Control of a Self-Balancing Motorcycle

Raúl Miguel Domingues Moreira da Silva

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Luís Tiago de Freixo Ramos Paiva

February 28, 2026

Resumo

Esta tese investiga a modelação, simulação e implementação experimental de estratégias de controlo para a estabilização de uma mota equipada com uma roda de inércia. É desenvolvido um modelo dinâmico não linear do sistema com base em equações do movimento derivadas a partir de princípios fundamentais, que é expresso em representação de espaço de estados. O modelo é então linearizado em torno do ponto de equilíbrio na posição vertical, o que permite aplicação de técnicas de controlo linear.

São desenvolvidas e avaliadas várias estratégias de controlo, nomeadamente Proporcional – Derivativo(PD),Proporcional – Integral – Derivativo (PID), Regulador Linear Quadrático (LQR) e Controlo por Modo Deslizante (SMC). Os controladores PD e PID são inicialmente ajustados usando o método de Ziegler–Nichols e posteriormente através de afinação manual. O seu desempenho é avaliado com base em métricas temporais clássicas, tais como o tempo de atraso, o tempo de subida, a percentagem máxima de overshoot e o tempo de repouso. A análise de polos e zeros é utilizada para estudar a estabilidade e as características dinâmicas de cada configuração em malha fechada.

De forma a ultrapassar limitações observadas nos controladores clássicos, nomeadamente a saturação dos actuadores e a sensibilidade a desvios nos sensores, é desenvolvido um controlador LQR com base no modelo linearizado em espaço de estados. São exploradas diferentes estratégias de ajuste dos controladores, incluindo a regra de Bryson, matriz de identidade e ajuste manual das matrizes de penalização. Os resultados obtidos são validados através de simulações em Simulink e Simscape, bem como por ensaios experimentais realizados na mota.

Por fim, é implementada e simulada uma estratégia de Controlo por Modo Deslizante com o objectivo de aumentar a robustez face a perturbações externas e incertezas do modelo. São analisadas tanto a abordagem clássica como variantes de modo deslizante.

Abstract

This thesis investigates the modelling, simulation, and experimental implementation of control strategies for the stabilization of a self-balancing motorcycle equipped with an inertial wheel. A nonlinear dynamic model of the system is derived using first-principles equations of motion and subsequently expressed in state-space form. The model is linearized around the upright equilibrium to enable the application of linear control techniques.

Several control strategies are designed and evaluated, including Proportional–Derivative (PD), Proportional–Integral–Derivative (PID), Linear Quadratic Regulator (LQR), and Sliding Mode Control (SMC). The PD and PID controllers are initially tuned using the Ziegler–Nichols method and through manual tuning. Their performance is evaluated using standard time-domain metrics such as delay time, rise time, maximum percent overshoot, and settling time. Pole–zero analysis is used to study the stability and dynamic characteristics of each closed-loop configuration.

To overcome limitations observed in classical controllers, particularly actuator saturation and sensitivity to sensor bias, an LQR controller is developed based on the linearized state-space model. Multiple tuning strategies are explored, including Bryson's rule, identity weighting, and manual adjustment of the weighting matrices. These results are validated through both Simulink and Simscape simulations, as well as physical experiments conducted on the motorcycle platform.

Finally, a Sliding Mode Control strategy is implemented and simulated to address robustness against disturbances and model uncertainties. Both classical and quasi-sliding mode approaches are analyzed.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) offer a global framework to address the most pressing societal, environmental, and technological challenges of our time [17]. This thesis aligns with several of these goals by contributing to research and innovation in the field of autonomous vehicles, promoting education and technical skills, and enabling sustainable urban mobility systems.

Figure 1: UN Sustainable Development Goals [17]

The specific Sustainable Development Goals supported by this work are:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable.

SDG	Target	Contribution	Performance Indicators and Metrics
4	4.4	Development of advanced control systems (PD, LQR, SMC) contributes to the acquisition of engineering and technical skills in control, robotics and autonomous systems.	Acquisition and application of technical competencies by students and researchers
9	9.5	Simulation and prototyping of an autonomous self-balancing vehicle contributes to scientific research and promotes innovation in mobility systems.	Integration of simulation and hardware validation in control design; research output and dissemination
11	11.2	The proposed system promotes compact, agile, and energy-efficient mobility solutions that could be scaled to real-world urban transport applications.	Potential applicability of two-wheeled autonomous systems in smart cities and urban mobility

Table 1: Contribution of this dissertation to the UN Sustainable Development Goals.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Luís Paiva, for his continued support, guidance, and availability throughout the development of this thesis.

I also thank the LSCOE (Laboratory of Systems, Control, Optimization, and Estimation) for providing access to the laboratory space and hardware resources necessary for the implementation of this project.

To my family, whose encouragement and unconditional support have been essential every step of the way, I am deeply grateful. I also extend my appreciation to my friends for their ongoing support and motivation throughout this academic journey.

To all of you, thank you.

Raúl Silva

"Our greatest glory is not in never falling, but in rising every time we fall"

Confucius

Contents

List of Acronyms and Symbols

ARE	Algebraic Riccati Equation
DC	Direct Current
FEUP	Faculdade de Engenharia da Universidade do Porto
HIL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
IoT	Internet of Things
LQR	Linear Quadratic Regulator
LQG	Linear Quadratic Gaussian
LSCOE	Laboratório de Sistemas, Controlo, Otimização e Estimação
MATLAB	Matrix Laboratory
MPC	Model Predictive Control
PD	Proportional-Derivative
PID	Proportional, Integral, and Derivative
PWM	Pulse Width Modulation
SIMULINK	Simulation and Linkage
SMC	Sliding Mode Control
LTI	Linear Time-Invariant
MIMO	Multi-Input Multi-Output
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
ARM	Advanced RISC Machine
RPM	Revolutions Per Minute
QSMC	Quasi-Sliding Mode Control
SDG	Sustainable Development Goal
UN	United Nations
θ	Lean angle (rad)
$\dot{\theta}$	Angular velocity of the frame (rad/s)
$\ddot{\theta}$	Angular acceleration of the frame (rad/s^2)
ω	Angular velocity of the inertial wheel (rad/s)
x	State vector
$\dot{x}$	Time derivative of the state vector
u	Control input (torque)
r	Reference signal
K	Feedback gain matrix (LQR)
Q	State penalty matrix (LQR)
R	Input penalty matrix (LQR)
A	State matrix
B	Input matrix

C	Output matrix
D	Feedthrough matrix
g	Gravitational acceleration (9.807 m/s^2)
m	Total mass of the motorcycle (kg)
m_r	Mass of the pendulum rod (kg)
m_w	Mass of the inertial wheel (kg)
r	Radius of the pendulum rod's cross-section (m)
R	Radius of the wheel (m)
l	Length of the pendulum (m)
l_{AB}	Distance from pivot to rod center of mass (m)
l_{AC}	Distance from pivot to wheel center of mass (m)
l_{AD}	Total length of the pendulum rod (m)
I_{wC}	Inertia of the wheel about its own center (kg·m^2)
I_{wA}	Inertia of the wheel about the pivot axis (kg·m^2)
I_{rB}	Inertia of the rod about its center of mass (kg·m^2)
I_{rA}	Inertia of the rod about the pivot axis (kg·m^2)
c	Sliding surface gain (SMC)
k	Proportional gain in sliding control
η	Reaching law gain in SMC
s	Sliding surface
$\dot{s}$	Derivative of the sliding surface
t	Time (s)
t_d	Delay time (s)
t_r	Rise time (s)
t_s	Settling time (s)
M_p	Maximum percent overshoot (%)

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is conducted as part of ongoing research at LSCOE. The project focuses on the autonomous driving of a self-balancing motorcycle, researching many aspects such as perception, estimation, control, and computational learning. This chapter provides an overview of the problem studied, describing the background, motivation, and objective of this dissertation. The chapter ends with an explanation of the dissertation's structure.

1.1 Background

The field of autonomous vehicle technology has advanced significantly in recent years, driven by the imperative for safer and more efficient transportation systems. Autonomous vehicles require sophisticated control algorithms and comprehensive sensor systems for operation. Although research predominantly emphasizes four-wheeled vehicles due to their inherent stability, two-wheeled systems present distinctive challenges and opportunities warranting investigation.

Two-wheeled vehicles, such as motorcycles, offer advantages in agility and spatial efficiency, making them suitable for urban deployment and congested environments. However, their inherent instability necessitates sophisticated control systems to maintain equilibrium. Achieving stable balance—particularly under dynamic conditions with varying speeds and external disturbances—requires addressing complex nonlinear dynamics and ensuring robust stability margins.

Research in self-balancing two-wheeled systems has explored several approaches, including gyroscopic stabilization and inertial control mechanisms. Nonlinear control techniques are a promising solution for dealing with the complex dynamics of these systems. In addition, integrating balance control with trajectory tracking opens new possibilities for autonomous navigation, particularly in racing or urban mobility scenarios.

1.2 Motivation

Autonomous vehicle technology has driven demand for innovative mobility solutions in constrained environments. Two-wheeled systems offer distinct advantages in agility, compactness,

and energy efficiency compared to conventional four-wheeled platforms, though they remain less studied. These characteristics make them particularly suitable for urban transportation and emergency response in congested areas.

Despite their potential, two-wheeled autonomous vehicles face considerable challenges due to their inherent instability and the need for precise control systems. Developing a robust framework for self-balancing and trajectory tracking is not only an interesting engineering challenge but also an opportunity to advance the state of the art in nonlinear control and robotics.

This thesis is motivated by the desire to address these challenges through innovative control strategies. Working with a self-balancing motorcycle, this research explores the intersection of theoretical control methodologies and practical implementation, contributing to the field of autonomous vehicle systems.

Furthermore, the knowledge gained from this work has the potential to influence real-world applications, such as autonomous delivery robots and personal mobility devices, promoting safer and more efficient transportation solutions.

1.3 Objective

The primary objective of this dissertation was to model, design, and compare different control strategies for a self-balancing scaled motorcycle using an inertial wheel as the balancing mechanism.

The study focuses on the development and analysis of classical and robust control approaches — namely, Proportional-Derivative (PD) control, Proportional-Integral-Derivative (PID) control, Linear Quadratic Regulator (LQR), and Sliding Mode Control (SMC)—evaluating their effectiveness through simulation and experimental testing.

In addition to the analysis of the controllers, the project verifies the performance of the developed models in realistic simulation environments, including Simscape, and through hardware testing using a prototype based on the Arduino Engineering Kit.

1.4 Dissertation Structure

This dissertation is organized into five chapters, each addressing a specific aspect of the development, control, and evaluation of a self-balancing motorcycle using an inertial wheel.

Chapter 1 – Introduction

This chapter introduces the problem of self-balancing two-wheeled vehicles and motivates the use of inertial wheel stabilization. The objectives of the dissertation are presented, together with the main contributions and an overview of the document structure.

Chapter 2 – Self-Balancing Autonomous Vehicles

This chapter provides the theoretical background required to understand self-balancing systems and control strategies. It reviews the state of the art in self-balancing mechanisms, control systems, and feedback strategies, with a focused bibliographic review of PID, LQR, and Sliding

Mode Control techniques. A comparative summary of these control approaches is included to motivate the controller selection adopted in this work.

Chapter 3 – Nonlinear Dynamic Modeling and Linearization of a Self-Balancing Motorcycle with Inertial Wheel

This chapter describes the physical and mathematical modeling of the self-balancing motorcycle. The mechanical structure is presented, followed by the derivation of the equations of motion using rigid-body dynamics and inertia theorems. Both nonlinear and linearized models are developed, forming the basis for controller design and simulation.

Chapter 4 – Design and Implementation of Control Techniques

This chapter details the design and implementation of the control strategies considered in this dissertation. PID/PD control, Linear Quadratic Regulator (LQR), and Sliding Mode Control (SMC) are presented, including tuning methodologies, controller architecture, and implementation in Simulink and Simscape. Time-domain performance metrics, disturbance rejection capability, and robustness are analyzed.

Chapter 5 – Results and Conclusions

This chapter compares the simulation and experimental results obtained for the different control strategies. The chapter concludes with the main findings of the dissertation and directions for future work.

Chapter 2

Self-Balancing Autonomous Vehicles

The development of autonomous vehicles, especially self-balancing motorcycles, represents a challenging field in robotics. These systems combine complicated dynamics and accurate control mechanisms to achieve stability and autonomy in dynamic environments.

This chapter presents a review of the state of the art and related work, focusing on areas like self-balancing mechanisms, linear and nonlinear control techniques, and autonomous vehicle racing. Additionally, the role of simulation tools in the advancement of research and development was discussed.

2.1 Introduction to Autonomous Two-Wheeled Vehicles

Autonomous vehicles represent an emerging technology in transportation and robotics, offering enhanced safety, efficiency, and operational flexibility [3]. These systems integrate sensors, control algorithms, and computational resources to enable autonomous navigation in complex environments [18].

Although research and development have concentrated on four-wheeled platforms due to their inherent stability and broad applicability, two-wheeled autonomous systems offer distinct advantages while presenting unique control challenges.

2.2 Control Techniques for Self-balancing - State of the Art

This section summarizes previous articles, studies and dissertations cited throughout the dissertation, presenting relevant contributions to the design, implementation, and simulation of autonomous balancing systems. These references explore various control strategies — including PID, LQR, MPC, and SMC — as well as the use of inertial sensors such as accelerometers and gyroscopes.

2.2.1 Control theory

The theoretical foundation for this project was based on fundamental and advanced concepts of control engineering, including both classical and modern approaches. The book Control Engineering by Bars, Hetthéssy, and Bányász [4] focuses mainly on single variable, linear, constant parameter systems, exploring the basic concepts of a control process, open- and closed-loop control, and stability analysis. It also includes a detailed coverage of PID control and advanced topics in control design.

The book Modern Control Engineering by Katsuhiko Ogata [20], served as a central reference by exploring the principles of control theory, covering essential topics such as state-space modeling, stability, and feedback control. Modern Control, specifically, is based on time-domain analysis and the concept of state, having developed since around 1960, making it suitable for the analysis and design of complex systems.

The books Sliding mode control: theory and applications by C. Edwards [12] and Applications of sliding mode control by Nabil Derbel [10] explore Sliding mode Control in depth, describing the basic principles and several applications of the control technique.

2.2.2 PID Control in Self-Balancing Systems

PID control remains one of the most commonly adopted strategies for self-balancing vehicles due to its simplicity, low computational cost, and ease of implementation. Several works in the literature investigate its applicability to inverted-pendulum-based platforms.

Hla Myo Tun et al. [18] studied the stabilization of a two-wheeled mobile robot modeled as an inverted pendulum. Their work focused on system modeling, sensor fusion using an IMU with a complementary filter, and a comparison between PID and LQR control strategies. Although LQR demonstrated superior performance in simulation, the authors successfully implemented PID control on hardware, highlighting its practicality and robustness in real systems.

Vasilevski et al. [27] presented the design of a self-balancing motorcycle prototype stabilized via an inertial wheel. The system was modeled as an inverted pendulum, and a PD controller was implemented using low-cost hardware and IMU feedback. This work demonstrated that PID-based control strategies can be effectively applied to self-balancing motorcycles using accessible components and straightforward control architectures.

Siller-Alcalá et al. [22] investigated the stabilization of an unmanned bicycle at zero speed using an inertial wheel. Their study compared PID and state-feedback control approaches, concluding that while PID control can stabilize the system in simulation, it presents practical limitations in real-time implementation due to sensitivity to modeling inaccuracies. The work highlights the challenges of applying classical PID control to inherently unstable systems.

2.2.3 Optimal Control (LQR) in Unstable Balancing Systems

Optimal control techniques, particularly the Linear Quadratic Regulator (LQR), are widely studied for stabilizing unstable and under-actuated systems such as inverted pendulums and self-balancing

vehicles.

Tang Teng Fong et al. [23] applied LQR control to a rotary inverted pendulum, emphasizing the influence of weighting matrix selection on closed-loop performance. Their work demonstrated the effectiveness of LQR for stabilizing nonlinear balancing systems after linearization around the upright equilibrium.

Mohammed et al. [9] extended LQR design to a multi-input, multi-output (MIMO) balancing platform, the Ball-On-Sphere system. Their study confirmed LQR as a suitable control strategy for complex under-actuated balancing systems once linearized dynamics are obtained.

Kanjanawanishkul [16] compared LQR with linear and nonlinear Model Predictive Control (MPC) for a self-balancing bicycle robot equipped with an inertial wheel. The study highlighted the trade-off between LQR's computational efficiency and MPC's ability to explicitly handle system constraints, positioning LQR as an effective baseline control strategy.

2.2.4 Sliding Mode Control For Robust Balancing

Sliding Mode Control (SMC) has been widely explored for self-balancing systems due to its inherent robustness to parameter uncertainty and external disturbances.

Tran et al. [24] investigated the application of SMC to an inverted pendulum system, emphasizing robustness under real-world uncertainties. The authors employed a genetic algorithm to tune the SMC parameters, demonstrating that robustness-oriented optimization is essential for successful physical implementation. This work highlights SMC as a strong candidate for balancing systems where model inaccuracies are significant.

2.2.5 Comparative Summary of Control Techniques

PID and PD controllers offer simplicity and ease of implementation, making them attractive for rapid prototyping and initial experimental validation. However, their performance is highly dependent on tuning and they exhibit limited robustness in the presence of disturbances and model uncertainties.

The LQR controller provides a systematic design methodology based on optimal control theory and achieves improved transient performance and robustness when applied to the linearized system. Its reliance on an accurate linear model, however, limits its effectiveness under strong nonlinearities.

Sliding Mode Control offers the highest robustness among the considered approaches, particularly in the presence of disturbances and modeling uncertainties. This comes at the cost of increased design complexity and practical issues such as chattering, which must be addressed through techniques such as boundary layers or quasi-sliding modes.

Based on these characteristics, all three approaches are investigated experimentally in the following chapters to assess their relative performance, practical feasibility, and suitability for implementation on the self-balancing motorcycle platform.

2.2.6 Autonomous Racing

Autonomous vehicle racing represents a highly demanding domain within robotics and control engineering. Racing environments push vehicles to operate at their dynamic limits, requiring precision, speed, and real-time decision-making.

Unlike traditional autonomous navigation, which prioritizes safety and efficiency, racing environments demand aggressive maneuvers, high-speed trajectory optimization, and adaptive control strategies. The survey by Johannes Betz et al. (2022) [5] provides an overview of the field.

Autonomous racing competitions, such as the Indy Autonomous Challenge and Roborace, have gained prominence as real-world testing platforms. These events challenge teams to develop advanced control algorithms capable of handling high-speed turns, obstacle avoidance, and real-time decision-making in dynamic multi-vehicle scenarios.

While most autonomous racing research focuses on four-wheeled vehicles, many of the control strategies, such as optimal trajectory planning and predictive control, are applicable to two-wheeled systems.

The primary challenges in autonomous motorcycle racing include managing the balance-velocity trade-off, optimizing trajectory tracking under extreme conditions, and implementing robust control algorithms that can adapt to changing track conditions.

2.3 Fundamental Principles and Control Theory

Having reviewed existing approaches, this section introduces the fundamental principles underlying self-balancing two-wheeled systems, contrasting passive gyroscopic stabilization with active inertial-based balancing mechanisms.

Core control-system concepts are then reviewed, including open- and closed-loop operation, feedback, and common system classifications. These foundations establish the theoretical framework required for the modeling, controller design, and experimental validation presented in subsequent chapters.

2.3.1 Self-Balancing Mechanisms

Self-balancing mechanisms enable two-wheeled vehicles to maintain stable equilibrium by counteracting gravitational destabilization. These principles involve the use of gyroscopic effects [14] and inertial systems, which help in maintaining equilibrium by counteracting the instability in two-wheeled designs.

Gyroscopic effects arise from angular momentum generated by rotating wheels. A spinning wheel exhibits resistance to changes in orientation, thereby providing passive stabilization proportional to rotational velocity. Consequently, motorcycles and bicycles exhibit enhanced stability at higher speeds, but experience reduced gyroscopic stabilization at low speeds, increasing vulnerability to external disturbances.

Inertial systems, on the other hand, maintain balance by using sensors, controllers, and actuators. Sensors like gyroscopes, accelerometers, and inertial measurement units (IMUs) continuously monitor the vehicle's tilt and angular velocity.

A control system processes this data and determines the corrective action needed to maintain balance [18]. This corrective action is executed by actuators, usually in the form of an inertial wheel. By accelerating or decelerating the wheel, the system creates a torque that counters the imbalance.

Below is a table comparing Gyroscopic effects and Inertial Systems.

Table 2.1: Comparison of Gyroscopic Effects and Inertial Systems.

Aspect	**Gyroscopic Effects**	**Inertial Systems**
Mechanism	Relies on angular momentum of spinning wheels or gyroscopes	Applies torque using sensors, controllers, and actuators
Stability	More effective at high speeds	Effective at low speeds and when stationary
Complexity	Passive system, relatively simple	Active system, requires advanced sensors and control algorithms
Use Cases	High-speed motorcycles and bicycles	Self-balancing robots, low-speed vehicles

2.3.2 Control Systems

A control system is a mechanism with the objective of controlling the output of another system. It usually involves observing a system's behavior, comparing it to a reference value and making adjustments in order to achieve the desired output. Below is a table with the key components of a control system.

Table 2.2: Key Components of a Control System.

Component	**Description**
Input	The desired value or set point the system aims to achieve.
Process/System	The physical or logical system being controlled.
Controller	A device or algorithm that determines the necessary actions to regulate the system.
Output	The actual result or behavior of the system.
Feedback (optional)	Measures the output and sends it back to the controller for adjustment.

2.3.3 Types of Control Systems

Control systems are classified along multiple dimensions: open-loop versus closed-loop operation, linear versus nonlinear dynamics, and continuous versus discrete-time implementation.

An open-loop system operates based on predefined inputs without relying on feedback to adjust the output. It does not monitor the real-time behavior of the system. This type of system has a

very simple design and implementation, however, it is less accurate when compared to non-linear systems and cannot deal with external disturbances or system variations [20].

One example of a linear system is a toaster, which heats for a predefined duration regardless of the current state of the toast. These systems are easy to manufacture and cheap, as well as having faster operation since feedback loop is not present. They also have clear disadvantages, like not being able to self-correct errors and their limited adaptability to changing conditions.

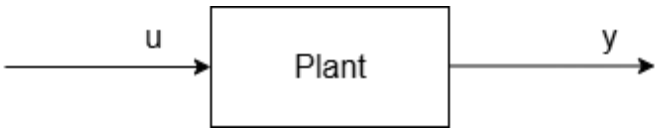


Figure 2.1: Open-loop control structure: The control input $u(t)$ is applied directly to the plant, producing the output $y(t)$ without feedback or error correction.

A closed-loop system, also known as feedback control system, constantly monitors its output and makes changes to the input in order to minimize any deviation from the desired result. [4] It includes a feedback mechanism, provides better accuracy and stability and can handle disturbances effectively.

An example of a closed-loop system is the cruise control in cars, which maintains a constant speed by adjusting throttle based on road conditions. These systems have better accuracy because they can automatically correct any errors and can adapt to external disturbances, however, they have a more difficult design, higher cost and if not properly tuned can show instability [20].

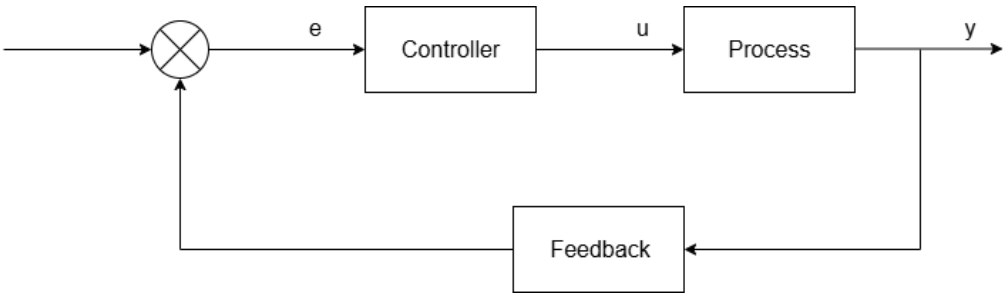


Figure 2.2: Closed-loop control structure: The reference input is compared with the measured output to generate the error signal $e(t)$, which is processed by the controller to produce the control action $u(t)$. The plant output $y(t)$ is fed back to continuously correct deviations and reject disturbances.

