FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Controlo de trajetórias com desvio de obstáculos de um veículo de locomoção híbrida

Inês Margarida Nicolau Soares

Mestrado Integrado em Bioengenharia

Supervisor: Paulo José Cerqueira Gomes da Costa

Co-supervisor: Vítor Hugo Machado Oliveira Pinto

July 30, 2021

Controlo de trajetórias com desvio de obstáculos de um veículo de locomoção híbrida

---

Trajectory control and obstacle avoidance in a vehicle with hybrid locomotion

Inês Margarida Nicolau Soares

Mestrado Integrado em Bioengenharia

July 30, 2021

Abstract

The burden to healthcare systems is being intensified as the prevalence of chronic diseases grows due to an increasingly elderly population. The current technological advancements and social values have prompted the expansion of automated applications. Therefore, automation of processes in the medical field, namely the adoption of robotic applications, can be implemented to channel healthcare personnel to high-value tasks. Robots can aid in improving care quality and ensuring safety in healthcare facilities. The specificities of these environments require robotic platforms with high manoeuvrability capabilities to allow, for example, autonomous operation between floors. A legged-wheeled robot is thus suitable for these facilities as their hybrid locomotion allows operation in different types of terrains with highly varying characteristics, both indoors and outdoors. Legged-wheeled robots present increased mobility, versatility and adaptability, when compared to vehicles using only one of the mechanisms. Since safety is a critical priority in the development of robotic systems for medical settings, simulation tools, which have a central role in robotics itself, can be used to create the environment the robot will operate in and evaluate its performance with realism, guaranteeing accurate execution of its goals. The developed work focused on designing a 3D realistic simulation model of a quadrupedal legged-wheeled robot combining both rigid and non-rigid joints. This model was fully tested and validated, and further used to develop a trajectory tracking method for the robot. Building a simulated inpatient unit allowed assessing the viability of the implemented algorithm.

Resumo

O aumento da população idosa a nível mundial acentua a prevalência de doenças crónicas, colocando sobre pressão crescente os sistemas de saúde. Ao longo dos últimos anos, os avanços tecnológicos e os valores sociais que têm emergido, levaram à expansão de sistemas automatizados. Assim, a automação de processos na área médica, nomeadamente a adoção de soluções robóticas, permite aos profissionais de saúde dedicarem-se a tarefas centradas no cuidado do doente. Os robôs demonstram contribuir para a melhoria dos cuidados de saúde e garantir que as instituições de saúde se tornam cada vez mais um local seguro. As especificidades associadas aos ambientes hospitalares exigem que as plataformas robóticas adotadas tenham inerente capacidade de manobra. A locomoção híbrida de veículos baseados em patas e rodas torna-os adequados para navegação neste tipo de meios, uma vez que facilita o movimento em diferentes tipos de terreno com caracterísiticas muito variáveis, tanto no interior como no exterior. Robôs com patas e rodas apresentam melhor mobilidade, versatilidade e adaptabilidade, quando comparados com veículos que incorporam isoladamente um dos mecanismos de locomoção. Dado que a segurança é criticamente prioritária no desenvolvimento de plataformas robóticas para os cuidados de saúde, a simulação, que, por si só, tem já um papel central em robótica, pode ser usada para recriar o ambiente no qual o robô opera, avaliando o seu desempenho de forma realista e garantindo que executa as tarefas com precisão. O trabalho desenvolvido incluiu a construção de um modelo realista 3D, em simulação, de um veículo com rodas e quatro patas, que combinam juntas rígidas e não-rígidas. Este modelo foi testado e validado no simulador, de modo a ser usado para desenvolver algoritmos de seguimento de trajetórias para o robô. Uma unidade de internamento foi desenhada no simulador para avaliar a viabilidade da estratégia desenvolvida.

Acknowledgements

Although the moment is of farewell as these five years approach an end, I could not conclude this phase without expressing my gratitude to the people that have supported me along this journey, especially to those who directly contributed to the success of this work.

First, I would like to thank Professor Paulo Costa, for his guidance and dedication. For the never-ending willingness in sharing his vast knowledge. For stimulating my growth and challenging me to get out of my comfort zone, widening my experience and skills. For his patience during the long Zoom meetings. And for showing and sharing his passion, letting me become even more mesmerised with robotics.

To Professor Vítor Pinto, for his constant support and availability. For the encouragement and teamwork. For always showing me his confidence, making me believe that it is always possible. And for his example of persistence.

To Professor José Lima, for helping me to take the first step that may allow pursuing a career as a researcher.

To my colleagues and specially to my friends, with whom I have shared the ups and downs of this road