2.3.4 Feedback in Control Systems

Feedback in control systems is the process of using the output of a system to influence the input in order to achieve a desired behavior. It is an essential concept that helps maintain stability and adapt to changes or disturbances.

The output is continuously measured and compared to the reference input. The difference between the reference input and the output, the error signal, is used to adjust the input of the system [20]. Below is a table with the advantages and disadvantages of feedback:

Table 2.3: Advantages and Disadvantages of Feedback in Control Systems.

Advantages	**Disadvantages**
Reduces errors by continuously adjusting the behavior of the system	Increases system complexity due to added feedback loops
Increases stability by counteracting deviations from the desired output	Poorly designed feedback can lead to oscillations or instability
Improves robustness, making the system tolerant to parameter variations and external disturbances	May slow down the system response due to the correction process
Increases accuracy, response time, and overall performance	Can amplify noise in the system if not properly filtered

2.3.5 Subtypes of Closed-Loop Systems

The subtypes of closed-loop systems consider different ways systems can be classified in according to their behavior, structure and design.

A closed-loop system can be time-variant or time-invariant. Time-variant systems are more complicated to analyze and control because they need adaptive controllers or time-dependent models. The parameters change over time and system characteristics evolve, while time-invariant systems are easier to design since the equations controlling the system do not change and the parameters remain constant over time [20].

Systems can also be linear or nonlinear. In linear systems, the output is proportional to the input, and they are controlled by linear differential equations, which are easier to solve and predict. Linear controllers like PID (Proportional – Integral – Derivative) are very effective. An example of these type of systems is an amplifier with a proportional gain.

In nonlinear systems, the relation between input and output is not proportional, which leads to complicated dynamics. They are controlled by nonlinear equations which require techniques like Lyapunov stability analysis [1]. Although more realistic for real-world systems, they are harder to design and implement.

Systems can be identified as continuous or discrete. Continuous systems operate with inputs and outputs that change over time and are used in systems that require real-time control [4], like fluid level control in tanks. Discrete systems operate using sampled data at specific intervals and are usually implemented in digital controllers or computers, like a micro-controller-based motor speed controller.

There are adaptive and non-adaptive systems. Adaptive systems can adjust their parameters automatically in response to changing conditions or system dynamics and usually include mechanisms to improve performance over time, while non-adaptive systems have fixed parameters that do not change during operation. They are simpler to design but less efficient in changing conditions.

2.4 Simulation and Testing Platforms

Simulation plays an important role in the development and validation of self-balancing autonomous motorcycles, providing a safe and controlled environment for testing control algorithms before real-world implementation.

Given the nonlinear unstable nature of two-wheeled vehicles, simulation allows for early-stage controller validation, minimizing risks before physical deployment. Moreover, it enables tuning of balance and trajectory tracking algorithms under many conditions, testing under external disturbances such as wind or uneven terrain, and iterative improvements without hardware constraints.

MATLAB Simulink is the primary simulation platform chosen for this research due to its powerful multibody modeling, state-space control system design, and hardware-in-the-loop (HIL) integration capabilities. Simulink provides an environment where system dynamics, sensors, and controllers can be integrated efficiently.

Additionally, state-space models are used to enable advanced control strategies such as the Linear Quadratic Regulator (LQR). The simulation structure is designed to replicate real-world conditions, allowing a smooth transition from simulation to physical testing.

Different control strategies were tested in simulation, including PD control for its simplicity and ease of implementation, LQR for its optimal balance between performance and control effort, and Sliding Mode Control (SMC) for its robustness to disturbances and model uncertainties. By adjusting parameters and observing how the system responded, the strengths and weaknesses of each approach were compared, helping to identify the best option for future real-world use.

While simulation is a powerful tool, real-world validation is essential. The prototype used in this study is based on an Arduino-controlled self-balancing motorcycle, allowing for hardware testing of the simulated controllers.

The results gained from MATLAB Simulink simulations are used to improve the real-world implementation. Parameter tuning is performed to achieve better real-world performance, such as adjusting gains for faster response times.

Additionally, simulation results help compensate for sensor noise and delays, which are ignored in simulation but important in hardware. Ensuring integration between the control system and actuators, such as inertial wheels and steering mechanisms, is also very important.

Simulink-based simulations significantly reduce development time and risks, offering a structured approach for designing and testing autonomous control strategies before real-world deployment.

2.5 System Design

Practical tests were performed on a self-balancing motorcycle engineering kit by Arduino Education [21]. The motorcycle project involves a two-wheeled robot designed to maintain balance and navigate by utilizing a rotating disc, referred to as the inertial wheel, to stabilize the motorcycle when it begins to tilt.

2.5.1 Arduino Nano 33 IoT

The system is powered by an Arduino Nano 33 IoT (Internet of Things):

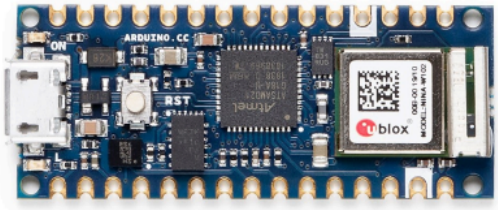


Figure 2.3: Arduino Nano 33 IoT [2]

The Arduino Nano 33 IoT is a microcontroller board built on the ARM Cortex-M0+ SAMD21 processor, operating at 3.3 V logic, with integrated Wi-Fi and Bluetooth Low Energy connectivity via the u-blox NINA-W102 module. It also includes a 6-axis inertial measurement unit, making it a good option for embedded systems that require wireless communication and motion sensing.

2.5.2 Arduino Nano Motor Carrier

It is paired with the Arduino Nano Motor Carrier, to control two DC motors:

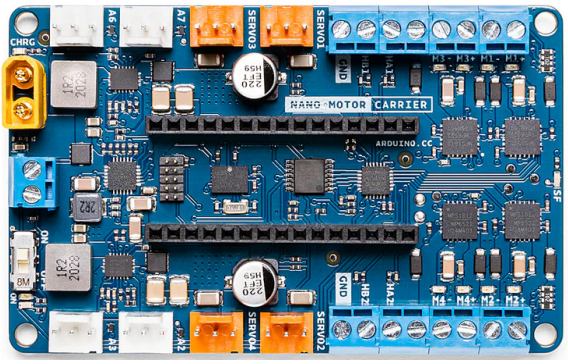


Figure 2.4: Arduino Nano Motor Carrier [19]

The Arduino Nano Motor Carrier is an expansion board designed to extend the capabilities of the Arduino Nano micro controllers. It integrates four motor driver channels that can control DC motors or two stepper motors, as well as an additional channel for a servo motor. The board includes a power management system that allows motors and the micro controller to be powered from a single source, supporting input voltages up to 12 V.

It also provides extra connectors for I²C, SPI, analog, and digital signals, along with onboard protection features such as overcurrent and thermal shutdown. The motor carrier uses H-bridges to drive the DC motors.

An H-bridge is an electronic circuit that enables control of the direction of current applied to a load. In this project, the H-bridges of the Arduino Nano Motor Carrier are used to supply 12 V to

the motors, while receiving 3.3 V control signals from the microcontroller. The H-bridge consists of four transistors arranged in a configuration that allows the current flow to be reversed, allowing the control of both the direction of rotation and the angular speed of the DC motor.

These transistors operate as four switches, activated in pairs, so that the current path depends on which pair is conducting, determining the rotation direction of the DC motor shaft [13].

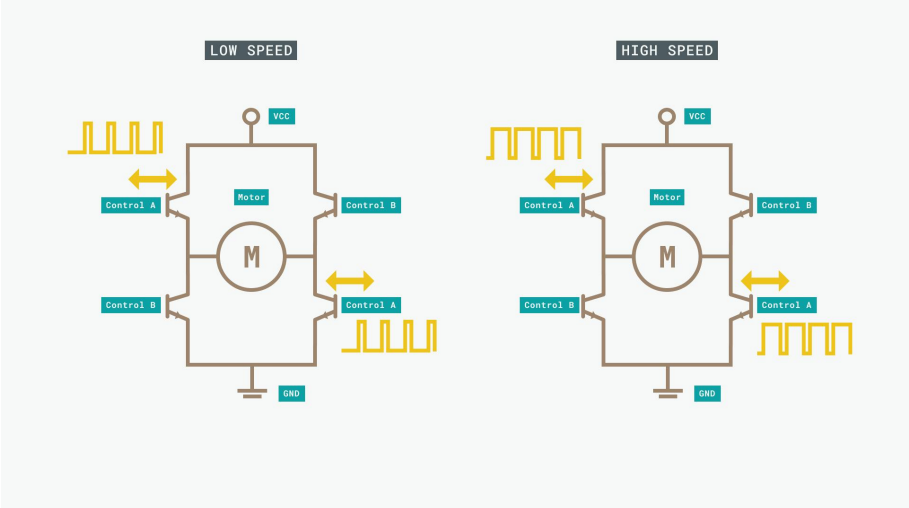


Figure 2.5: H-Bridge current flow [13]

To regulate DC motor speed, Pulse Width Modulation (PWM) is applied. PWM is a digital modulation technique in which the width of the pulses in a periodic signal is varied while keeping the frequency constant.

The fraction of the period during which the signal is active is defined as the duty cycle, which ranges from 0% to 100%. By adjusting the duty cycle, the effective voltage applied to the DC motor can be controlled, regulating its speed [13].

2.5.3 Inertial Measurement Unit (IMU)

An Inertial Measurement Unit (IMU) is an electronic device that measures linear acceleration, angular velocity, and, in some cases, the surrounding magnetic field. IMUs integrate multiple sensors into a single package, typically combining accelerometers, gyroscopes, and magnetometers.

The accelerometer measures variations in linear acceleration along three orthogonal axes (x, y, and z). When subjected to stress or vibration, its sensing element generates voltage changes that are interpreted as motion data. The gyroscope provides information on angular velocity, also along the x, y, and z axes, enabling the estimation of orientation and rotational dynamics.

The magnetometer measures magnetic field intensity and direction, and in IMU applications it is commonly used to detect the Earth's magnetic field, allowing orientation with respect to the magnetic north, similar to a digital compass [13].

In this project, the IMU is used to measure the vertical orientation of the self-balancing motorcycle and to detect deviations that indicate loss of balance. Specifically, the carrier board integrates

the BNO055, a 9-axis absolute orientation sensor (accelerometer, gyroscope, and magnetometer), which communicates with the microcontroller via the I^2C interface.

2.5.4 Inertial Wheel DC Motor

A DC motor with an encoder is used to control the inertial wheel:

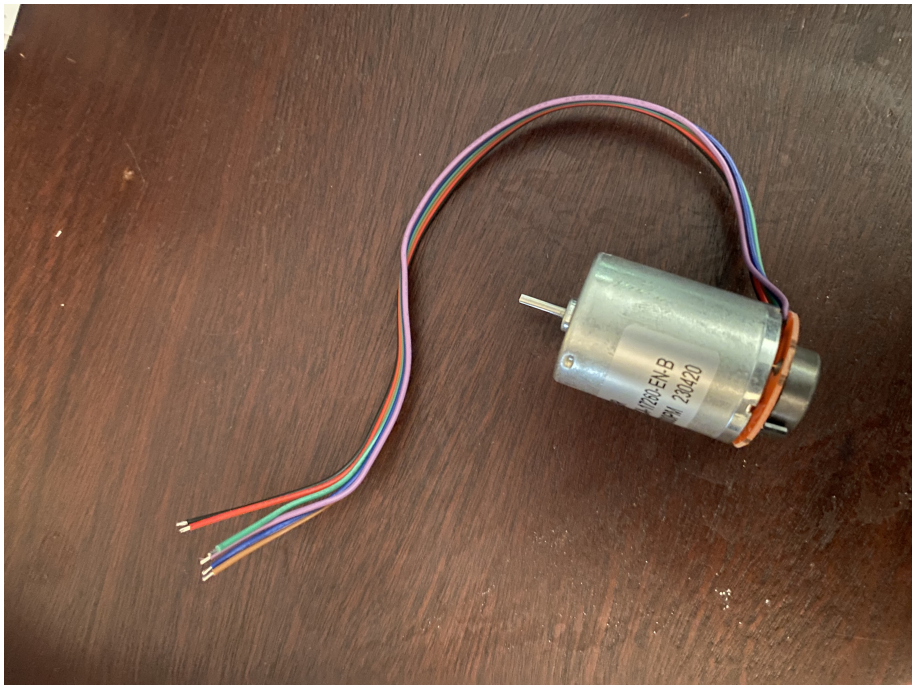


Figure 2.6: Inertial wheel DC motor

A direct current (DC) motor is an actuator that converts electrical energy into rotational motion by generating torque on its shaft when a current is applied across its terminals. The DC motor speed can be regulated by adjusting the supply voltage, while the rotation direction can be reversed by inverting the current flow.

It includes a built-in Hall Sensor based encoder that can measure speed, direction, or positioning of the DC motor. A Hall effect sensor operates on the principle of the Hall effect, which refers to the generation of a voltage difference across an electrical conductor when it is subjected to a magnetic field [13].

In the DC motor encoder, as the rotor disc rotates, magnetic poles pass in front of the sensors. Each time a pole is detected, the encoder produces a digital pulse, commonly referred to as a *tick*. By measuring the frequency of these pulses, the rotational speed of the DC motor can be determined.

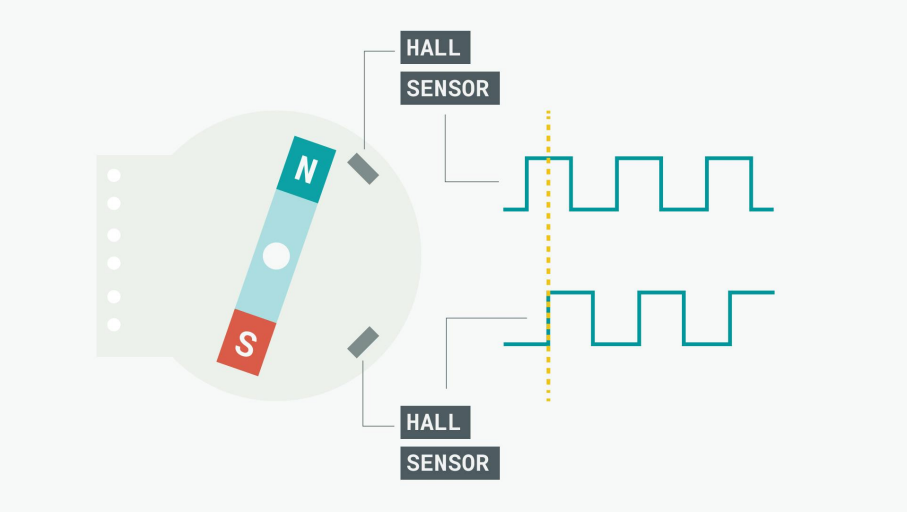


Figure 2.7: Hall sensor illustration [13]

The encoder integrates two Hall effect sensors, each providing a digital output. The sensors are positioned with a phase shift of 90° relative to each other, resulting in two square-wave signals that are out of phase. This configuration is known as a quadrature output.

The phase relationship between the two signals makes it possible to determine the direction of rotation: when output A leads output B, the motor rotates forward, otherwise, the motor rotates in the opposite direction.

Table 2.4: DC Motor Specifications.

Parameter	**Value**
Operating Voltage	6.0 V
No Load Speed	7500 RPM
No Load Current	90 mA
Stall Torque	150 g.cm
Stall Current	2.6 A

2.5.5 Rear Wheel Micro-Geared DC Motor

The system incorporates a micro-geared DC (Direct Current) motor to drive the rear wheel:

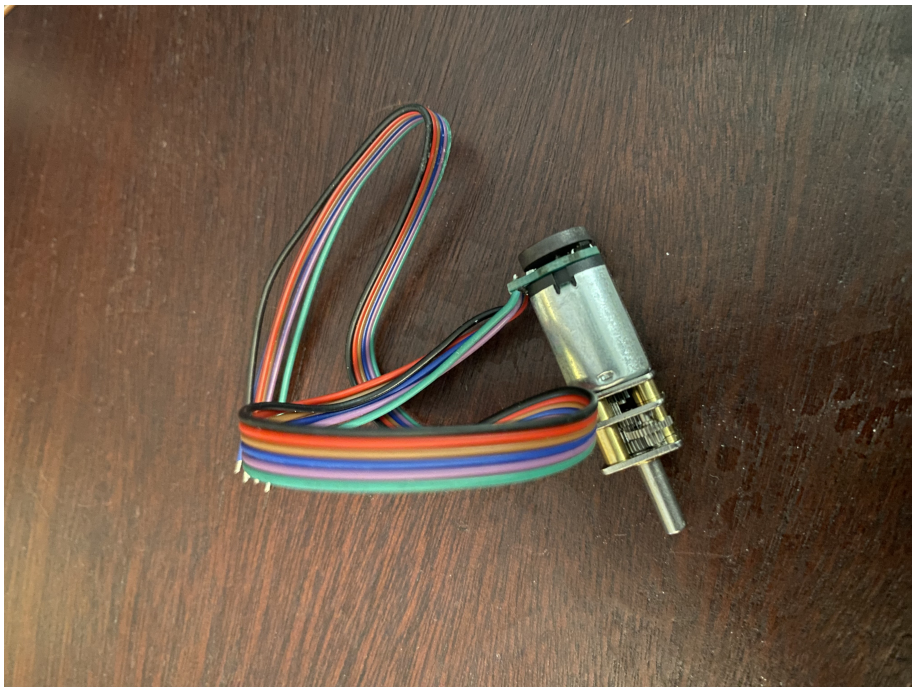


Figure 2.8: Micro-Geared DC Motor with Encoder

To generate enough torque for driving the self-balancing motorcycle, a micro-geared DC motor is used. The gearbox provides a reduction ratio of 100:1, which increases the available torque at the cost of the maximum output speed. In general, higher reduction ratios result in greater torque amplification but lower rotational velocity [13].

Table 2.5: Micro-Geared DC Motor Specifications.

Parameter	**Value**
Operating Voltage	12 V
No Load Speed	300 RPM
No Load Current	70 mA
Stall Torque	2220 g.cm
Stall Current	1.03 A
Gear Ratio	100:1
Ticks per Revolution (without gearbox)	12
Ticks per Revolution (with gearbox)	1200

2.5.6 Standard Servo Motor

A standard servo motor is used to steer the handlebar of the self-balancing motorcycle:

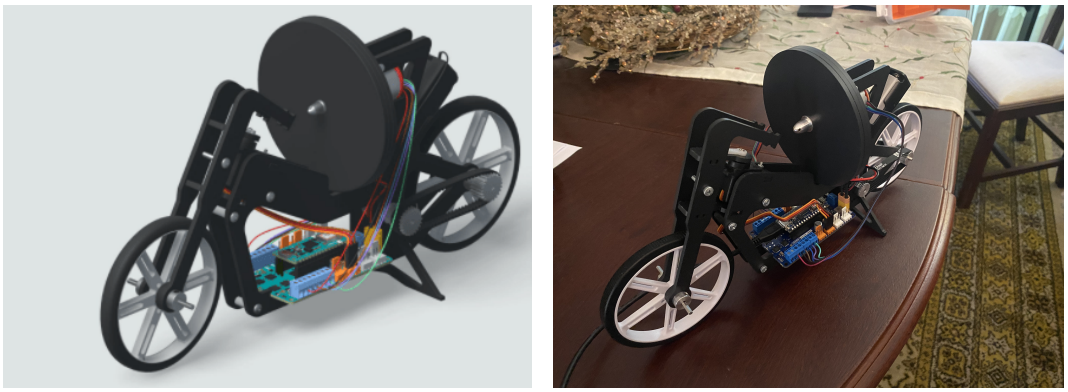
Figure 2.9: Servo motor

Servo motors can be classified into two main types: standard and continuous rotation. In the self-balancing motorcycle, a standard servo is used to control the steering angle. A standard servo consists of a DC motor coupled with a gearbox, a position sensor, and a control circuit. Its operation relies on pulse-width modulation (PWM), where control signals are typically constrained between 1 ms and 2 ms in duration and repeated at intervals of 20 ms [13].

Figure 2.10: Servo motor positioning [13]

2.5.7 Final System

After assembling all the pieces in the kit, this was the final result:



(a) Arduino Version of the Self-Balancing Motorcycle [21]

(b) Personal Version of the Self-Balancing Motorcycle

Figure 2.11: Self-Balancing Motorcycle

The self-balancing motorcycle can be programmed with Simulink to control its balance algorithm.

Chapter 3

Nonlinear Dynamic Modeling and Linearization of a Self-Balancing Motorcycle with Inertial Wheel

The problem of balancing a stationary motorcycle can be formulated as a control problem equivalent to that of stabilising an inverted pendulum. When observed from the rear, a stationary motorcycle can be approximated as a rigid pendulum rotating about the wheel–ground contact line, with an inertial wheel mounted on the pendulum rod.

This analogy allows the use of well-established modelling, which is provided by [21], and control techniques developed for inverted pendulum systems.

3.1 Modelling Assumptions

The following modeling assumptions enable a tractable mathematical representation of the system dynamics:

- The motorcycle rotates only about the wheel–ground contact axis, imposing a single rolling constraint.
- Wheel thickness and deformation are considered negligible.
- Friction in all mechanical joints is neglected.
- The DC motor exhibits a linear torque–current relationship within its operating limits.
- Sensor measurements provide accurate and instantaneous feedback, without delay or noise.

Under these assumptions, the self-balancing motorcycle dynamics can be described as a planar inverted pendulum actuated by an inertial wheel.

3.2 System Description

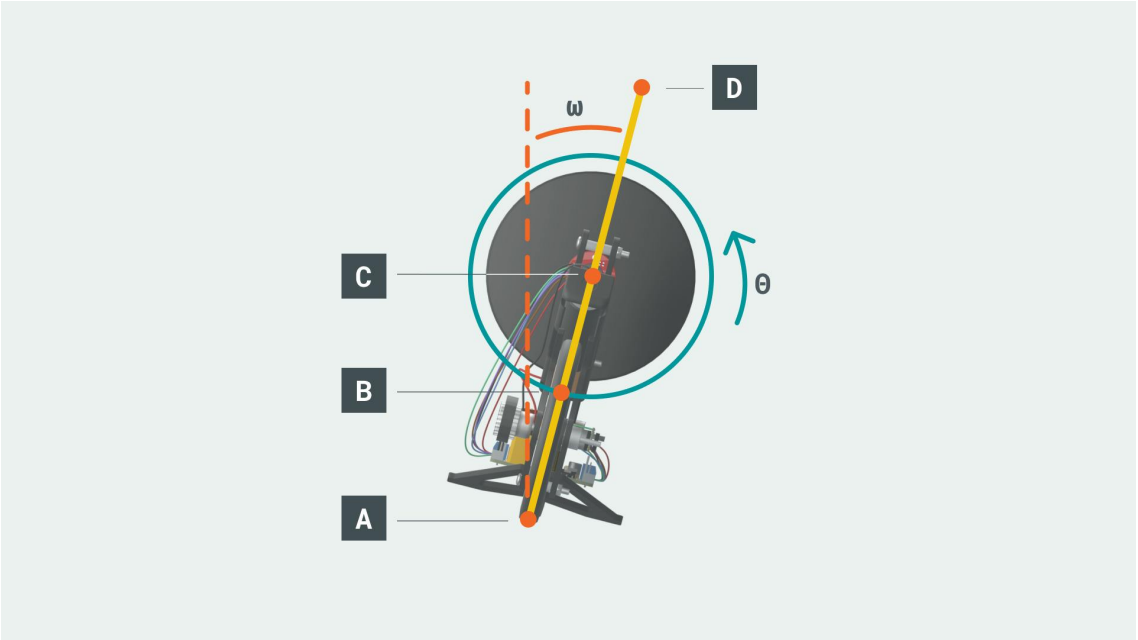


Figure 3.1: System Model [21]

The system consists of two main components. The first is a pendulum rod connected to the ground at point A through a revolute joint with one degree of freedom. The rod is modelled as a uniform solid cylinder of radius r.

The second component is an inertial wheel connected to the pendulum rod at point C through another revolute joint, also with one degree of freedom. The inertial wheel is modelled as a solid cylinder of radius R. The rotation axes of both revolute joints are parallel, ensuring planar motion.

Point B represents the centre of mass of the frame structure, positioned midway between the pivot point A and the upper end D. The inertial wheel couples directly to a DC motor that generates the stabilizing control torque.

3.3 Degrees of Freedom and State Variables

The system has two mechanical degrees of freedom: the lean angle of the pendulum rod and the rotation of the inertial wheel. The state of the system is described using the following variables:

- θ: angular displacement of the pendulum rod, measured from the upright equilibrium position ($\theta = 0$),
- $\dot{\theta}$: angular velocity of the pendulum rod,
- ω: angular velocity of the inertial wheel relative to the pendulum rod.

The rotational position of the inertial wheel is not explicitly included as a state variable, since only its angular velocity and acceleration influence the system dynamics.

3.4 Moments of Inertia

According to Newton's second law for rotation, the torque τ applied to a rigid body is related to its angular acceleration by

$$\tau = I\ddot{\theta}, \tag{3.1}$$

where I is the moment of inertia about the axis of rotation.

Before presenting the moments of inertia used in this work, it is important to introduce the fundamental theorems that relate the inertia properties of rigid bodies with respect to different axes.

These theorems allow inertia values to be transferred between axes and geometries and are essential for deriving the expressions used for planar and three-dimensional bodies.

3.4.1 Parallel Axis Theorem

The Parallel Axis Theorem relates the moment of inertia of a rigid body about an arbitrary axis to the moment of inertia about a parallel axis passing through the center of mass. This theorem is particularly useful when the inertia is known with respect to the center of mass, but the dynamics require the inertia about a displaced axis.

For a rigid body of mass m, the moment of inertia I about an axis located at a distance d from a parallel axis through the center of mass is given by:

$$I = I_{\mathrm{CM}} + mr^2 \tag{3.2}$$

where I_{CM} is the moment of inertia about the center-of-mass axis. The additional term mr^2 represents the increase in rotational inertia due to the translational offset of the mass distribution.

This theorem is extensively used in mechanical modeling, as many systems rotate about axes that do not pass through their center of mass.

3.4.2 Moments of Inertia of Three-Dimensional Bodies

For three-dimensional rigid bodies, such as solid cylinders, the mass is distributed along the third dimension. In these cases, the moments of inertia must be derived directly from volume integrals of the mass distribution or obtained from standard analytical results.

A solid cylinder of radius r, height h, and mass m exhibits different inertia properties depending on the axis of rotation. The moment of inertia about the longitudinal symmetry axis is given by:

$$I_z = \frac{1}{2} m r^2 \tag{3.3}$$

For axes passing through the center of mass and perpendicular to the cylinder axis, the moment of inertia becomes:

$$I_x = I_y = \frac{1}{12} m \left(3r^2 + h^2\right) \tag{3.4}$$

These expressions reflect the three-dimensional mass distribution of the cylinder and are required when modeling rotational dynamics involving tilting or transverse motion.

Applying equation (3.2), the moment of inertia of the inertial wheel about point A is

$$I_w^A = I_w^C + m_w l_{AC}^2, \tag{3.5}$$

where I_w^C is the moment of inertia of the wheel about its centre of mass. Using equation (3.3) results in:

$$I_w^C = \frac{1}{2} m_w R^2 \tag{3.6}$$

Similarly, the moment of inertia of the pendulum rod about point A is

$$I_r^A = I_r^B + m_r l_{AB}^2, \tag{3.7}$$

where I_r^B denotes the rod's moment of inertia about its centre of mass. Applying equation (3.4) results in:

$$I_r^B = \frac{1}{12} m_r \left(3r^2 + l_{AD}^2\right) \tag{3.8}$$

So, the final inertia components become:

$$I_w^A = \frac{1}{2} m_w R^2 + m_w l_{AC}^2 = m_w \left(\frac{1}{2} R^2 + l_{AC}^2\right) \tag{3.9}$$

$$I_r^A = \frac{1}{12} m_r \left(3r^2 + l_{AD}^2\right) + m_r l_{AB}^2 \tag{3.10}$$

3.5 External Torques

Two gravity-induced torques act about the wheel–ground axis at point A. The first is due to the mass of the pendulum rod, and the second is due to the mass of the inertial wheel.

The gravitational torque acting on the pendulum rod is given by

$$\tau_{g,r} = m_r g l_{AB} \sin(\theta), \tag{3.11}$$

while the gravitational torque acting on the inertial wheel is

$$\tau_{g,w} = m_w g l_{AC} \sin(\theta). \tag{3.12}$$

The total gravitational torque acting on the system is therefore the sum of these two contributions:

$$\tau_g = \tau_{g,r} + \tau_{g,w} = m_r g l_{AB} \sin(\theta) + m_w g l_{AC} \sin(\theta). \tag{3.13}$$

In addition to gravity, the DC motor driving the inertial wheel applies a control torque τ_m. This torque accelerates the wheel and produces an equal and opposite reaction torque acting on the pendulum rod.

3.6 Dynamics of the Inertial Wheel

The absolute angular position of the inertial wheel can be expressed as the sum of the pendulum angle and the relative wheel rotation. Defining ω as the angular velocity of the inertial wheel relative to the pendulum rod, the absolute angular velocity of the wheel is

$$\dot{\phi} = \dot{\theta} + \omega. \tag{3.14}$$

Differentiating once more yields the absolute angular acceleration:

$$\ddot{\phi} = \ddot{\theta} + \dot{\omega}. \tag{3.15}$$

Applying Newton's second law for rotation to the inertial wheel about its centre of mass C gives

$$\tau_m = I_w^C(\ddot{\theta} + \dot{\omega}). \tag{3.16}$$

Solving for $\dot{\omega}$ leads to

$$\dot{\omega} = \frac{\tau_m}{I_w^C} - \ddot{\theta}. \tag{3.17}$$

This equation highlights the coupling between the wheel dynamics and the angular acceleration of the pendulum rod.

3.7 Equations of Motion of the Pendulum

The rotational dynamics of the pendulum rod about point A are obtained by applying Newton's second law for rotation. The sum of torques acting on the pendulum rod is given by the total gravitational torque minus the DC motor reaction torque:

$$\tau_g - \tau_m = (I_r^A + I_w^A)\ddot{\theta}. \tag{3.18}$$

Substituting the expressions for the gravitational torque and the moments of inertia yields

$$m_r g l_{AB} \sin(\theta) + m_w g l_{AC} \sin(\theta) - \tau_m = \left[m_w \left(\frac{1}{2} R^2 + l_{AC}^2 \right) + \frac{1}{12} m_r (3r^2 + l_{AD}^2) + m_r l_{AB}^2 \right] \ddot{\theta}. \quad (3.19)$$

Defining the composite inertia term

$$K = m_w \left(\frac{1}{2} R^2 + l_{AC}^2 \right) + \frac{1}{12} m_r (3r^2 + l_{AD}^2) + m_r l_{AB}^2, \quad (3.20)$$

the pendulum equation of motion becomes

$$\ddot{\theta} = \frac{g(m_r l_{AB} + m_w l_{AC})}{K} \sin(\theta) - \frac{1}{K} \tau_m. \quad (3.21)$$

3.8 State-Space Representation

The state vector is defined as

$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \\ \omega(t) \end{bmatrix}. \quad (3.22)$$

Using (3.21) and (3.17), the nonlinear state-space model of the system can be written as

$$\dot{x}(t) = \begin{bmatrix} \dot{\theta} \\ \dfrac{g\,(m_r l_{AB} + m_w l_{AC})}{K} \sin(\theta) - \dfrac{1}{K} \tau_m \\ \dfrac{\tau_m}{I_w^C} - \ddot{\theta} \end{bmatrix}. \quad (3.23)$$

3.9 Linearisation and Linear Time-Invariant State-Space Model

For controller design, the nonlinear model is linearised about the upright equilibrium position, defined by

$$\theta = 0, \qquad \dot{\theta} = 0, \qquad \omega = 0. \quad (3.24)$$

Applying a first-order Taylor expansion, $\sin(\theta) \approx \theta$ is valid for $|\theta| < 0.25\,\text{rad}$ ($\approx 14^\circ$), ensuring the linearization remains accurate for small-angle deviations about vertical equilibrium.

Substituting this approximation into (3.21) results in the linearised pendulum dynamics

$$\ddot{\theta} = \frac{g\,(m_r l_{AB} + m_w l_{AC})}{K} \theta - \frac{1}{K} \tau_m. \quad (3.25)$$

For compactness, it was defined:

$$a = \frac{g\,(m_r l_{AB} + m_w l_{AC})}{K}. \tag{3.26}$$

Thus,

$$\ddot{\theta} = a\,\theta - \frac{1}{K}\tau_m. \tag{3.27}$$

Using this expression in the inertial wheel equation (3.17),

$$\dot{\omega} = \frac{\tau_m}{I_w^C} - \ddot{\theta} = \frac{\tau_m}{I_w^C} - \left(a\,\theta - \frac{1}{K}\tau_m\right) = -a\,\theta + \left(\frac{1}{K} + \frac{1}{I_w^C}\right)\tau_m. \tag{3.28}$$

Collecting the state variables in the vector

$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \\ \omega(t) \end{bmatrix}, \qquad u(t) = \tau_m(t), \tag{3.29}$$

the linearised system can be written in the standard Linear Time-Invariant (LTI) form:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.30}$$

with

$$A = \begin{bmatrix} 0 & 1 & 0 \\ a & 0 & 0 \\ -a & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ -\dfrac{1}{K} \\ \dfrac{1}{K} + \dfrac{1}{I_w^C} \end{bmatrix}. \tag{3.31}$$

This linear time-invariant state-space representation forms the basis for the controller design and analysis presented in the subsequent chapters.

The Arduino Engineering Kit Rev2 provides all the required parameter values to substitute into the general symbolic form:

Table 3.1: Physical parameters and inertia-related quantities of the self-balancing motorcycle model.

Symbol	Description	Value	Units
g	Gravitational acceleration	9.807	$\mathrm{m/s^2}$
m_r	Mass of the pendulum rod	0.2948	kg
m_w	Mass of the inertial wheel	0.0695	kg
R	Radius of the inertial wheel	0.05	m
r	Radius of the pendulum rod	0.02	m
l	Length of the pendulum rod	0.13	m
l_{AD}	Distance from base to rod tip	0.13	m
l_{AC}	Distance from base to inertial wheel	0.13	m
l_{AB}	Distance from base to rod centre of mass	0.065	m
I_w^C	Wheel inertia about its centre of mass	8.688×10^{-5}	$\mathrm{kg\,m^2}$
I_w^A	Wheel inertia about point A	1.3×10^{-3}	$\mathrm{kg\,m^2}$
I_r^B	Rod inertia about its centre of mass	4.447×10^{-4}	$\mathrm{kg\,m^2}$
I_r^A	Rod inertia about point A	1.7×10^{-3}	$\mathrm{kg\,m^2}$
K	Composite inertia about point A	3.0×10^{-3}	$\mathrm{kg\,m^2}$
a	Linearised gravity coefficient	93.68	$\mathrm{s^{-2}}$

The final state-space representation becomes:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 93.68 & 0 & 0 \\ -93.68 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ -3.39 \times 10^2 \\ 1.185 \times 10^4 \end{bmatrix} \tau_m, \tag{3.32}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \tau_m. \tag{3.33}$$

This linearized model is suitable for applying classical linear control techniques such as PID and LQR, as it allows for more straightforward controller design while still capturing the essential dynamics of the self-balancing motorcycle system.

3.9.1 Controllability and Observability Analysis

Before designing any state-feedback controller, it is essential to verify whether the system is controllable and observable. These two properties determine whether it is possible to fully control the system from the inputs and fully reconstruct the internal states from the outputs, respectively [4].

For a linear time-invariant (LTI) system represented in state-space form by matrices A, B, and

C, the system is said to be controllable if the controllability matrix

$$\mathscr{C} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

has full rank, that is, $\text{rank}(\mathscr{C}) = n$, where n is the number of state variables [7].

Similarly, the system is said to be observable if the observability matrix:

$$\mathscr{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

has full rank n.

In this project, both properties were verified numerically using MATLAB. The controllability and observability matrices were constructed for the system defined by the linearized state-space model.

The results confirmed that both the controllability and observability matrices have full rank, equal to the number of states (three in this case). This means that the system is fully controllable and observable, making it suitable for full-state feedback control design.

These properties validate the theoretical basis of the control approach used in this work and confirm that no states are unreachable or unmeasurable from the proposed configuration.

3.10 Transfer Function

The transfer function is one of the most essential representations of a linear time-invariant (LTI) system. As described in [8], the transfer function $F(s)$ is defined as the ratio between the Laplace transform of the output and the Laplace transform of the input, assuming zero initial conditions:

$$F(s) = \frac{Y(s)}{U(s)}. \tag{3.34}$$

This representation characterizes how the system responds to inputs in the frequency domain and allows the analysis of stability, steady-state behaviour, and transient characteristics using algebraic tools rather than differential equations.

The transfer function is obtained from the state-space model by considering the Laplace-transformed system equations, as shown in [8]:

$$sX(s) - x(0) = AX(s) + BU(s), \qquad Y(s) = CX(s) + DU(s). \tag{3.35}$$

Assuming zero initial conditions, the first equation can be rearranged to solve for the state vector:

$$X(s) = (sI - A)^{-1} B U(s). \tag{3.36}$$

Substituting this expression into the output equation yields:

$$Y(s) = \big(C(sI - A)^{-1} B + D\big) U(s). \tag{3.37}$$

Since $Y(s) = F(s)U(s)$, the transfer function of the system is therefore:

$$F(s) = C(sI - A)^{-1} B + D. \tag{3.38}$$

This result highlights that the transfer function can always be derived directly from the system matrices A, B, C, and D. It also demonstrates the connection between the state-space and frequency-domain representations of a system: both describe the same dynamics, but the transfer function expresses them in a form convenient for analysing poles, zeros, and stability.

As further discussed in [8], the transfer function also reveals the order of the system, which corresponds to the degree of the denominator polynomial in $F(s)$. This is equivalent to the number of states in the minimal state-space representation. Understanding this relationship is important because it links the internal dynamics of the system (represented by the state variables) to the external behaviour captured by the input–output relationship.

From the state-space representation of the self-balancing motorcycle model, the corresponding transfer function is obtained by applying the formula. This leads to the following expression:

$$F(s) = \frac{-338.8\,s}{s^3 - 1.776 \times 10^{-15} s^2 - 93.68\,s} \tag{3.39}$$

From the transfer function, it is observed that the system is third order, as indicated by the highest power of s in the denominator. This matches the number of states in the original state-space model, confirming the well-known relationship between the system order and the dimension of its minimal state-space representation.

The transfer function can also be rearranged to facilitate identification of its poles and zeros:

$$F(s) = \frac{-338.8\,s}{s\,(s - 9.679)\,(s + 9.679)} \tag{3.40}$$

From transfer function theory, the roots of the numerator correspond to the system's zeros, while the roots of the denominator correspond to its poles. Inspection of the denominator of the transfer function reveals that one of the poles lies in the right half of the complex plane (i.e., it has a positive real part).

This directly implies that the open-loop self-balancing motorcycle model is unstable. The purpose of the controllers designed in the following chapters is therefore to shift the closed-loop poles into the left half-plane and stabilise the system.

The next chapter analyzes the transfer function in greater detail, with a focus on the influence of its poles and zeros on the system's dynamic behavior.

Chapter 4

Design and Implementation of Control Techniques

This chapter analyzes the design and implementation of three control techniques for the inertial wheel used to stabilize the self-balancing motorcycle. The performance of the different control strategies is evaluated using a set of standard time-domain response metrics commonly used in control system analysis.

As described in [8], these parameters provide a way to define the transient and steady-state behaviour of a system following a reference input. Specifically, the controller responses were analysed in terms of delay time, rise time, peak time, maximum percent overshoot, and settling time.

The delay time t_d is defined as the time required for the system response to reach half of its final steady-state value for the first time.

In this thesis, the rise time t_r is defined as the time required for the response to rise from 0% to 100% of its final value. The maximum percent overshoot quantifies how much the response exceeds its steady-state value during the transient regime and is defined as

$$\frac{y_{\max} - y(\infty)}{y(\infty)} \times 100\%,$$

where $y_{\max}$ denotes the maximum value of the response. Finally, the settling time t_s is the time required for the response to reach and remain within a specified tolerance band around the final value, typically chosen as 2% or 5%; for the 2% criterion, this condition is satisfied when

$$\frac{|y(t) - y(\infty)|}{|y(\infty)|} \leq 0.02 \quad \text{for} \quad t \geq t_s.$$

Together, these parameters provide a description of the transient behaviour of the controlled system, allowing us to compare the controllers in terms of performance: speed, damping, and stability.

4.1 PD and PID Control

This section introduces PID control fundamentals and develops a Simulink implementation based on the linear state-space model of the self-balancing motorcycle. Controller gains are first obtained using the Ziegler–Nichols method, providing fast response but inducing large overshoot and oscillatory behaviour, particularly for PID.

To improve damping and settling, manual tuning is performed for both PID and PD, with the manually tuned PD configuration achieving the best overall transient performance. Finally, pole–zero analysis is used to interpret these results, showing how different tunings shift the closed-loop poles and explaining the observed trade-off between speed, overshoot, and stability.

4.1.1 PID control principles

PID control represents the most widely deployed feedback strategy in industrial applications, valued for its simplicity and effectiveness. It combines three complementary correction mechanisms—proportional (instantaneous error correction), integral (steady-state error elimination), and derivative (transient damping)—to achieve specified performance objectives [4].

The control system $u(t)$ in a PID controller is:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt}$$

where:

- $e(t) = r(t) - y(t)$ is the error, the difference between the reference input $r(t)$ and the output $y(t)$
- K_p: Proportional gain.
- K_i: Integral gain.
- K_d: Derivative gain.

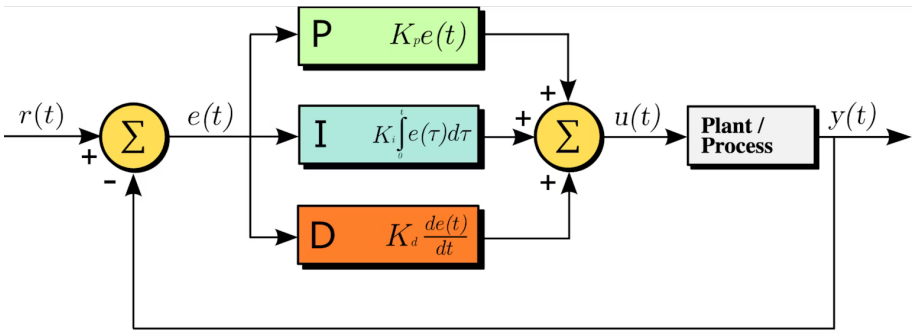


Figure 4.1: PID control system. [26]

Control Actions

The control signal $u(t)$ in a PID controller combines three control actions:

The Proportional (P) control action produces a response proportional to the current error $e(t)$, scaled by the proportional gain K_p. This action provides immediate correction based on the magnitude of the error but cannot eliminate steady-state error on its own.

The Integral (I) control action addresses the accumulation of past errors by integrating the error over time. This action, scaled by the integral gain K_i, ensures that the system reaches and remains at the desired setpoint, effectively eliminating steady-state error.

The Derivative (D) control action predicts future error behavior by computing the rate of change of the error. Scaled by the derivative gain K_d, this action helps reduce oscillations and improves system stability and response speed by counteracting rapid changes in the error.

Table 4.1: Advantages and Disadvantages of PID Control.

Advantages	**Disadvantages**
Simple to understand and easy to implement in many systems.	Requires careful tuning of the parameters (K_p, K_i, K_d) for optimal performance.
Usable in many systems (e.g., temperature control, robotics).	Can be less effective for highly nonlinear systems or systems with significant uncertainties.
Provides good control by minimizing error through the proportional, integral, and derivative terms.	Sensitive to noise, especially in the derivative term, which may lead to bad performance
Can achieve fast response and high accuracy when properly tuned.	The integral term can cause slow system response if not tuned carefully, potentially leading to overshoot or instability.
Helps stabilize systems and reduce steady-state error through integral action.	In some systems, derivative action may amplify noise or cause chattering.

4.1.2 Simulink system model

A Simulink model was developed to simulate the PID and PD control strategies. The model incorporates the state-space representation derived in Chapter 3 and applies a step input as the reference signal, which in this case is set to zero. The output of the state-space block corresponds to the lean angle of the self-balancing motorcycle.

The PID controller is implemented by feeding back the system output through three parallel paths. The proportional term is obtained by applying a gain directly to the measured output, while the derivative term is realised using a derivative block followed by a gain to scale the derivative action. In addition, the integral term is implemented by integrating the output signal and applying a gain to weight the integral contribution.

The three components are then summed to generate the final control action applied to the system. The PD control follows the same principle, but the integral part is removed.

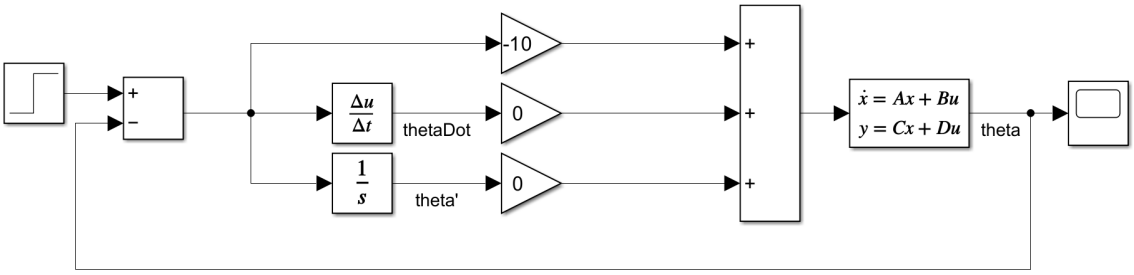


Figure 4.2: System model for PD controller

4.1.3 Ziegler–Nichols Method

The Ziegler–Nichols method [25] was used as the first strategy to tune the controllers. The procedure involves the following steps:

1. Set K_d and K_i to zero.
2. Gradually increase K_p from zero until the system output exhibits sustained oscillations at a constant frequency.
3. Measure the value of $K_u = K_p$ at this point (the ultimate gain), and the oscillation period T_u.
4. Compute the controller gains K_p, T_i, and T_d according to the following table:

Table 4.2: Ziegler-Nichols Tuning Rules.

Control Type	K_p	T_i	T_d
P	$0.5K_u$	–	–
PI	$0.45K_u$	$T_u/1.2$	–
PD	$0.8K_u$	–	$T_u/8$
Classic PID	$0.6K_u$	$T_u/2$	$T_u/8$

The value of the constant gain block was gradually increased until the system began oscillating sinusoidally, allowing the identification of the ultimate gain K_u. The oscillation period P_u was determined by measuring the time between two successive peaks in the response.

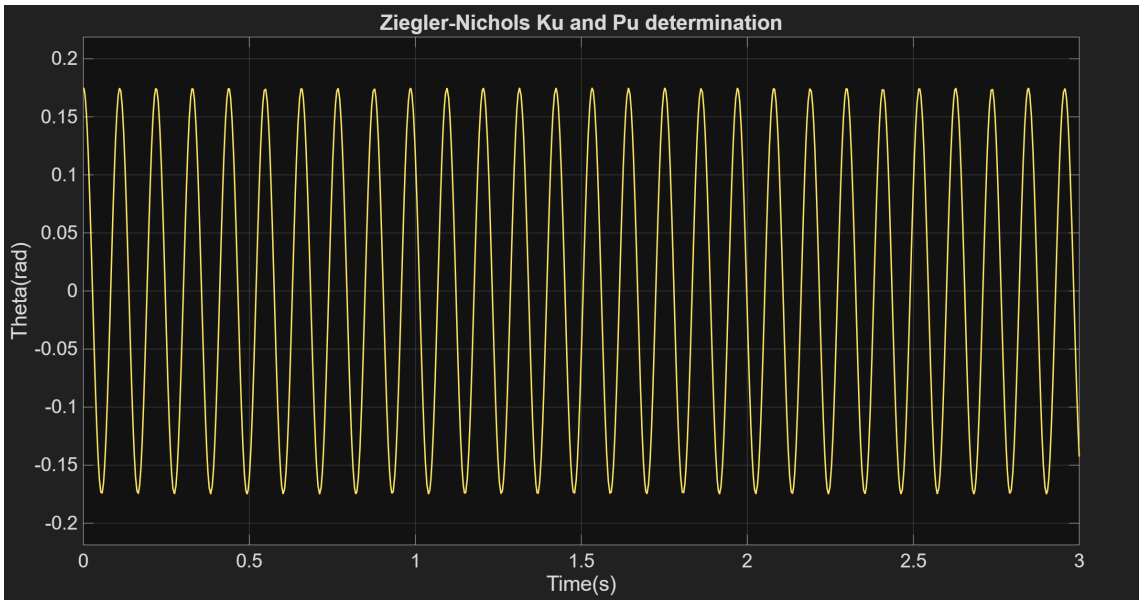


Figure 4.3: Ku and Pu determination

The calculated parameters were:

$$K_u = -10 \tag{4.1}$$

$$P_u = 0.110\,\mathrm{s} \tag{4.2}$$

Using the Ziegler–Nichols tuning rules for a PID controller, the gains were computed as follows:

$$K_p = 0.6 \cdot K_u = 0.6 \cdot (-10) = -6 \tag{4.3}$$

$$K_i = \frac{K_p}{0.5 \cdot P_u} = \frac{-6}{0.5 \cdot 0.110} = -109.09 \tag{4.4}$$

$$K_d = \frac{P_u}{8} \cdot K_p = \frac{0.110}{8} \cdot (-6) = -0.0825 \tag{4.5}$$

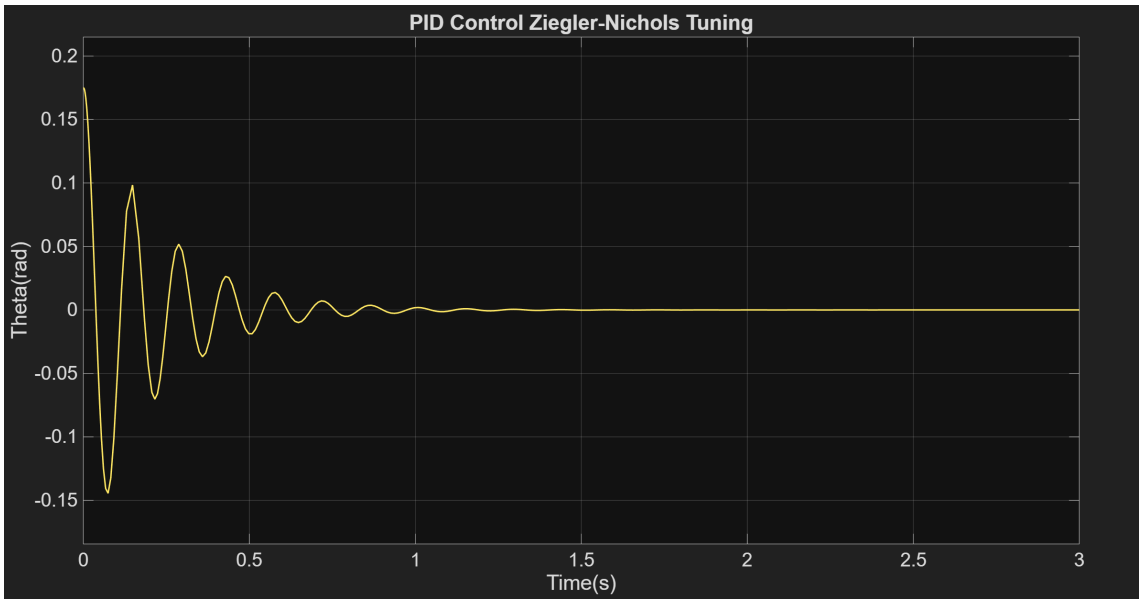


Figure 4.4: Closed-loop lean angle response using a PID controller tuned via the Ziegler–Nichols method. Simulation performed using the linearized model with zero reference input and an initial lean angle displacement of $\theta(0) = 0.175\,\text{rad}$. Controller gains: $K_p = -6$, $K_i = -109.09$, $K_d = -0.0825$.

Table 4.3: Metrics for the PID controller (Ziegler–Nichols).

Metric	**Value**
Delay time t_d	25 ms
Rise time t_r	37.6 ms
Maximum percent overshoot M_p	83.1% (0.1455)
Settling time t_s	969.4 ms

As shown in Figure 4.4, the PID controller is able to drive the lean angle to zero, but the system presents substantial oscillations.

To reduce these oscillations, a PD controller was also implemented. According to the Ziegler–Nichols table for PD control:

$$K_p = 0.8 \cdot K_u = 0.8 \cdot (-10) = -8 \tag{4.6}$$

$$K_d = \frac{P_u}{8} \cdot K_p = \frac{0.110}{8} \cdot (-8) = -0.110 \tag{4.7}$$

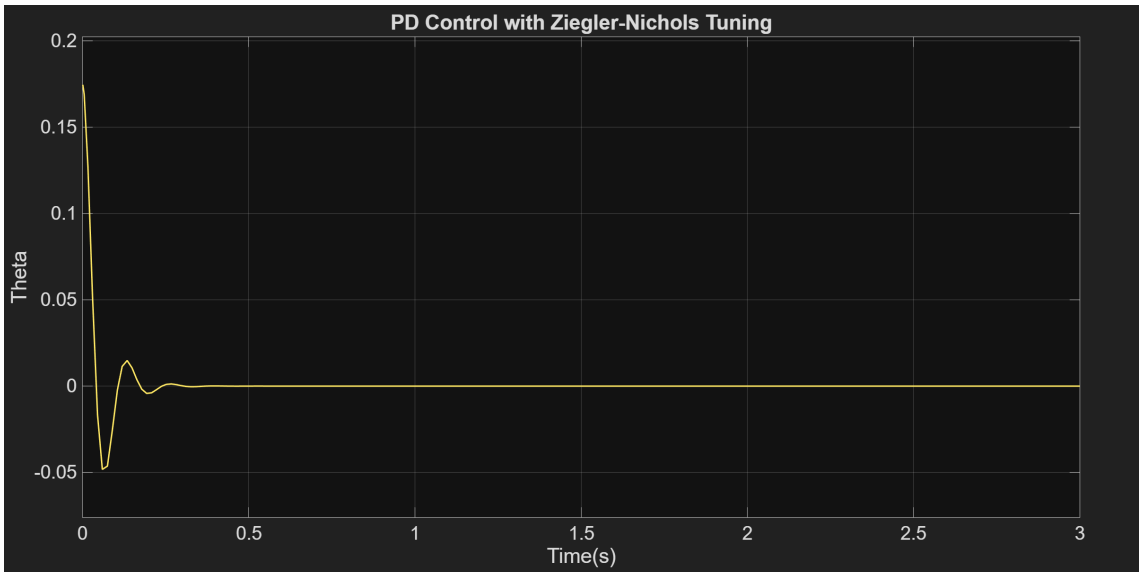


Figure 4.5: Closed-loop lean angle response using a PD controller tuned via the Ziegler–Nichols method. Simulation performed using the linearized model with zero reference input and an initial lean angle displacement of $\theta(0) = 0.175\,\text{rad}$. Controller gains: $K_p = -8$, $K_d = -0.110$.

Table 4.4: Metrics for the PD controller (Ziegler–Nichols).

Metric	**Value**
Delay time t_d	23 ms
Rise time t_r	41 ms
Maximum percent overshoot M_p	28.11% (0.0492)
Settling time t_s	203 ms

By removing the integral term, the system exhibited significantly fewer oscillations, a lower overshoot, and a faster settling time. However, some residual oscillations remained. To further improve the performance, a manual tuning procedure was applied.

4.1.4 Manual Tuning Method

After several simulations and iterative adjustments, the following gain values were found to provide an acceptable system response:

$$\text{PID: } K_p = -50, \quad K_i = -5, \quad K_d = -1 \tag{4.8}$$

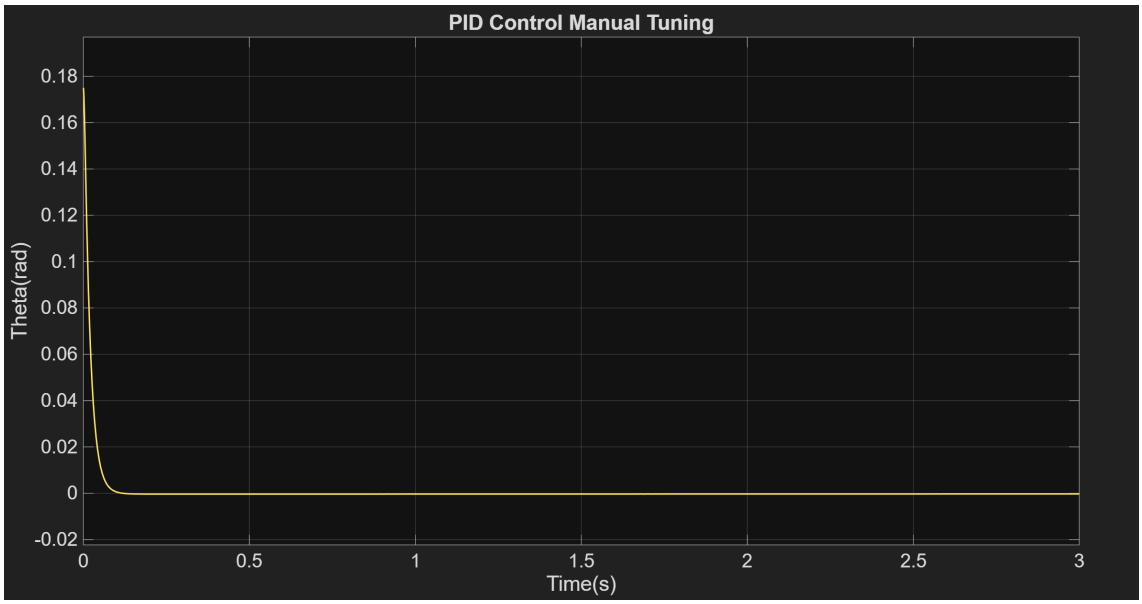


Figure 4.6: Closed-loop lean angle response using a manually tuned PID controller. Simulation performed using the linearized model with zero reference input and an initial lean angle displacement of $\theta(0) = 0.175\,\text{rad}$. Controller gains: $K_p = -50$, $K_i = -5$, $K_d = -1$.

Table 4.5: Metrics for the PID controller (manual tuning).

Metric	**Value**
Delay time t_d	15.4 ms
Rise time t_r	100 ms
Maximum percent overshoot M_p	0%
Settling time t_s	73 ms

$$\text{PD: } K_p = -150, \quad K_d = -2.5 \tag{4.9}$$

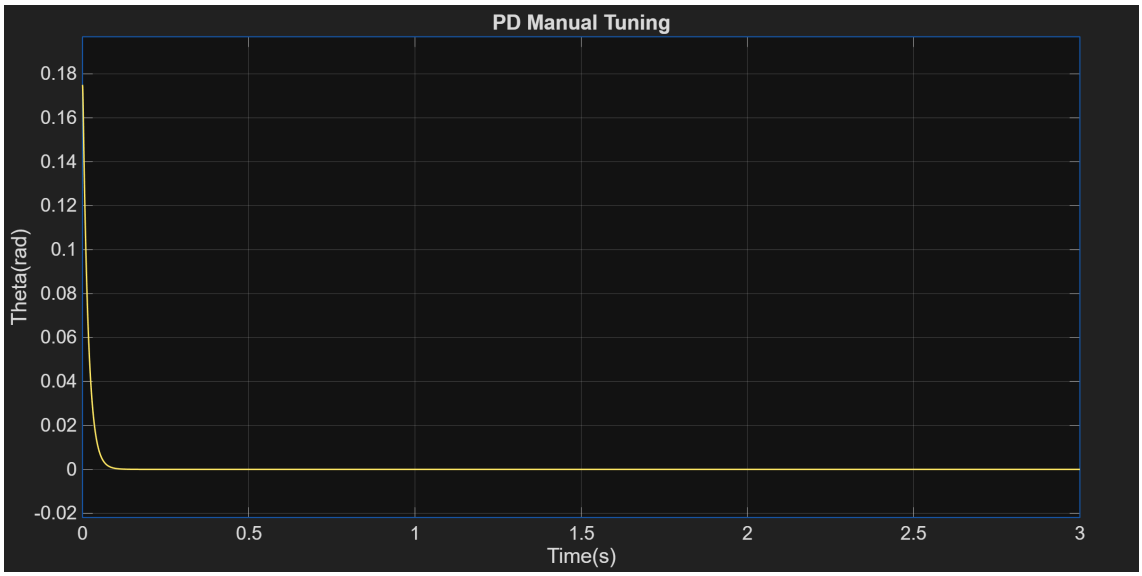


Figure 4.7: Closed-loop lean angle response using a manually tuned PD controller. Simulation performed using the linearized model with zero reference input and an initial lean angle displacement of $\theta(0) = 0.175\,\text{rad}$. Controller gains: $K_p = -150$, $K_d = -2.5$.

Table 4.6: Metrics for the PD controller (manual tuning).

Metric	**Value**
Delay time t_d	12.3 ms
Rise time t_r	92.9 ms
Maximum percent overshoot M_p	0%
Settling time t_s	64.3 ms

The Ziegler–Nichols tunings prioritise a fast initial response, which is reflected in the short rise times for both controllers (PID: 37.6 ms, PD: 41 ms). However, this comes at the cost of large overshoot and longer settling times.

In particular, the PID Ziegler–Nichols design shows a very high overshoot (83.1%) and a long settling time (969.4 ms), indicating a lightly damped transient with high oscillatory behaviour. The PD Ziegler–Nichols controller performs better, but still presents some overshoot (28.11%) and a settling time of 203 ms.

In contrast, the manually tuned controllers eliminate overshoot entirely for both PID and PD, which indicates improved damping. Although the manually tuned designs generally show longer rise times (PID: 100 ms, PD: 92.9 ms), the overall transient is smoother and converges faster to the steady-state region. This is clear in the shorter settling times for the PD manual tuning (64.3 ms) and PID manual tuning (73 ms). Overall, the manual tuning PD control showed the best results.

4.1.5 Pole-Zero Map for stability analysis

A pole–zero map provides a graphical representation of the closed-loop dynamics by plotting the locations of the system's poles and zeros in the complex s-plane. The position of each pole directly determines the behaviour of an associated dynamic mode: poles in the left half-plane correspond to stable, exponentially decaying responses, while poles near the imaginary axis lead to slow or weakly damped motion, and poles in the right half-plane indicate instability.

Zeros affect the shape of the transient response by changing how different modes contribute to the output [15]. From the theory([8]) the influence of a pole on the system response depends directly on its location in the complex plane.

Let's consider two poles, p_i and p_j, with real parts λ_i and λ_j, respectively, both negative but with λ_i much more negative than λ_j. The contribution of each pole to the time response contains a term of the form $e^{\lambda t}$. For the pole with a very negative real part, the term $e^{\lambda_i t}$ decays extremely quickly, meaning that its influence disappears almost immediately.

On the other hand, the pole whose real part is closer to zero produces a term $e^{\lambda_j t}$ that decays much more slowly. This slow-decaying component persists for longer and therefore dominates the long-term behaviour of the system. It may then be concluded that the pole with the real part closest to zero is the most influential, as its associated mode decays the slowest.

The imaginary part of a pole determines the oscillatory nature of its contribution. A complex pole $p_k = \sigma_k \pm j\eta_k$ generates sinusoidal terms in the response. The value η_k corresponds to the angular frequency of the oscillation. Complex poles with large imaginary parts therefore introduce high-frequency oscillations, whereas those with smaller imaginary parts generate lower-frequency oscillations. The real part dictates the rate of decay, while the imaginary part dictates how fast the oscillation happens.

Zeros also affect system behaviour, although their influence is typically more indirect. Zeros shape the response by altering the contribution of nearby poles. If a zero z_i lies close to a pole p_j, the residue associated with that pole becomes small, meaning its effect on the output is diminished. This is commonly referred to as partial pole–zero cancellation. Consequently, a zero located near a pole reduces the influence of that pole.

The pole–zero map in Figure 4.8 illustrates the poles and zeros of the open-loop, uncontrolled system. From this representation, it is clear that one of the poles lies in the right half-plane—that is, it has a positive real part. This directly indicates that the open-loop system is unstable, since any pole located in the right half of the complex plane causes the system response to diverge over time.

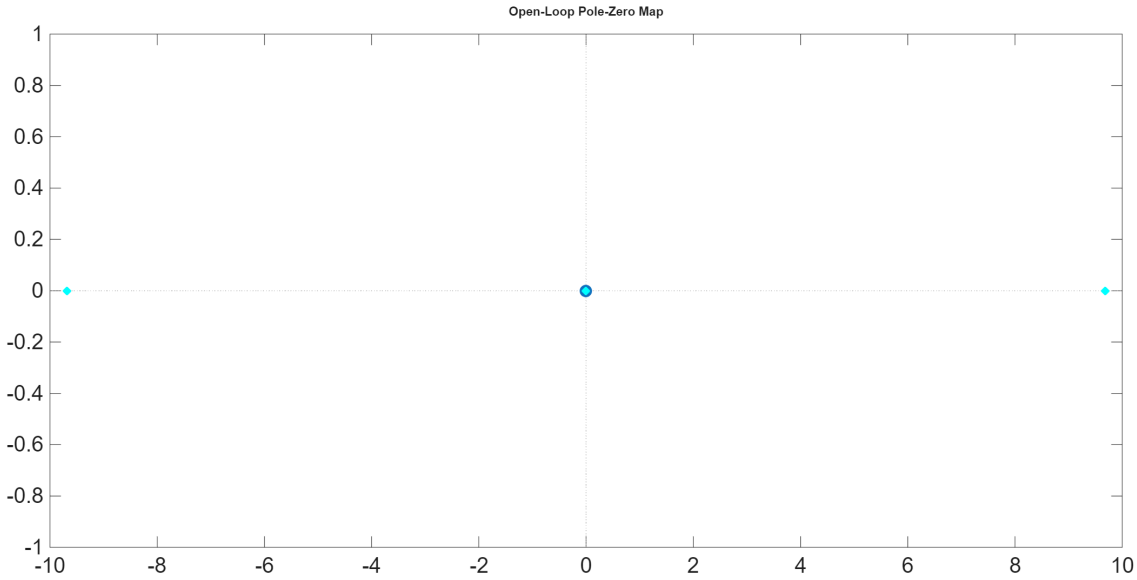


Figure 4.8: Open-Loop System Pole-Zero Map

Table 4.7: Poles and zeros of the uncontrolled system.

Poles	**Zeros**
0	
$+9.6790$	3.77×10^{-15}
-9.6790	

The pole–zero maps in Figures 4.9–4.10 show how each controller configuration changes the closed-loop dynamics by shifting the system's poles and zeros in the complex plane. Starting with the PID Ziegler–Nichols controller, the introduction of proportional, integral, and derivative action adds an additional pole to the closed-loop transfer function, as expected from the inclusion of integral action.

The remaining dominant poles form a complex-conjugate pair with relatively small negative real parts and large imaginary components, indicating lightly damped, high-frequency oscillations. Although the system becomes marginally stable, the high-frequency oscillatory modes and a near pole–zero cancellation at low frequency reduce the robustness of the resulting dynamics.

The PID Manual Tuning configuration also introduces an integrator pole for the same reasons. In contrast to the Ziegler–Nichols design, however, the remaining poles are placed more conservatively. Two poles are located deep in the left half-plane, corresponding to fast and strongly damped modes, while a third real pole lies closer to the imaginary axis but remains stable. This pole structure leads to smoother transient behaviour and improved damping compared to the Ziegler–Nichols PID controller.

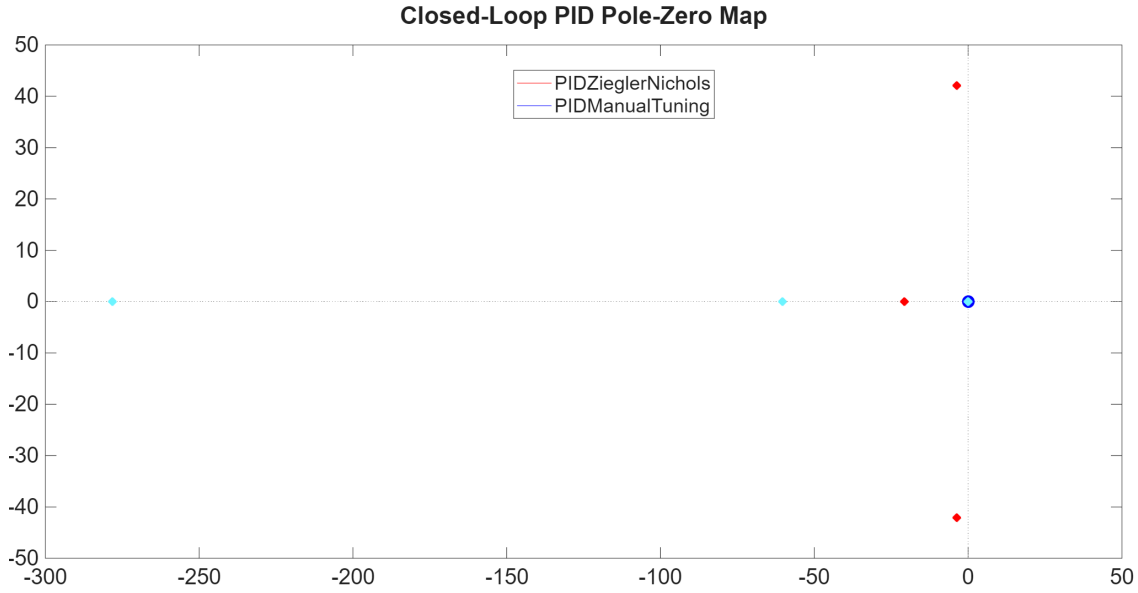


Figure 4.9: Closed-Loop PID Pole-Zero Map

Table 4.8: Poles and zeros of the PID controllers.

PID Controller	**Poles**	**Zeros**
Ziegler–Nichols	$-3.6431 + 42.1339i$ $-3.6431 - 42.1339i$ -20.6646 0	0 3.77×10^{-15}
Manual Tuning	-278.2837 -60.4135 -0.1008 0	0 3.77×10^{-15}

The PD controller tuned using the Ziegler–Nichols method does not include integral action and therefore does not introduce an additional pole into the closed-loop system. The remaining poles form a complex-conjugate pair with relatively small negative real parts and large imaginary components, producing high-frequency, lightly damped oscillations similar to those observed with the Ziegler–Nichols PID tuning.

In contrast, the manually tuned PD controller repositions these poles entirely along the real axis, eliminating the imaginary component and thus suppressing oscillatory behaviour. This pole placement results in strongly damped dynamics and a smoother transient response.

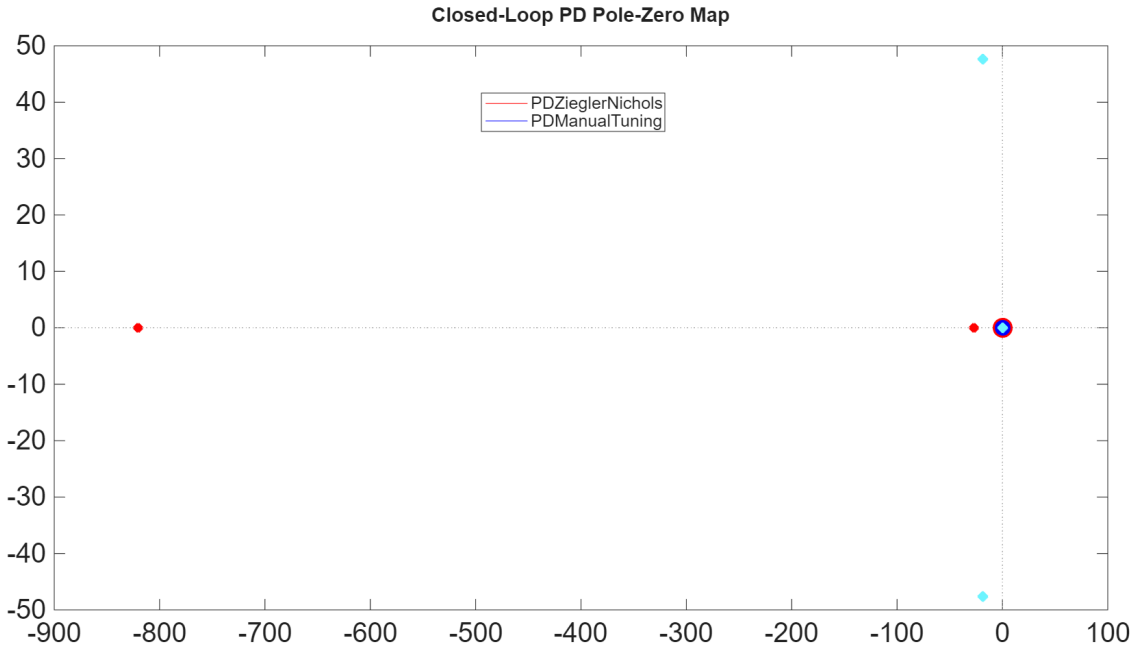


Figure 4.10: Closed-Loop PD Pole-Zero Map

Table 4.9: Poles and zeros of the PD controllers.

PD Controller	Poles	Zeros
Ziegler–Nichols	$-18.6339 + 47.6390i$ $-18.6339 - 47.6390i$ 0	3.77×10^{-15}
Manual Tuning	-820.2617 -26.7332 0	3.77×10^{-15}

4.2 Simscape Alternative Model

This section describes the development of the Simscape-based physical model of the self-balancing motorcycle and its validation against the Simulink implementation. A disturbance torque is then introduced to evaluate closed-loop robustness, showing effective disturbance rejection.

Finally, the hardware workflow is established through testing of the IMU, encoder, battery, DC motor, and servo, followed by a high-level control architecture with sensor processing and safety logic. Physical PD experiments present short-term balance but reveal inertial-wheel saturation caused by residual angle bias, motivating the development of other control strategies in subsequent sections.

4.2.1 Developing the model

Inside the Simscape model are three solids: two cylindrical and one brick. The brick acts as a platform for the pendulum and the cylindrical represent the rod of the pendulum and the inertial wheel. There is a world frame that acts as a global referencial frame. It is inertial and at rest, which means when connecting the brick to the frame, it acts as a fixed platform inertial frame.

The mechanism configuration defines the mechanical and simulation parameters such as gravity, and the solver configuration (f(x) = 0) defines the solver settings of differential equations. The speed and position of the first joint, as well as the speed of the second joint, are sent to the controller. The first joint receives a disturbance torque and the second joint receives the torque from the controller in order to balance the inverted pendulum.

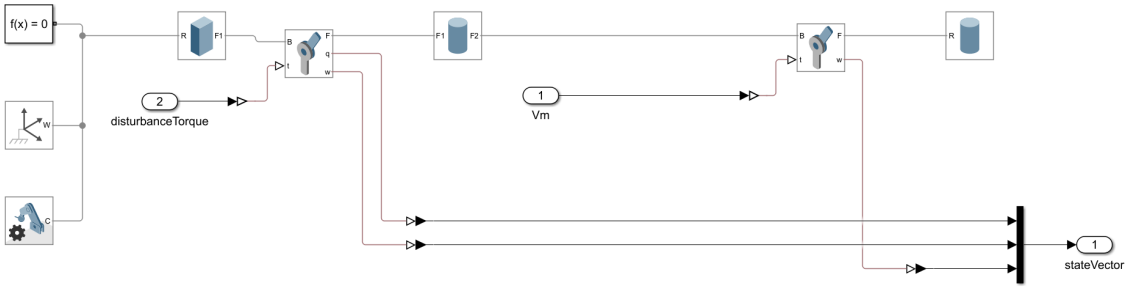


Figure 4.11: Simscape Subsystem

The angle output of the second joint is sent to the PD block input, and the output is sent to the second joint torque input. A pulse generator block is used to simulate the disturbance torque that is sent to the first joint torque input.

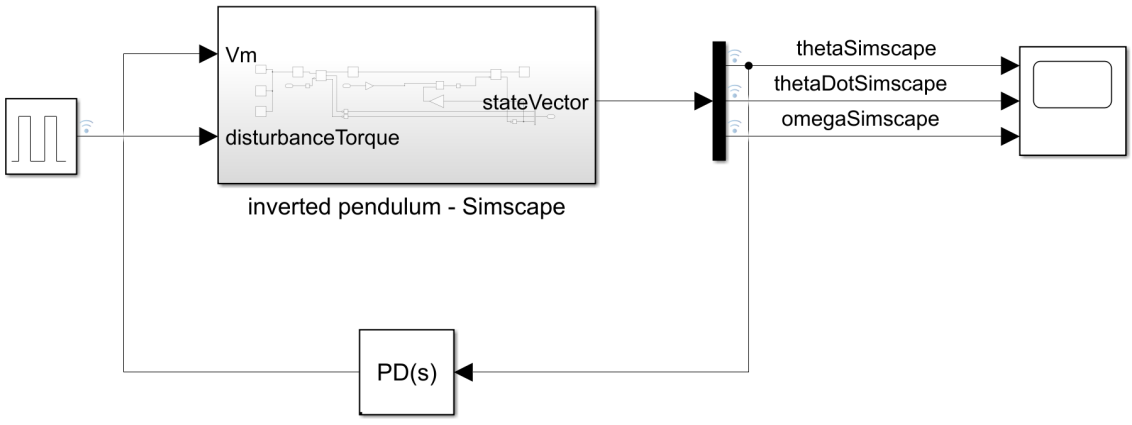


Figure 4.12: Simscape System

For the first simulation, the disturbance torque was set to zero in order to compare the behavior of the Simscape model with the Simulink model. In the first joint, an initial position of 10 degrees was defined.

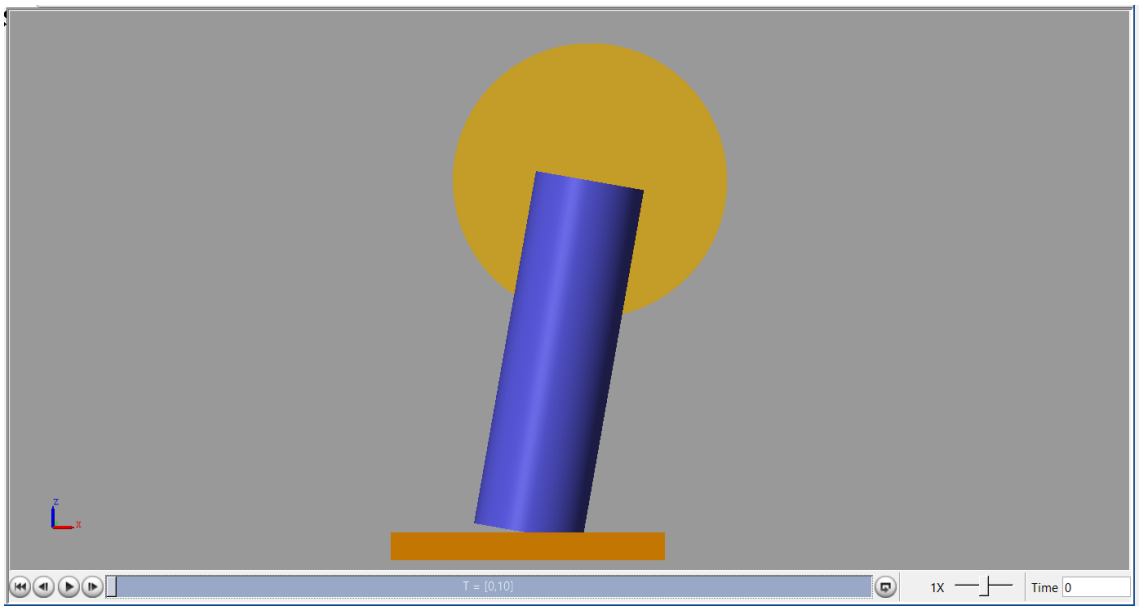


Figure 4.13: Simscape Model Initial Position

By simulating both the Simscape model and the Simulink model, it was observed that they provide identical outputs, which is what was expected since the same controller is used in both models.

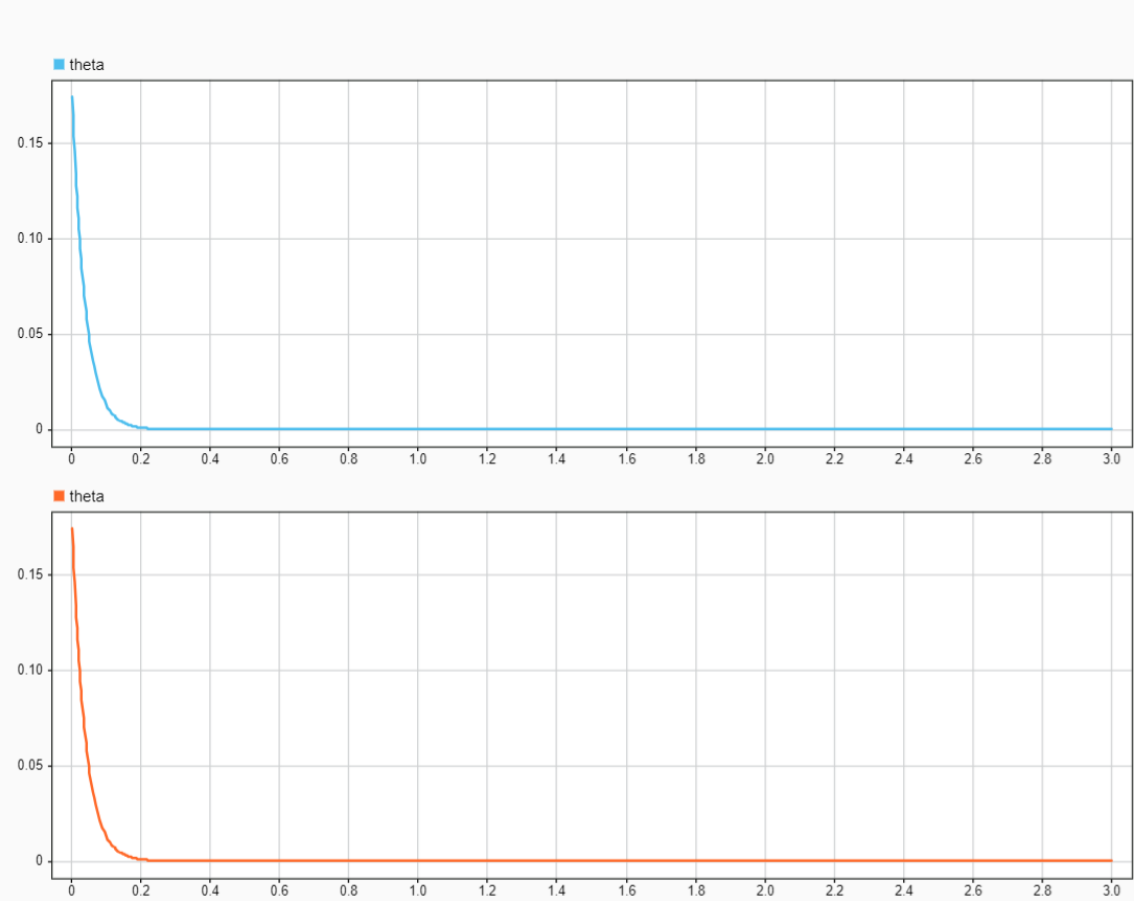


Figure 4.14: Scope Comparison between Simscape and Simulink

4.2.2 Disturbance Torque

After the model was working, a disturbance torque was implemented to ensure that the controller can balance the pendulum in various situations.

A disturbance input was introduced to the first joint to simulate an event in which the self-balancing motorcycle loses balance, for example, due to uneven terrain. This disturbance occurs every second after the start of the simulation using a pulse generator.

The controller was able to successfully stabilize the self-balancing motorcycle in presence of disturbances, as shown in Figure 4.15.

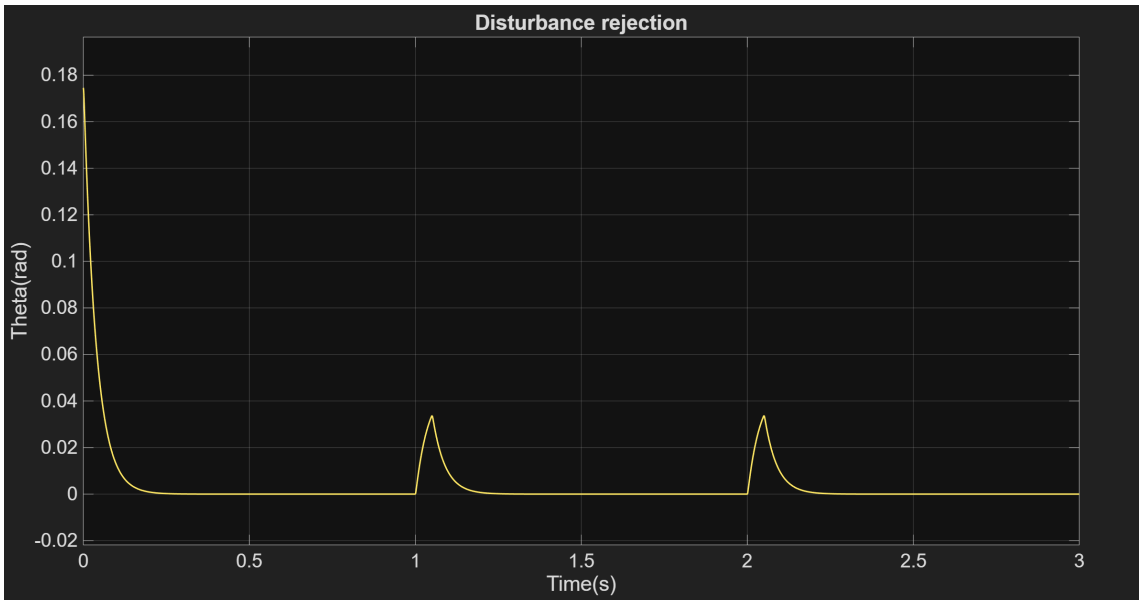


Figure 4.15: Disturbance Scope Analysis

4.3 Hardware Implementation of Closed Loop Control

In this section the hardware components were tested and a high-level architecture was created, combining the previously designed control with the hardware components to implement a balance control application that runs on the Arduino board.

4.3.1 Component testing

Before proceeding to the physical implementation of the control models, each component was tested to confirm that everything is working as expected.

IMU testing

In order to obtain accurate outputs from the IMU, the sensors need to be initialized to locate the gravity vector and magnetic field, as well as properly offset the coordinates.

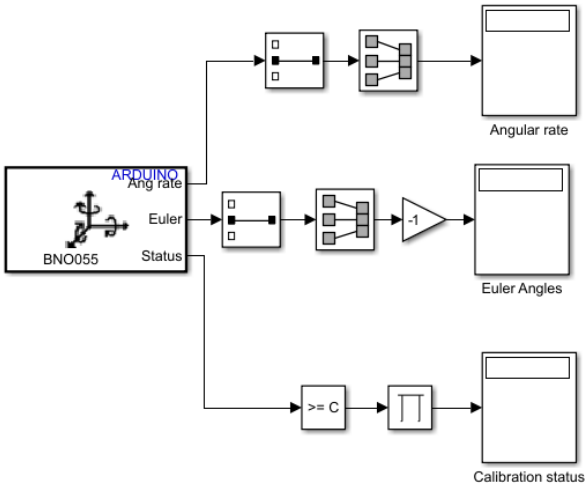


Figure 4.16: IMU calibration

The status output represents the calibration status vector, which includes:

- IMU system
- Gyroscope
- Accelerometer
- Magnetometer

Each vector element has a value of 0 to 3 to indicate the degree of calibration of the sensor. For the self-balancing motorcycle, the magnetometer needs to be fully calibrated, so the status output was compared to a vector [0;0;0;3].

The angular rate output represents the angular velocity around each of the 3 spatial axes. IMU testing indicated that the second element of the vector corresponds to the angular velocity about the Y-axis, which is the controlled variable.

The same thought process was used for the Euler angles. The second element represents the lean angle of the self-balancing motorcycle around the Y axis.

The BNO055 IMU defines θ and $\dot{\theta}$ with opposite polarity; therefore, a constant block of -1 was added to the Euler angles to achieve the same polarity.

Rotary encoder testing

The DC motors used have a built-in encoder which outputs the angle of the DC motor. Since we want to obtain the angular speed, a Simulink model was developed, that reads the data from the encoder and calculates the angular speed.

The data from the encoder represents ticks. 48 ticks corresponds to one full rotation of the DC motor. With this information, a simulink model was built using a filtered derivative block to obtain the angular speed of the DC motor, since the derivative of the position of the DC motor is the velocity.

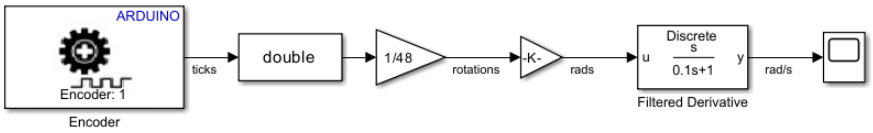


Figure 4.17: Rotary Encoder Testing

After experimenting with different time constants, the chosen value was 0.1s. Higher time constants offer smooth signals, while lower time constants results in a faster reaction to changes.

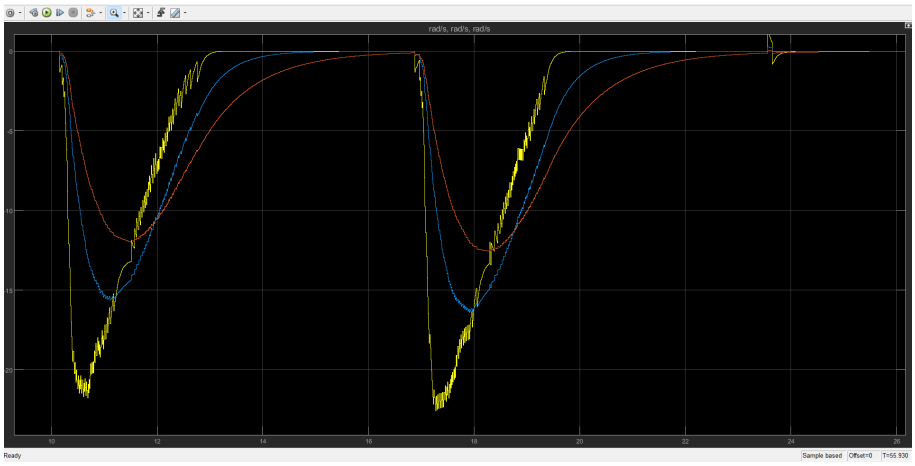


Figure 4.18: Different Time Constants Analysis

In order to test different values of time constants, three filtered derivatives were added with values of 0.1, 0.5 and 1 seconds. This was only for analysis, when implementing the program in the self-balancing motorcycle only the filtered derivative with the 0.1s time constant was used.

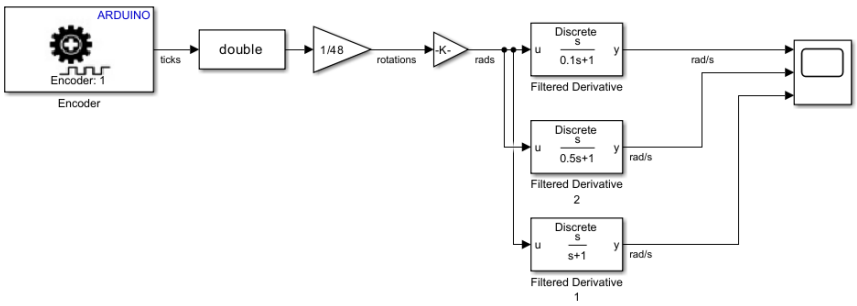


Figure 4.19: Rotary Encoder System with different Time Constants

Battery testing

In this section, the battery was tested to make sure it was fully charged. Lower values of battery can result in a poor performance. The battery simulink block returns a value from 0 to 4095, so a gain of 1/236 was introduced to obtain de measurment in Volts.

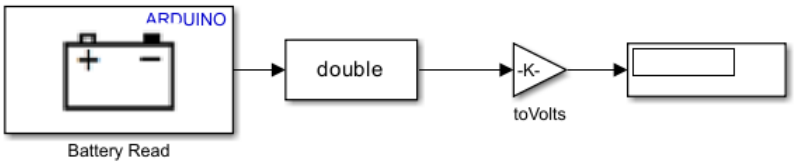


Figure 4.20: Battery testing

Inertial wheel DC motor

After testing the encoder, the DC motor was tested using a slider from the Simulink library. The DC motor accepts values within the range of -255 to 255. Negative values make the DC motor spin in the opposite direction of the positive values. A saturation block was also added as a safety measure to prevent exceeding the DC motor limits.

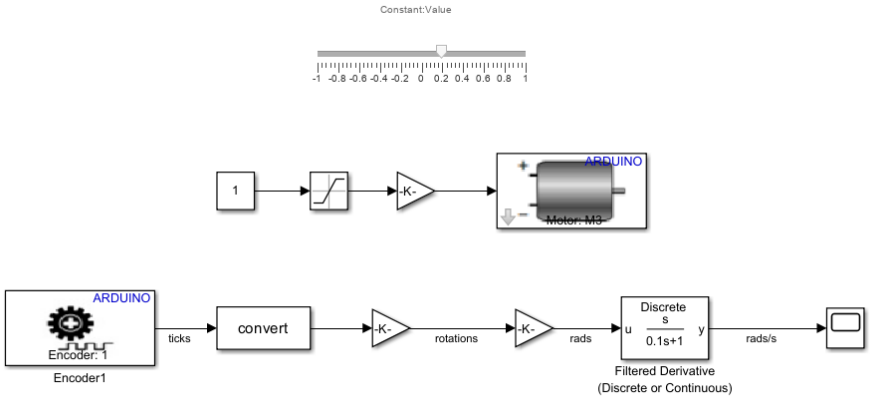


Figure 4.21: DC Motor Testing

Servo motor testing

The servo motor can be used for steering the self-balancing motorcycle.

The servo write block accepts an input between 0 and 180, corresponding to the angle of the servo motor shaft. The servo was calibrated in a way that the 90 degree angle corresponds to the front wheel pointing forward. A saturation block was added due to the range of motion of the steering wheel being smaller than 180, as a safety measure to prevent the servo motor from pushing beyond the limits.

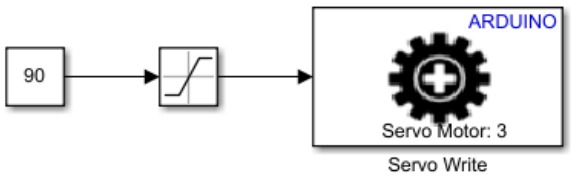


Figure 4.22: Servo test and calibration

4.3.2 High-Level Architecture

A high-level architecture was created consisting of two subsystems, the controller and the motorcycle. The controller outputs the control command to the motorcycle, and the motorcycle returns the raw sensors to the controller for error compensation.

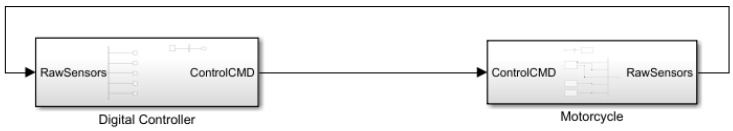


Figure 4.23: High-Level Architecture

4.3.3 Digital Controller

4.3.3.1 Sensor Processing

Inside the digital controller subsystem, another subsystem that contains the sensors processing was included. To improve the model readability, a Bus Selector and a Bus Creator were added, which are simulink blocks that help with signal routing. Using the programs previously tested in section 4.1, the sensors are brought together into this subsystem.

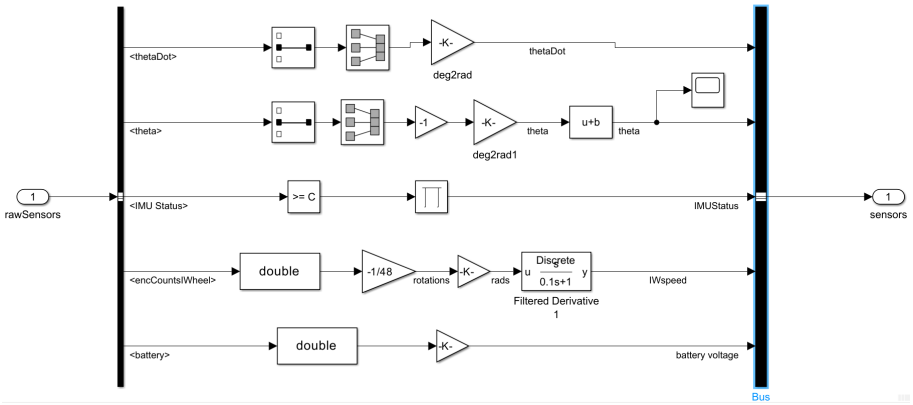


Figure 4.24: Signal Processing subsystem

4.3.3.2 Safety Logic

In this section safety logic is implemented to counteract possible errors and conditions that may occur during when testing the self-balancing motorcycle, like:

- IMU calibration
- Battery level
- Losing balance
- Enabling control by the user

This logic control can be implemented by the Stateflow library in Simulink.

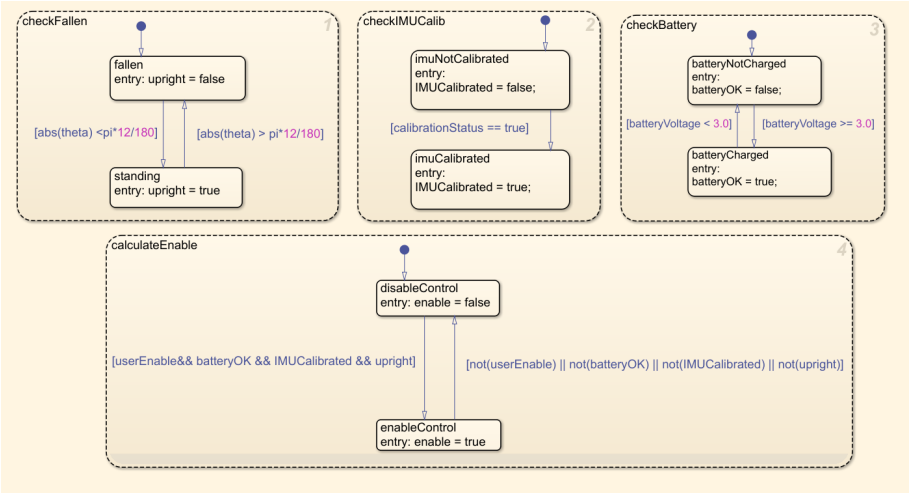


Figure 4.25: Safety Logic Subsystem

This chart contains four parallel states that execute simultaneously. Each state handles a specific event. The last state verifies all the information in the other states and determines the output of the chart.

4.3.3.3 Implementing Feedback Control

Another subsystem was created in the digital controller that contains a feedback control. The first feedback controller that was tested was the PD control developed in the first section of this chapter.

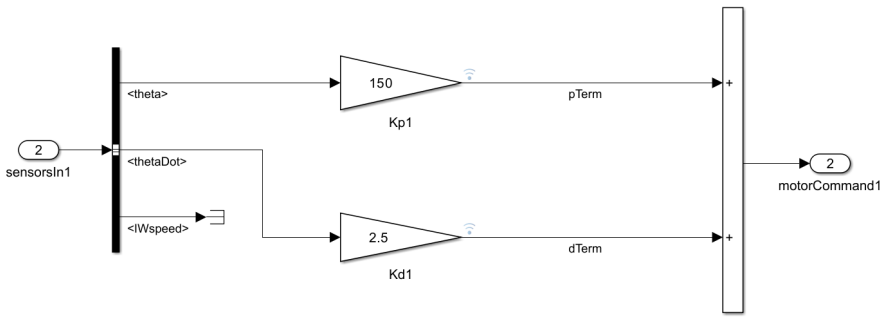


Figure 4.26: PD Control

4.3.3.4 Final Digital Control Subsystem

The final digital control subsystem includes all of the implementations mentioned in this section, as well as a user enable switch that allows the user to freely turn on or off the balancing mechanism. The DC motor is controlled by a True/False switch, that decides if the DC motor receives values from the controller subsystem or not.

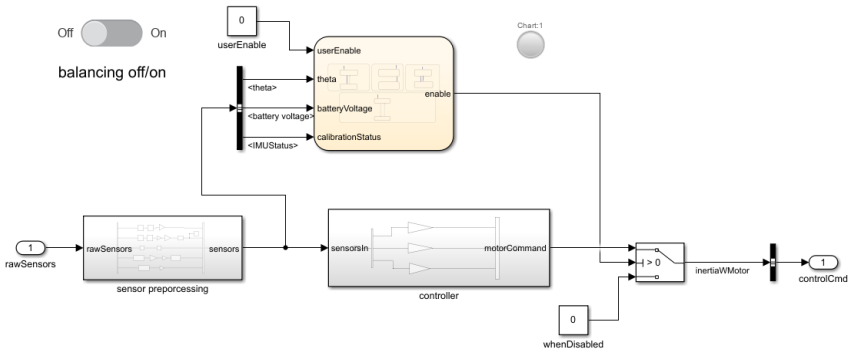


Figure 4.27: Final Digital Subsystem

4.3.4 Self-Balancing Motorcycle Subsystem

The digital controller subsystem outputs the control values to the self-balancing motorcycle subsystem, which sends those values to the DC motor and read the data from the IMU to send back to the digital controller subsystem.

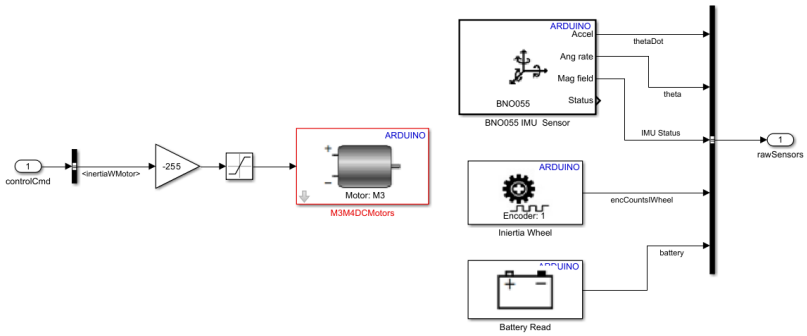


Figure 4.28: Final Self-Balancing Motorcycle Subsystem

4.3.5 PD Manual Tuning Physical Implementation Tests

In this section, the previously developed control code is uploaded to the self-balancing motorcycle, and physical experiments are performed to evaluate the behaviour of the theoretical PD controller. The constant gain values obtained through manual tuning are used for these tests.

The results show that the self-balancing motorcycle is able to maintain balance for approximately 10 seconds before the inertial wheel saturates, after which the system becomes unstable.

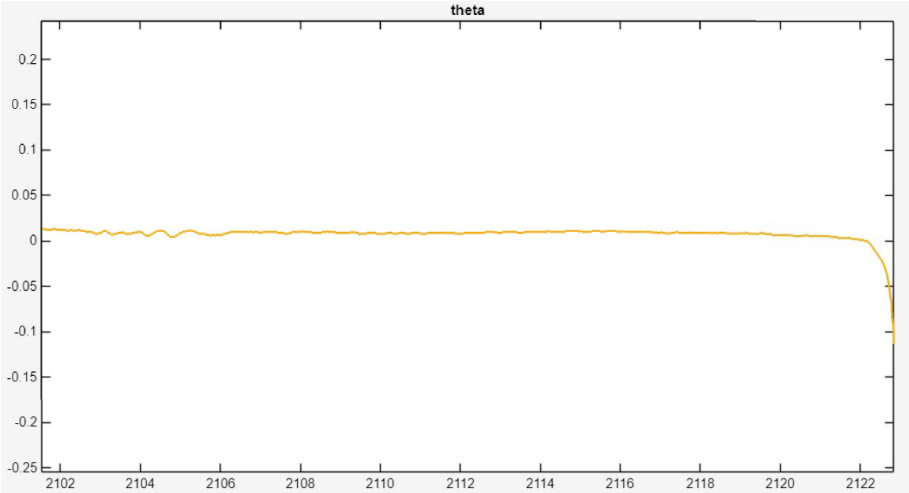


Figure 4.29: PD Control Physical Implementation

This behaviour occurs because the IMU measurement of the lean angle is never identically zero, meaning that even in a near-upright position the PD controller continues to generate a nonzero control torque. Over time, this persistent residual actuation drives the inertial wheel toward its saturation limit, ultimately causing a loss of stability. This limitation motivates the need for an improved control strategy, which is addressed in the following section.

4.4 Linear Quadratic Regulator Control

This section presents the design and evaluation of a Linear Quadratic Regulator (LQR) for stabilizing the self-balancing motorcycle based on a linearized state-space model. The controller is designed using Bryson's rule as an initial tuning strategy, followed by identity and manual tuning to improve transient performance.

Simulation and physical implementation results demonstrate fast stabilization, effective disturbance rejection, and robust real-world behavior, with the manually tuned configuration achieving the best overall performance.

4.4.1 LQR control principles

The Linear Quadratic Regulator (LQR) is an optimal full-state feedback controller that minimizes a quadratic cost functional combining state deviation and control effort [6, 9, 20]. It is widely adopted in control engineering and robotics because it guarantees closed-loop stability while balancing performance objectives and energy consumption.

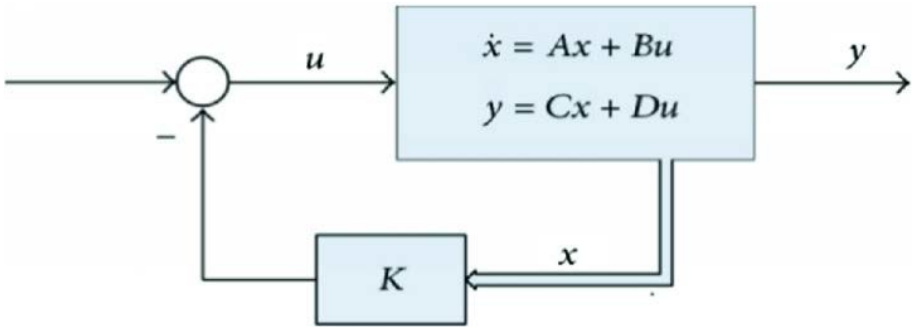


Figure 4.30: LQR Regulator [9]

LQR is applied to a linear system described by the state-space equation:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.10}$$

where:

- $x(t)$ is the state vector,
- $u(t)$ is the control input,
- A is the system dynamics matrix,
- B is the input matrix.

The objective is to minimize the quadratic cost function:

$$J = \int_0^\infty \left(x^T Qx + u^T Ru\right) dt \tag{4.11}$$

where:

- Q is a positive semi-definite state weighting matrix,
- R is a positive definite control weighting matrix.

The optimal control law is a state-feedback controller given by:

$$u(t) = -Kx(t) \tag{4.12}$$

where the optimal gain matrix K is formulated as:

$$K = R^{-1}B^T P \tag{4.13}$$

and P is the solution to the Algebraic Riccati Equation (ARE):

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{4.14}$$

It is proven to be a strong approach used to control linearized systems in the state-space domain in various cases such as a rotary inverted pendulum [23], Ball-On-Sphere System [9], and lateral path tracking for autonomous vehicles [28].

4.4.2 LQR Controller Design

To design the LQR controller, several simulations were performed to identify the most effective configuration. Initially, the Bryson's rule was applied to compute the weighting matrices Q and R. Then, the Q matrix was initialized as the identity matrix and $R = 1$. Finnaly, manual tuning was carried out, which resulted in improved performance when compared to the default Bryson configuration.

Bryson's Rule for LQR Tuning

Bryson's Rule is a systematic approach for selecting weighting matrices Q and R, particularly valuable when meaningful maximum acceptable values for states and control inputs are available.

According to Bryson's Rule, each diagonal entry of matrix Q is computed as the inverse square of the maximum acceptable value of the corresponding state variable. Similarly, each diagonal element of matrix R is computed as the inverse square of the maximum acceptable value of the input variable:

$$q_{ii} = \frac{1}{|x_{i,\max}|^2}, \quad r_{jj} = \frac{1}{|u_{j,\max}|^2} \tag{4.15}$$

This method ensures that each variable is penalized proportionally to how much it is allowed to deviate from acceptable limits. Larger deviations from smaller limits are penalized more heavily.

The state limitations used to define the weighting matrix according to Bryson's rule are

- Lean angle: $\theta_{\max} = 10^\circ = 0.1745$ rad
- Angular velocity: $\dot{\theta}_{\max} = 3$ rad/s
- Rotor speed: $\omega_{\max} = 500$ rad/s

In addition to the state constraints, a limitation on the control action is also imposed. The maximum available DC motor torque is specified in the DC motor datasheet as a stall torque of 150 g·cm. This value is therefore adopted as the maximum control input $\tau_{m,\max}$ when applying Bryson's rule to define the control weighting matrix R.

Using Bryson's Rule:

$$q_{11} = \frac{1}{(0.1745)^2} \approx 32.83 \tag{4.16}$$

$$q_{22} = \frac{1}{3^2} = 0.1111 \tag{4.17}$$

$$q_{33} = \frac{1}{500^2} = 4 \times 10^{-6} \tag{4.18}$$

$$R = \frac{1}{150^2} \approx 4.44 \times 10^{-5}. \tag{4.19}$$

Therefore, the matrix Q becomes:

$$Q = \begin{bmatrix} 32.83 & 0 & 0 \\ 0 & 0.1111 & 0 \\ 0 & 0 & 4 \times 10^{-6} \end{bmatrix} \tag{4.20}$$

The parameter R becomes:

$$R = 4.44 \times 10^{-5}. \tag{4.21}$$

The corresponding gain matrix is:

$$K = \begin{bmatrix} -926.1260 & -61.6660 & -0.3002 \end{bmatrix} \tag{4.22}$$

In addition to the initial tuning using Bryson's rule, two other LQR tuning strategies were explored to improve system performance:

- **Identity Tuning:** The state weighting matrix Q was defined as the identity matrix and the control weighting R was set to 1.

  $$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad R = 1$$

  This configuration led to the following feedback gain matrix:

  $$K_3 = \begin{bmatrix} -667.9209 & -70.0214 & -1.0000 \end{bmatrix}$$

- **Manual Tuning:** The matrices Q and R were manually adjusted based on simulation results. The chosen values were:

  $$Q = \begin{bmatrix} 14500 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \times 10^{-5} \end{bmatrix}, \qquad R = 3,$$

which resulted in the following state-feedback gain matrix:

$$K = \begin{bmatrix} -69.8769 & -0.9298 & -0.0018 \end{bmatrix}.$$

4.4.3 Simulink Simulation of the LQR Controller

A Simulink simulation model was developed based on the complete state-space representation of the system. The control law used in this simulation is of the form:

$$u(t) = -Kx(t)$$

where K is the gain matrix computed using MATLAB's `lqr()` function, and $x(t)$ is the full state vector:

$$x = \begin{bmatrix} \theta & \dot{\theta} & \omega_r \end{bmatrix}^\top$$

The state-space representation becomes:

$$\dot{x} = (A - BK)x$$

The Simulink implementation consists of integrators and matrix gain blocks to model the system dynamics and apply state feedback. The gain matrix K is used to compute the control input based on the current state of the system.

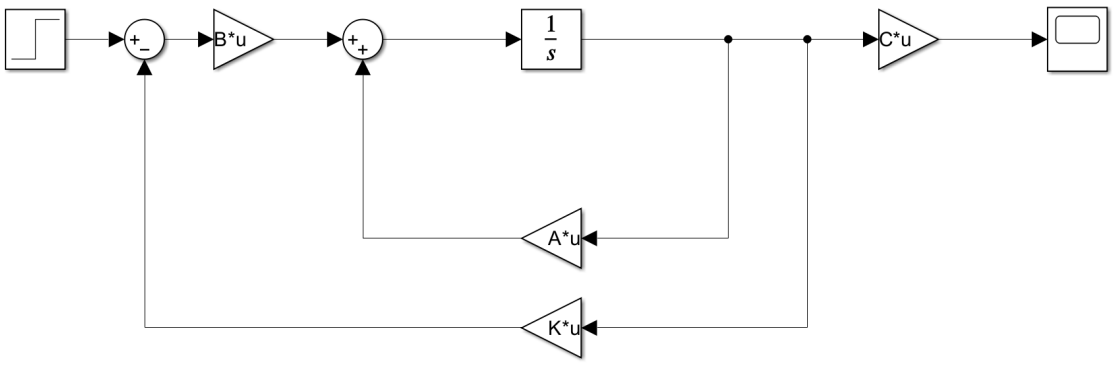


Figure 4.31: Scope analysis of the LQR controller implementation

The Simulink diagram in Figure 4.31 represents the state-space representation. The LQR controller is applied through a gain matrix K, feeding back all three states to stabilize the system.

The goal of this simulation was to validate the Simulink code for future physical implementations in the self-balancing motorcycle. To do this, the output of the LQR-controlled system was analyzed by inspecting its connected `Scope` block.

4.4.3.1 Analysis of the Control Response

Figure 4.32 presents three simulation results for comparison. The red curve corresponds to the controller designed using Bryson's rule. The green curve corresponds to the identity tuning and the blue curve is the manual tuning method.

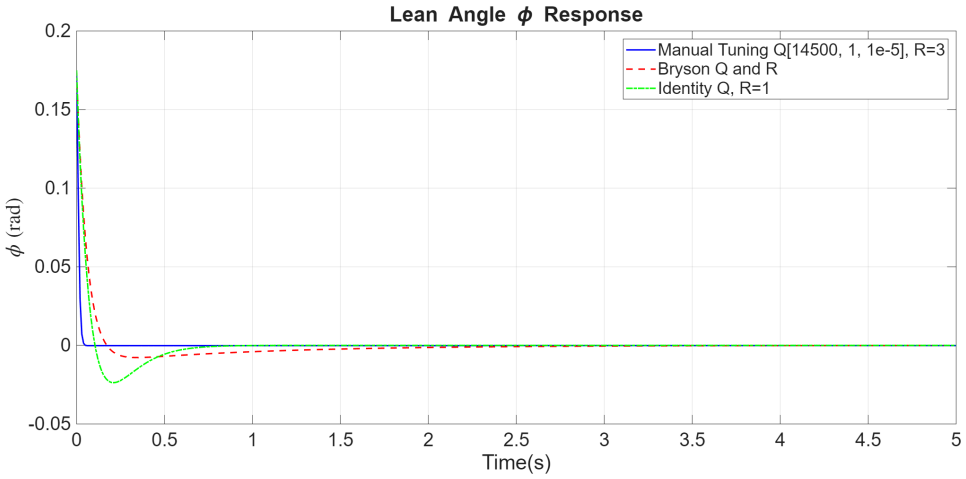


Figure 4.32: Comparison of system response under different LQR configurations.

Table 4.10: Time-domain performance metrics for θ using LQR with Bryson's rule.

Metric	**Value**
Delay time	40 ms
Rise time	110 ms
Maximum percent overshoot	4.42%
Settling time	1110 ms

Table 4.11: Time-domain performance metrics for θ using LQR with identity weighting.

Metric	**Value**
Delay time	35 ms
Rise time	170 ms
Maximum percent overshoot	13.5%
Settling time	570 ms

Table 4.12: Time-domain performance metrics for θ using manually tuned LQR.

Metric	**Value**
Delay time	13 ms
Rise time	50 ms
Maximum percent overshoot	0%
Settling time	40 ms

By comparing the performance metrics obtained for the three LQR tuning strategies, clear improvements in closed-loop behaviour can be observed as the controller design evolves from the Bryson-based tuning to the identity weighting and finally to the manually tuned configuration.

When moving from the Bryson tuning to the identity weighting, the system response becomes significantly faster. The rise time is reduced from 110 ms to 170 ms and the delay time is slightly improved, indicating a more responsive controller. Additionally, the maximum percent overshoot increases from 4.42% to 13.5%, revealing that the identity tuning sacrifices damping for speed.

Although this results in a quicker transient response, it also introduces more oscillations. The settling time is reduced from 1110 ms to 570 ms, showing that the identity weighting achieves a better compromise between responsiveness and convergence speed compared to the Bryson tuning.

Further improvements are made when transitioning from the identity weighting to the manually tuned LQR controller. In this case, all transient performance metrics improve simultaneously. The delay time is reduced to 13 ms and the rise time to 50 ms, producing a very fast response. At the same time, the maximum percent overshoot is completely eliminated.

The settling time is drastically reduced to 40 ms, showing a significant improvement in stability and robustness. This shows that manual tuning was able to retain the fast dynamics introduced by the identity weighting while eliminating oscillations and achieving better damping.

Overall, the progression from Bryson to identity and finally to manual tuning reflects a progression from conservative, constraint-driven control toward a more performance-oriented design. While Bryson's rule provides a safe initial tuning and identity weighting improves speed, manual tuning allows the controller to fully explore the system dynamics, resulting in the best overall performance among the three approaches.

4.4.4 Disturbance Analysis

After achieving a satisfactory result through manual tuning, the model response to a disturbance torque using the same physical model from Section 4.2 was analyzed.

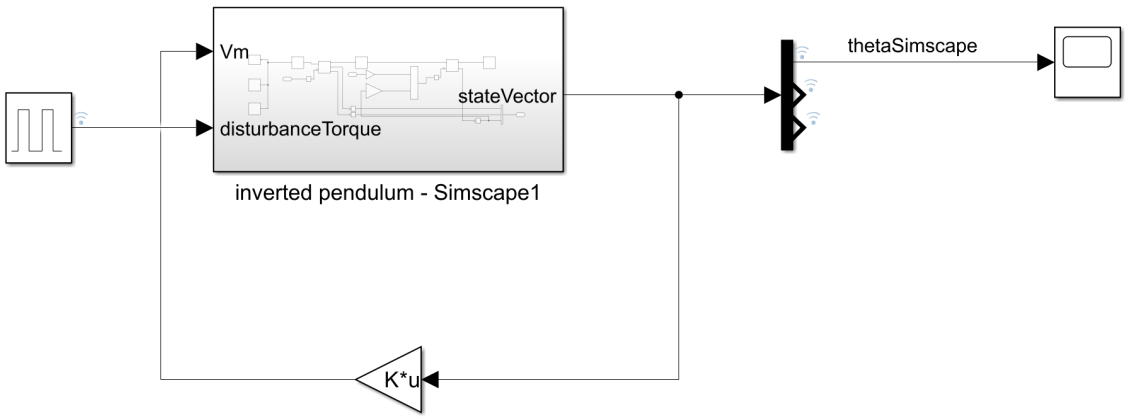


Figure 4.33: Simulink model for disturbance analysis

Similar to the PD control, the control action u is multiplied by the gain matrix K and sent to the second joint of the physical model.

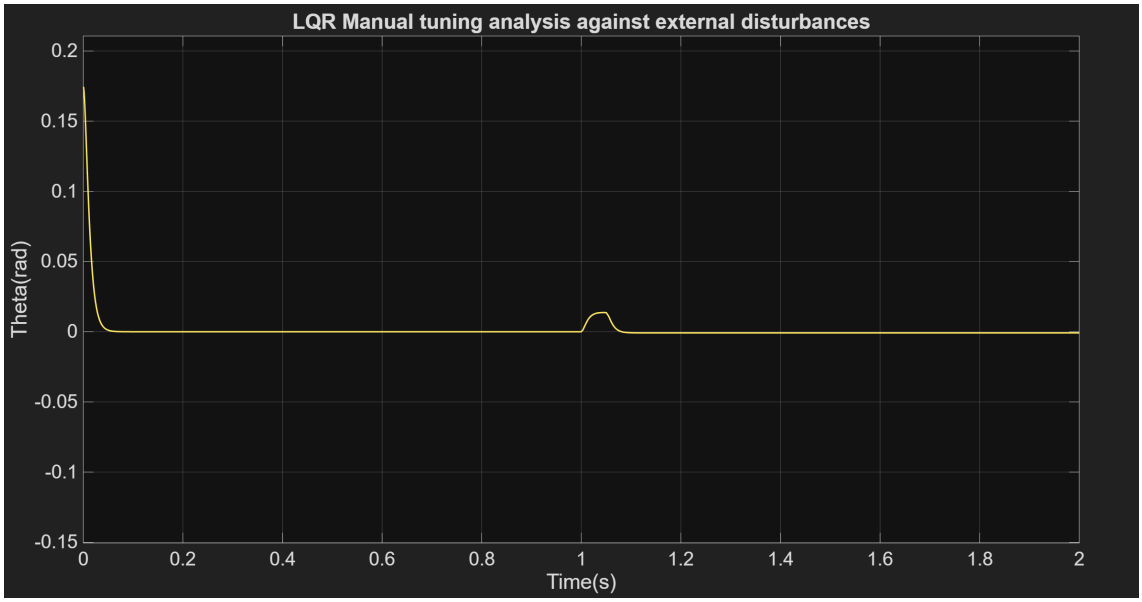


Figure 4.34: System response to disturbance torque with manually tuned controller.

As shown in Figure 4.34, the controller is able to stabilize the angle at an upright position after an oscillation.

4.4.5 Physical Implementation tests

As with the PD controller, the LQR gains were implemented on the physical self-balancing motorcycle to compare the simulation results with real-world performance. This controller proved highly successful. After minor adjustments to the theoretically computed gains, the self-balancing motorcycle was able to maintain stable balance.

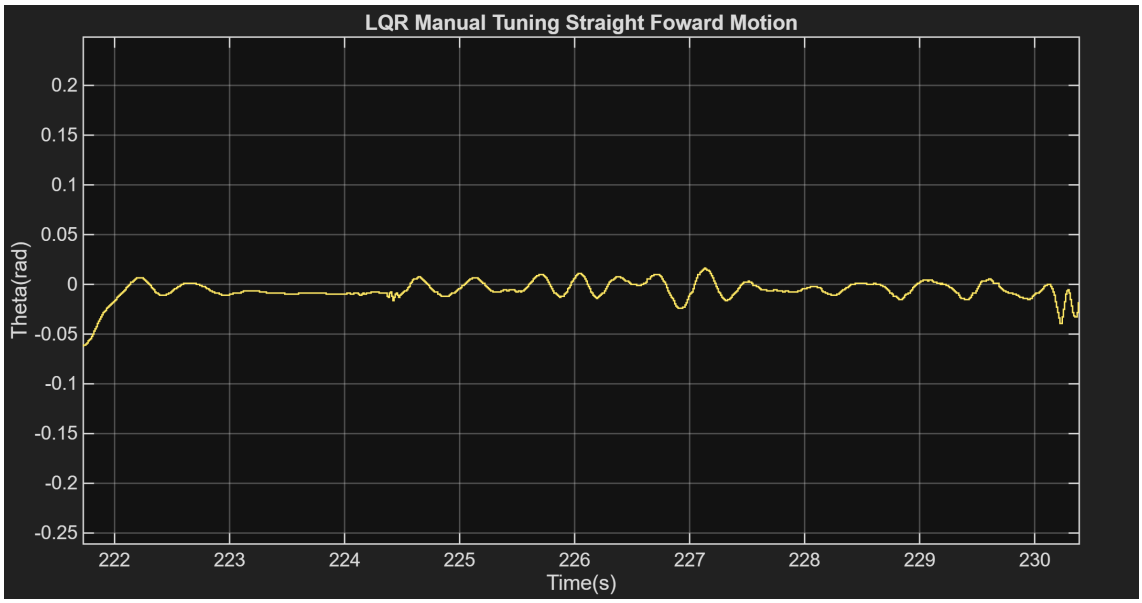


Figure 4.35: LQR Physical Implementation test

Because the LQR formulation explicitly regulates the inertial wheel angular velocity, the saturation problem observed in the PD implementation was effectively eliminated. Additional experiments were conducted to assess the robustness of the controller in practical scenarios, including balancing during straight-line motion, forward and backward movement.

In all tested cases, the self-balancing motorcycle remained stable, demonstrating the effectiveness and versatility of the LQR control strategy.

4.5 Sliding Mode Control

This section presents the design, implementation, and validation of a Sliding Mode Control (SMC) strategy for stabilizing the self-balancing motorcycle. The formulation includes sliding surface definition, reaching law design, and chattering mitigation through quasi-sliding mode control. Simulation results demonstrate fast stabilization, strong disturbance rejection, and improved practical feasibility when using saturation-based boundary layers.

4.5.1 Sliding Mode Control principles

Sliding Mode Control (SMC) addresses control of inherently unstable systems through enforcement of convergence to a user-defined sliding surface. SMC provides inherent robustness against parametric uncertainty and external disturbances, properties valuable for systems with model mismatch or unmodeled dynamics [24]. However, one disadvantage is chattering, where the system state jitters back and forth.

Sliding mode control tries to constrain the trajectories so that they can only exist in a line within the phase plane [12]. This constrained behavior is useful because the system now behaves like a first order linear system. The equation for this line can be written as:

$$s = bx_1 + x_2 = 0$$

$$x_2 = \dot{x_1}$$

$$\dot{x_1} = -bx_1$$

This equation is the switching function, s, and when s is set to zero, the sliding surface is obtained. The switching function can be any function of the system states, however, it is common to write it as a linear combination of states [10]:

$$s = f(x) = C^T x$$

- C is m_x by n_y,
- the number of rows of the matrix C is the number of states in the system,
- the number of columns is the number of inputs u into the system.

In order to design sliding mode control there are some things to consider. First, a sliding surface needs to be defined. After,a control scheme to reach the sliding surface needs to be found. Finally, it is required a method to deal with chattering.

Reaching Laws

An important concept in Sliding Mode Control (SMC) is the mechanism that drives the system states toward the sliding surface, typically defined as $s(x) = 0$. The dynamics of this convergence are defined by what is known as the reaching law [12].

The idea is that whenever the sliding variable s is positive, its time derivative $\dot{s}$ should be negative, and vice versa. This ensures that the states always move toward the sliding surface regardless of their initial position. In mathematical terms, this leads to the following reachability condition:

$$s \cdot \dot{s} < 0 \tag{4.23}$$

This condition implies that the sign of s must always be opposite to the sign of $\dot{s}$. A simple way to ensure this is by defining the reaching law as:

$$\dot{s} = -\eta \cdot \text{sign}(s) \tag{4.24}$$

This is known as the constant rate reaching law, where $\eta > 0$ is a positive gain that controls the rate at which the states approach the sliding surface. The parameter η plays a crucial role in the controller's behavior:

- A larger η results in faster convergence to the surface, increasing robustness to disturbances and model uncertainties.

- However, excessively large values of η may cause high-frequency oscillations known as chattering, which are undesirable in practical systems due to actuator limitations and energy loss.

Therefore, η represents a trade-off between robustness and smoothness.

There are many other variations of reaching laws designed for specific applications, like boundary-layer approximations that replace the discontinuous sign function with continuous alternatives (e.g., saturation or hyperbolic tangent). These formulations aim to maintain reachability while minimizing chattering in real implementations.

The time derivative of the sliding surface $s(x)$ can be computed using the chain rule [10] as:

$$\dot{s} = \frac{ds}{dx} \cdot \dot{x} \tag{4.25}$$

Given the previously defined sliding surface $s(x) = C^\top x$, its gradient with respect to the state vector is:

$$\frac{ds}{dx} = C^\top \tag{4.26}$$

For a general nonlinear system described by:

$$\dot{x} = f(x) + g(x)u(t) \tag{4.27}$$

where $f(x)$ represents the natural system dynamics and $g(x)$ describes how the input affects the system [10], the derivative of the sliding surface becomes:

$$\dot{s} = C^\top \left[f(x) + g(x)u(t) \right] \tag{4.28}$$

This expression defines how the sliding variable s evolves under the influence of the control input $u(t)$. To enforce a desired reaching law $\dot{s} = h(s(x))$, which ensures convergence to the sliding surface (e.g., constant or exponential laws), it can be said that:

$$C^\top \left[f(x) + g(x)u(t) \right] = h(s(x)) \tag{4.29}$$

Solving for the control input $u(t)$ yields the general sliding mode control law:

$$u(t) = \left(C^\top g(x) \right)^{-1} \left[-C^\top f(x) + h(s(x)) \right] \tag{4.30}$$

This expression is known as the equivalent control law with reaching law injection. It defines how the controller should act to bring the system state toward the sliding surface, while following the desired convergence behavior set by $h(s(x))$.

This formulation is very powerful because it can be applied to many nonlinear systems, and it explicitly separates the influence of the system dynamics from the shaping of the convergence through the reaching law.

It is important to note that in order to compute the control input $u(t)$ using the expression above, the controller must rely on an estimate of the system dynamics, namely $f(x)$ and $g(x)$. However, one of the strengths of Sliding Mode Control is its robustness. Even if the model of the system is not perfectly accurate, the control action still tends to drive the system toward the sliding surface.

This is because the reaching law is enforced with a high-gain discontinuous input, which is designed to overpower uncertainties, external disturbances, or unmodeled dynamics. As long as the model mismatch is bounded and the system satisfies certain reachability conditions, the controller maintains stability and convergence.

This robustness property makes Sliding Mode Control very appealing for systems subject to parameter variations or uncertain operating environments [11].

Chattering is undesirable in practical applications, as it can cause excessive wear on actuators and excite unmodeled high-frequency dynamics. In classical Sliding Mode Control (SMC), this condition arises because the control law enforces the sliding condition using a discontinuous switching function, typically based on the $\text{sign}(s)$ operator.

While this ideal discontinuous control guarantees convergence to the sliding surface in theory, it is not directly implementable in real systems due to actuator bandwidth limitations and measurement noise.

To address this limitation, several practical modifications of SMC have been proposed. One widely adopted approach consists in relaxing the requirement of exact sliding motion on the surface $s(x) = 0$, and instead allowing the system trajectories to evolve within a thin neighborhood around it.

This leads to what is commonly referred to as quasi-sliding mode behavior, in which the sliding condition is approximately satisfied within a bounded region rather than enforced exactly.

Quasi-Sliding Mode Control (QSMC)

In quasi-sliding mode control, the system is no longer required to remain exactly on the sliding surface, but instead within a predefined boundary layer surrounding it. This boundary layer defines a tolerance region around $s(x) = 0$ in which the system exhibits smooth dynamics while retaining the robustness properties of classical SMC.

A common way to achieve quasi-sliding motion is through the boundary layer method, which replaces the discontinuous switching function $\text{sign}(s)$ with a continuous approximation. One frequently used choice is the saturation function, defined as [10]:

$$\text{sat}(s,\phi) = \begin{cases} 1, & s > \phi \\ \frac{s}{\phi}, & -\phi \leq s \leq \phi \\ -1, & s < -\phi \end{cases} \tag{4.31}$$

Here, ϕ is referred to as the boundary layer thickness and determines the width of the region around the sliding surface in which the control law transitions smoothly instead of switching

abruptly. When $|s| \leq \phi$, the control action varies linearly with s, significantly reducing high-frequency switching and thus mitigating chattering.

Although the system no longer satisfies the ideal sliding condition $s(x) = 0$, the state remains confined within the boundary layer, preserving the main robustness advantages of sliding mode control while improving practical implementability.

The value of ϕ is very important in balancing performance and smoothness:

- A larger ϕ reduces chattering but can introduce steady-state error.
- A smaller ϕ improves accuracy but may reintroduce high-frequency switching.

4.5.2 SMC Implementation

The system is usually expressed in the form:

$$\dot{x} = f(x) + g(x)u, \tag{4.32}$$

where $f(x)$ represents the natural dynamics of the system and $g(x)$ describes how the control input affects it, which allows for defining:

$$f(x) = Ax, \qquad g(x) = B. \tag{4.33}$$

This system is then implemented in Simulink by feeding the state vector from the inverted pendulum subsystem into the SMC block. Since the dynamics are represented by the matrix A, a Matrix Multiply block is used to compute $f(x) = Ax$. A Constant block is used to define the matrix B, and both are connected to the corresponding inputs of the SMC block.

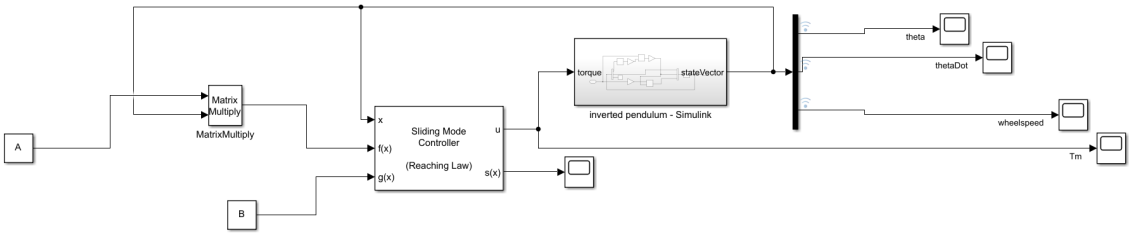


Figure 4.36: Sliding Mode Control implemented in Simulink

Unlike the linear quadratic regulator (LQR), which requires the plant to be linearized, sliding mode control (SMC) is inherently nonlinear and can be applied directly to nonlinear systems.

The SMC algorithm was implemented using the full nonlinear equations of motion rather than the linearized model. Specifically, the inverted pendulum model supplied with the Arduino Engineering Kit Rev2 was used in Simulink to represent the self-balancing motorcycle's dynamics.

In the corresponding block diagram, the input torque is fed through blocks that compute the nonlinear gravitational term (using the sine of the pendulum angle) and the coupling between the

pendulum and inertial wheel. These blocks generate the angular accelerations of the lean angle and the wheel, which are then integrated to produce the state variables—the lean angle, lean angular velocity and wheel speed.

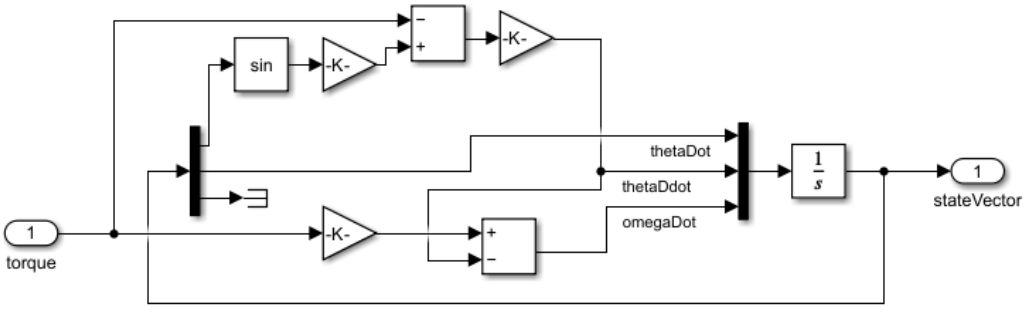


Figure 4.37: Nonlinear Inverted Pendulum model

Control Parameters

To control the system, the following SMC parameters were used:

- **Sliding Mode:** Regulation mode (to drive the states to zero).
- **Sliding Coefficient Matrix:** Since the aim is to control θ and $\dot{\theta}$, the sliding surface is defined as $s = c \cdot \theta + \dot{\theta}$, corresponding to matrix $[c, 1, 0]$.
- **Reaching Law:** Constant rate.
- **Boundary Layer:** Simulations were done using both sign and saturation functions to evaluate performance and reduce chattering.

After extensive simulation tuning, the sliding coefficient was set to $c = 100$, and the reaching rate to $\eta = 1000$. The initial lean angle was set to $\theta = 0.175$ rad. The system stabilizes quickly and effectively drives the angle to zero.

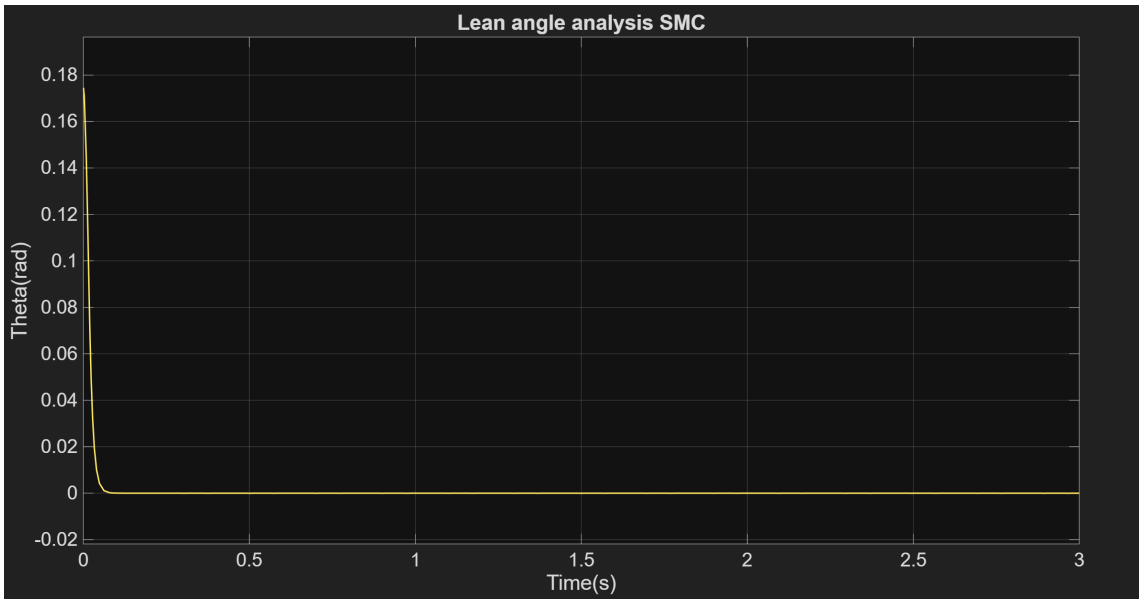


Figure 4.38: Sliding Mode Control response with sign boundary function and $c = 100, \eta = 1000$.

Table 4.13: Time-domain performance metrics for θ using SMC with sign function.

Metric	**Value**
Delay time	17 ms
Rise time	70 ms
Maximum percent overshoot	0%
Settling time	49.2 ms

Simulation with Sign Boundary Layer

It can be observed that the system experiences significant chattering. This behavior is due to the use of the `sign` function, which introduces high-frequency oscillations in the control input when the sliding surface is reached. By analyzing the switching function output, it is confirmed that chattering begins approximately at $t = 20ms$, which corresponds to the moment the sliding surface $s(x) = 0$ is reached.

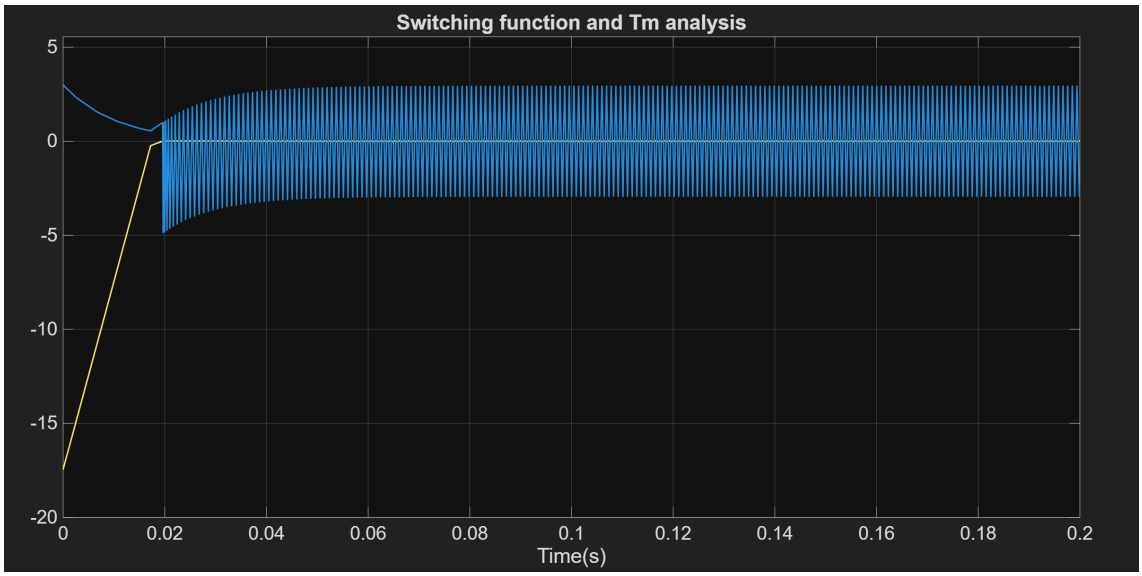


Figure 4.39: Switching function and Tm analysis

To mitigate this effect, the boundary layer method was applied by replacing the `sign` function with a saturation function. This creates a smooth transition zone around the sliding surface, significantly reducing chattering. In this simulation, the boundary layer width parameter ϕ was set to 4.

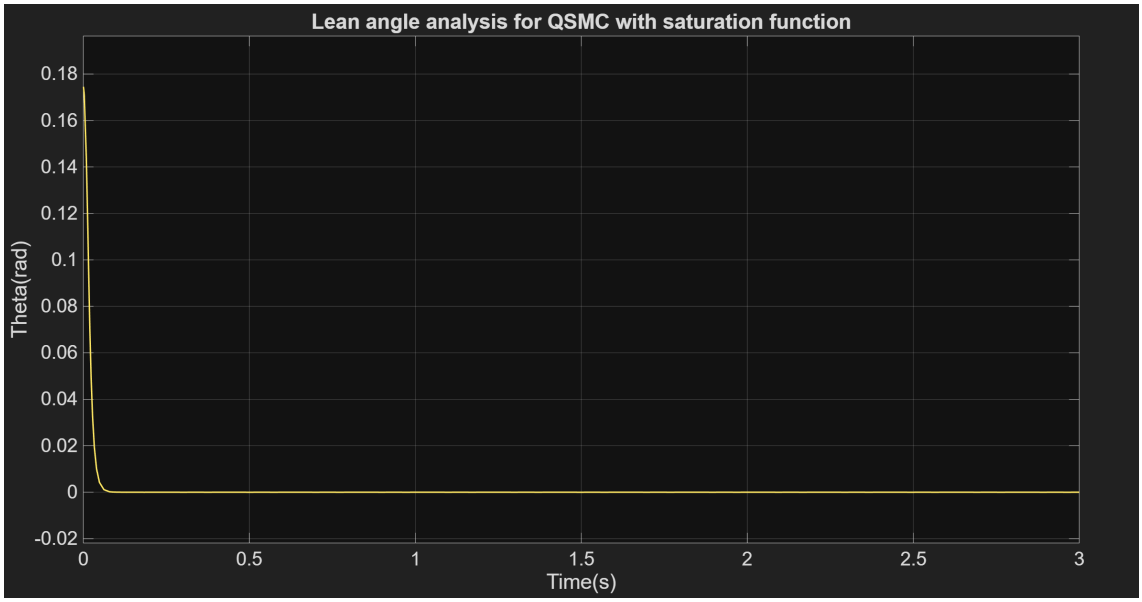


Figure 4.40: Quasi-Sliding Mode Control response with saturation boundary function.

Table 4.14: Time-domain performance metrics for θ using quasi sliding mode control with saturation boundary layer.

Metric	**Value**
Delay time	17.3 ms
Rise time	78 ms
Maximum percent overshoot	0%
Settling time	50.4 ms

As observed in the results, the controller is still capable of driving θ to zero effectively. However, this time the control action is smoother, and chattering is significantly reduced, resulting in a more realistic and implementable control signal for physical systems.

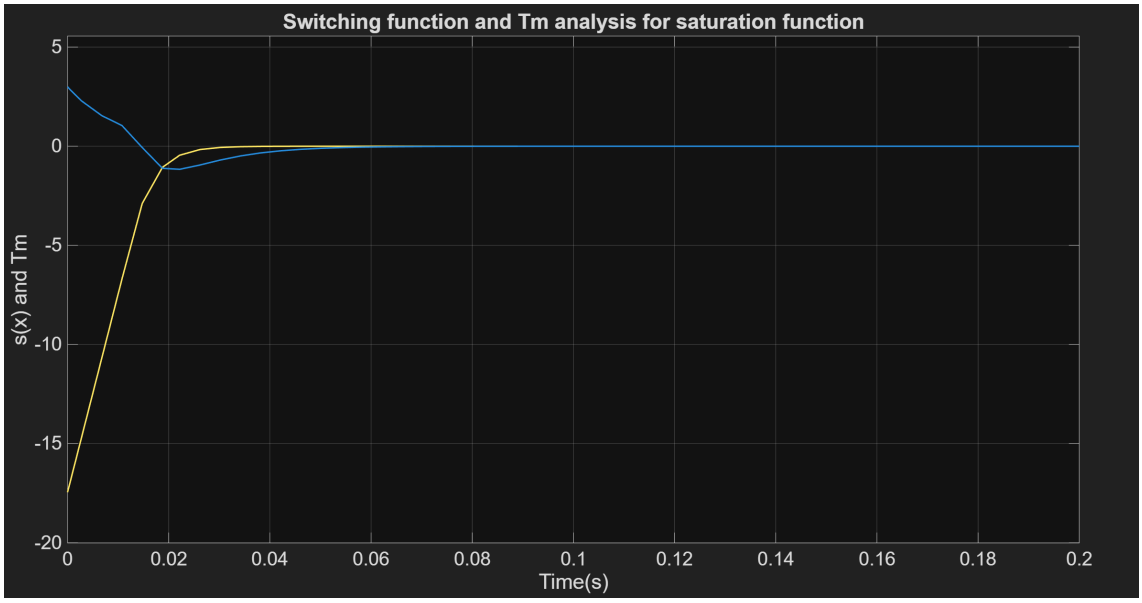


Figure 4.41: Sliding Mode Controller settings with saturation function enabled and $\phi = 4$.

SMC Disturbance analysis

Just like in the PD and LQR control, the SMC controller was tested using the Simscape alternative model and a disturbance was simulated every second. As shown in the figure below, the controller was able to regain balance after every disturbance.

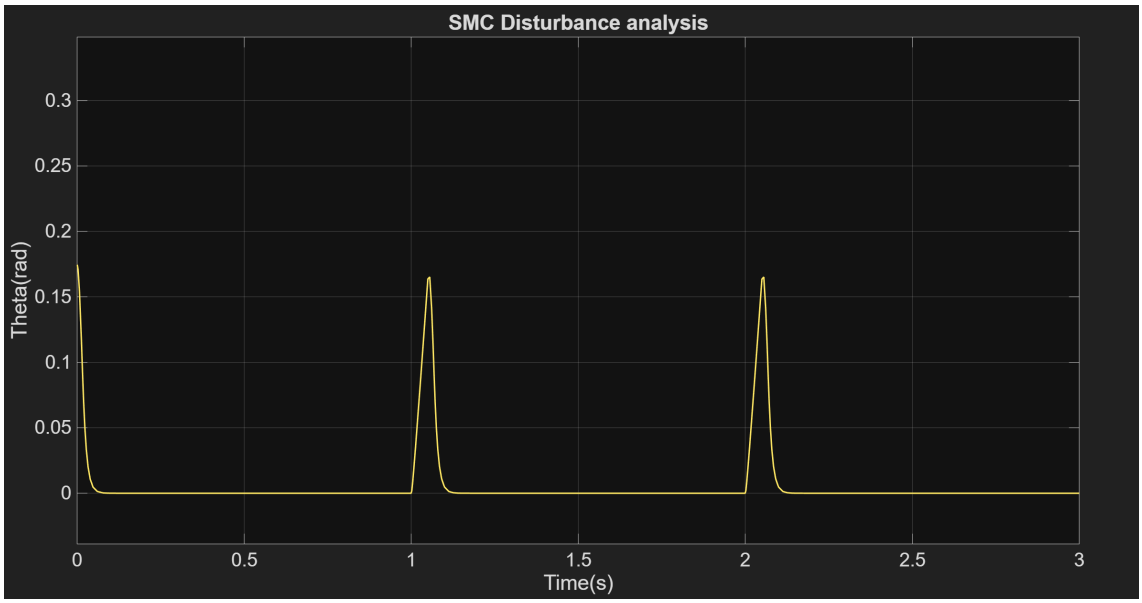


Figure 4.42: Sliding Mode Controller Disturbance analysis

Relevance to Future Work

The reference tracking feature of the Sliding Mode Controller is particularly promising for future extensions of this project. In the context of autonomous self-balancing motorcycles, lean angle control becomes important for maneuvering through curves. For wide-radius turns, the system can rely on the existing balancing control, as the natural dynamics are sufficient to stabilize the vehicle.

However, for tighter curves, such as those encountered in urban navigation or racing scenarios, an additional torque must be commanded to achieve the required lean angle. By using the reference tracking mode, the desired lean angle could be provided as a time-varying reference input x_{ref}, allowing the controller to adjust the inertial wheel torque dynamically and achieve stable turning behavior.

This capability would enable the self-balancing motorcycle to maintain balance while executing aggressive or sharp maneuvers, increasing its operating modes and real-world applicability.

Chapter 5

Results and Conclusions

The experimental results presented above are now synthesized to evaluate the performance of each controller. This chapter summarises the main results obtained in this work and draws conclusions on the effectiveness of the studied control strategies. The objective of the project was to stabilise a self-balancing motorcycle around the upright equilibrium using an inertial wheel actuator. A nonlinear dynamic model was obtained from first principles, converted into a state-space formulation, and linearised around the equilibrium in order to allow classical and optimal control design. The performance of each controller was evaluated in simulation and, for the PD and LQR controllers, on physical hardware.

5.1 Summary of Modelling Outcomes

A simplified inverted-pendulum model captures the three-state dynamics: lean angle θ, lean rate $\dot{\theta}$, and inertial wheel speed ω. The nonlinear gravitational term $\sin(\theta)$ was linearized about $\theta = 0$, valid for $|\theta| < 0.25\,\text{rad}$. Both Simulink and Simscape implementations were developed; cross-validation under identical initial conditions and control inputs demonstrated good agreement, thereby validating the lumped-parameter assumptions and the numerical correctness of both implementations.

5.2 Results for PD and PID Control

The first controllers implemented were PD and PID regulators due to their simplicity and their use in engineering applications. Both controllers were initially tuned using the Ziegler–Nichols method and then adjusted through manual tuning.

The Ziegler–Nichols control led to aggressive gains that produced fast initial responses but also introduced large overshoot and oscillatory behaviour. The corresponding pole–zero maps confirmed that these responses were consistent with lightly damped complex poles with large imaginary parts.

Manual tuning improved the transient behaviour of both PID and PD controllers by reducing oscillations and eliminating overshoot. Although the manually tuned controllers generally presented larger rise times than the Ziegler–Nichols designs, they achieved much shorter settling times and smoother convergence.

Among the four PID/PD configurations, the manually tuned PD controller provided the best overall simulated performance. Pole–zero analysis confirmed that the manual PD tuning moved the closed-loop poles onto the real axis, eliminating oscillatory modes and increasing damping.

Although manually tuned PD control demonstrated stable performance in simulation, physical implementation revealed a critical limitation: persistent sensor bias (non-zero lean angle measurements) generates residual control torque that progressively saturates the inertial wheel. The system maintained balance for approximately 10 seconds before saturation, beyond which stability was lost.

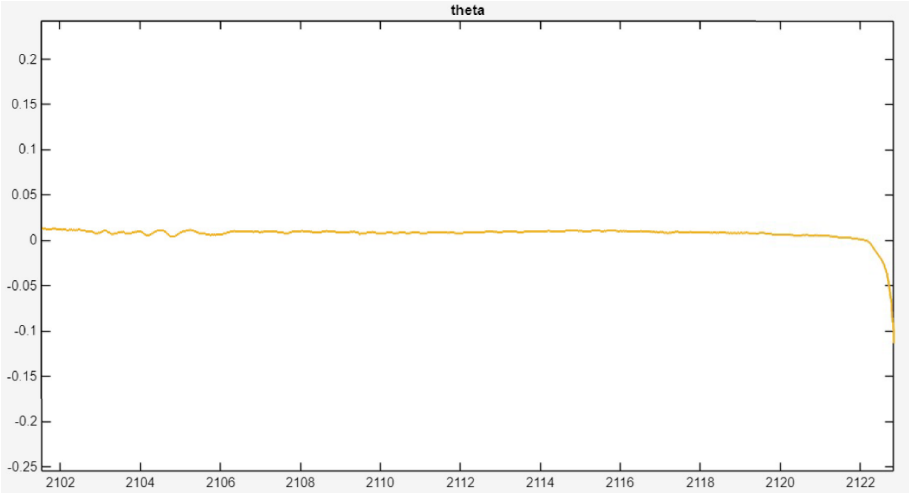


Figure 5.1: PD Control Physical Implementation

This practical constraint highlights the necessity of wheel-speed regulation, motivating the LQR approach.

5.3 Results for LQR Control

To overcome the limitations observed in PD/PID control, a Linear Quadratic Regulator (LQR) was designed based on the linearised state-space model. The LQR approach provides a method to compute a full-state feedback gain K that minimises a quadratic cost function. Three tuning strategies were evaluated: Bryson's rule, identity weighting, and manual tuning.

Bryson's rule produced a conservative controller that respected predefined state and input limits. This tuning presented a stable response with limited overshoot, but a long settling time. Identity weighting increased responsiveness and reduced the settling time, but at the cost of increased overshoot, indicating reduced damping.

Finally, manual adjustment of the weighting matrices led to the best overall response: the controller achieved a fast transient, eliminated overshoot, and reduced settling time substantially. These results demonstrate that although Bryson's rule provides a physically grounded initial tuning, further adjustments are required to achieve high performance in practical situations.

A big advantage of LQR in this application is that the controller regulates the inertial wheel speed state. As a result, the wheel saturation issue in PD physical tests was eliminated. Physical experiments confirmed the robustness of the LQR controller: after minor adjustments to the theoretically computed gains, the self-balancing motorcycle was able to maintain balance reliably.

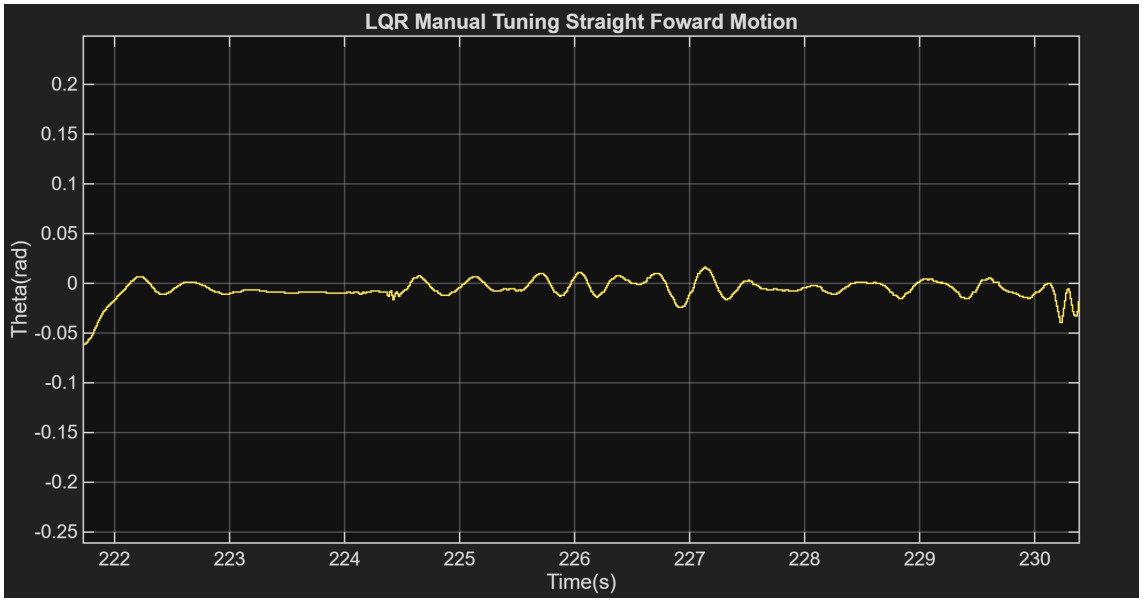


Figure 5.2: LQR Physical Implementation test

Further testing demonstrated stable behaviour not only in stationary balancing but also during straight-line motion, forward and backward movement, and small-radius turning manoeuvres. These tests indicate that LQR provides a practical and robust balance controller for the self-balancing motorcycle, outperforming the PD/PID approaches in real implementation.

5.4 Results for Sliding Mode Control

Sliding Mode Control (SMC) was explored as a nonlinear control alternative due to its known robustness to disturbances and modelling uncertainty. The controller was implemented using a linear sliding surface of the form $s = c\theta + \dot{\theta}$ and a constant-rate reaching law.

Simulations showed that the sign-based switching implementation was able to drive θ to zero quickly, however, the discontinuity in the control action generated chattering when the states approached the sliding surface. This effect was clearly observed both in the control signal and in the switching function.

To mitigate chattering, a quasi-sliding mode approach was adopted by replacing the discontinuous sign function with a saturation function and introducing a boundary layer. This modification reduced high-frequency switching while maintaining performance.

The resulting control action was more realistic for physical implementation. Although physical tests were not performed for SMC in this work, the simulation results indicate that SMC is promising for future extensions, particularly for reference tracking applications such as commanding non-zero lean angles for turning manoeuvres.

5.5 Overall Comparison and Main Conclusions

Table 5.1 summarises the most relevant time-domain performance metrics obtained in simulation for the manually tuned PD, LQR, and quasi-sliding mode controllers. All three controllers achieved zero overshoot, indicating adequate damping and successful stabilisation around the upright equilibrium.

Table 5.1: Comparison of time-domain performance metrics for manually tuned controllers (simulation).

Metric	**PD (Manual)**	**LQR (Manual)**	**SMC (Quasi)**
Delay time t_d [ms]	12.3	13.0	17.3
Rise time t_r [ms]	92.9	50.0	78.0
Maximum overshoot M_p [%]	0	0	0
Settling time t_s [ms]	64.3	40.0	50.4

The manually tuned PD controller exhibited the slowest transient response, with the largest rise time and a relatively longer settling time. While its behaviour is smooth and non-oscillatory in simulation, the absence of explicit regulation of the inertial wheel speed limits its practical applicability, as confirmed by physical experiments.

The manually tuned LQR controller achieved the fastest overall response, presenting the shortest rise time and settling time while maintaining zero overshoot. This behaviour reflects the optimal nature of the LQR formulation, which explicitly penalises both state deviations and actuator dynamics. These advantages were confirmed experimentally, where LQR demonstrated reliable balancing performance under a range of operating conditions.

The quasi-sliding mode controller demonstrated competitive transient performance in simulation, achieving zero overshoot and settling times comparable to those of LQR. Although its rise and settling times are slightly longer in the simulated results, Sliding Mode Control was not validated through physical experimentation in this work. Consequently, no definitive conclusions can be drawn regarding its real-world performance.

Given its inherent robustness to disturbances, modelling uncertainties, and sensor imperfections, SMC remains a strong candidate for practical implementation. It is plausible that, when

properly tuned and implemented in hardware, Sliding Mode Control could match or even outperform linear control strategies in real-world conditions. This makes SMC a promising direction for future experimental work, particularly in applications involving non-ideal sensing, external disturbances, or reference tracking.

In conclusion, this thesis demonstrates that a self-balancing motorcycle can be stabilised using feedback control based on a simplified inverted-pendulum model.

While PD/PID control can stabilise the system in simulation, LQR offers the most reliable and robust performance in real hardware by accounting for actuator dynamics and regulating wheel speed. Sliding Mode Control is a promising nonlinear alternative for future work, especially in scenarios requiring robustness and reference tracking.

5.6 Future Work

Although the controllers developed in this thesis successfully stabilised the self-balancing motorcycle, there are several avenues for further research and development.

First, turning dynamics need investigation. When navigating curves, the self-balancing motorcycle experiences an additional moment about the wheel–ground axis. For large-radius turns this moment is negligible and balance can be maintained using only the inertial wheel, but for tight turns it becomes comparable to the balancing torque.

Incorporating this "turning" torque into the control law would allow the self-balancing motorcycle to lean inward during small-radius manoeuvres.

Second, the linear quadratic regulator (LQR) could be extended to a Linear Quadratic Gaussian (LQG) controller by integrating a Kalman filter. This would estimate the full state vector from noisy sensor data and could improve performance and robustness to disturbances. Similarly, the sliding-mode controller developed here was only validated in simulation.

Implementing this nonlinear controller on the physical self-balancing motorcycle and tuning it to minimise chattering effects would be an important next step. The reference-tracking capability of SMC could also be used to command non-zero lean angles, allowing leaning during tight curves.

Third, other advanced nonlinear control strategies can be explored. Model Predictive Control (MPC), for instance, can handle input and state constraints explicitly and may provide better performance for highly nonlinear systems.

Finally, hardware improvements could be done. Adding proximity sensors would enable object-avoidance capabilities, and upgrading the DC motor and sensing hardware would allow higher-speed tests and more aggressive manoeuvres.

These directions would build on the current work and bring the self-balancing motorcycle closer to practical deployment.

References

[1] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[2] *ARDUINO Microcontrolador Arduino Nano 33 IoT*. pt. URL: https://mauser.pt/catalog/product_info.php?products_id=096-7553 (visited on 07/27/2025).

[3] Sunu Babu and Anju Pillai. "Design and Implementation of Two-Wheeled Self-Balancing Vehicle Using Accelerometer and Fuzzy Logic". In: Jan. 2016, pp. 45–53. ISBN: 978-81-322-2525-6. DOI: 10.1007/978-81-322-2526-3_6.

[4] R. Bars. *Control Engineering*. Advanced Textbooks in Control and Signal Processing. Springer Singapore, 2019. DOI: 10.1007/978-981-10-8297-9.

[5] Johannes Betz et al. "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing". en. In: *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 458–488. ISSN: 2687-7813. DOI: 10.1109/OJITS.2022.3181510. URL: https://ieeexplore.ieee.org/document/9790832/ (visited on 10/30/2024).

[6] Yüksel Ediz Bezci et al. "Classical and intelligent methods in model extraction and stabilization of a dual-axis reaction wheel pendulum: A comparative study". In: *Results in Engineering* 16 (2022), p. 100685. ISSN: 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2022.100685. URL: https://www.sciencedirect.com/science/article/pii/S2590123022003553.

[7] Daniel Boley and Biswa Nath Datta. "LINEAR CONTROL SYSTEMS". en. In: ().

[8] Delft University of Technology. *Control Theory*. https://www.aerostudents.com/courses/control-theory/controlTheoryFullVersion.pdf. Accessed: 2025-02-03. 2021.

[9] Department of Elect/Elect Engineering, Nile University of Nigeria et al. "Design of an Optimal Linear Quadratic Regulator (LQR) Controller for the Ball-On-Sphere System". In: *International Journal of Engineering and Manufacturing* 10.3 (June 8, 2020), pp. 56–70. ISSN: 23053631, 23065982. DOI: 10.5815/ijem.2020.03.05. URL: http://www.mecs-press.org/ijem/ijem-v10-n3/v10n3-5.html (visited on 02/19/2025).

[10] Nabil Derbel, Jawhar Ghommam, and Quanmin Zhu. *Applications of Sliding Mode Control*. Vol. 79. Jan. 2017. ISBN: 978-981-10-2373-6. DOI: 10.1007/978-981-10-2374-3.

[11] Nahid Ebrahimi, Sadjaad Ozgoli, and Amin Ramezani. "Model-free sliding mode control, theory and application". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 232 (June 2018), p. 095965181878059. DOI: 10.1177/0959651818780597.

[12] C. Edwards and S. Spurgeon. *Sliding Mode Control: Theory And Applications*. Series in Systems and Control. Taylor & Francis, 1998. ISBN: 978-0-7484-0601-2. URL: https://books.google.pt/books?id=uH2RJhIPsiYC.

[13] *Engineering Kit!* URL: https://engineeringkit.arduino.cc/aekr2/module/engineering/lesson/03-introduction-to-mechatronics (visited on 09/27/2025).

[14] Pallav Gogoi et al. "Design and Fabrication of Self Balancing Two Wheeler Vehicle Using Gyroscope". In: *International Journal of Engineering and Technology* 9 (June 2017), pp. 2051–2058. DOI: 10.21817/ijet/2017/v9i3/1709030206.

[15] Jesse B. Hoagg and Dennis S. Bernstein. "Nonminimum-phase zeros - much to do about nothing - classical control - revisited part II". In: *IEEE Control Systems Magazine* 27.3 (2007), pp. 45–57. DOI: 10.1109/MCS.2007.365003.

[16] Kiattisin Kanjanawanishkul. "LQR and MPC controller design and comparison for a stationary self-balancing bicycle robot with a reaction wheel". In: *Kybernetika* 54 (Mar. 2015), pp. 173–191. DOI: 10.14736/kyb-2015-1-0173.

[17] Martin. *Communications materials*. en-US. URL: https://www.un.org/sustainabledevelopment/news/communications-material/ (visited on 07/19/2025).

[18] Hla Myo Tun et al. "Research on Self-balancing Two Wheels Mobile Robot Control System Analysis". In: *Electrical Science Engineering* 4 (Apr. 2022), pp. 7–20. DOI: 10.30564/ese.v4i1.4398.

[19] *Nano 33 IoT | Arduino Documentation*. URL: https://docs.arduino.cc/hardware/nano-33-iot/ (visited on 07/22/2025).

[20] Katsuhiko Ogata. *Modern control engineering*. en. 5th ed. OCLC: 1492322544. Prentice Hall, 2022. ISBN: 978-0-13-615673-4.

[21] *Self-Balancing Motor Cycle Using Arduino Engineering Kit Rev 2 - MATLAB & Simulink*. URL: https://www.mathworks.com/help/simulink/supportpkg/arduino_ug/aek-selfbalancing-motorcycle-example.html (visited on 10/30/2024).

[22] I. I. Siller-Alcalá et al. "Using a Flywheel to Stabilize a Self-Balancing Bicycle". In: *WSEAS Transactions on Systems and Control* 19 (2024). Licensed under CC BY 4.0, pp. 73–84. ISSN: 2224-2856. DOI: 10.37394/23203.2024.19.8. URL: https://wseas.com/journals/control/2024/a165103-006.pdf.

[23] Tang Teng Fong et al. "Design and Analysis of Linear Quadratic Regulator for a Non-Linear Positioning System". In: *Applied Mechanics and Materials* 761 (May 18, 2015), pp. 227–232. ISSN: 1662-7482. DOI: 10.4028/www.scientific.net/AMM.761.227. URL: https://www.scientific.net/AMM.761.227 (visited on 02/19/2025).

[24] Hoang-Chinh Tran et al. "Simulation and Experiment of Sliding Control for Reaction Wheeled Inverted Pendulum". In: *Robotica & Management* 25.1 (2020). ISSN inferred from Robotica & Management series, pp. 33–37. ISSN: 1221-4590.

[25] University of Cambridge, Department of Engineering. *Lecture 17 - PID controller (notes)*. https://www.eng.cam.ac.uk/. Lecture slides on PID control fundamentals. 2021. (Visited on 08/16/2025).

[26] Arturo Urquizo. *English: PID controller overview*. Dec. 2011. URL: https://commons.wikimedia.org/wiki/File:PID_en.svg (visited on 07/27/2025).

[27] Damjan Vasilevski, Damjan Pecioski, and Simona Domazetovska Markovska. "DESIGN AND IMPLEMENTATION OF SELF-BALANCING MOTORCYCLE". en. In: *Mechanical Engieneering-Scientific Journal* 41.2 (2023), pp. 109–113. ISSN: 18575293, 18579191. DOI: 10.55302/MESJ23412670109v. URL: https://www.mesj.ukim.edu.mk/journals/article/view/106 (visited on 12/18/2024).

[28] Zhaoqiang Wang et al. "Improved Linear Quadratic Regulator Lateral Path Tracking Approach Based on a Real-Time Updated Algorithm with Fuzzy Control and Cosine Similarity for Autonomous Vehicles". In: *Electronics* 11.22 (Nov. 11, 2022), p. 3703. ISSN: 2079-9292. DOI: 10.3390/electronics11223703. URL: https://www.mdpi.com/2079-9292/11/22/3703 (visited on 02/19/2025).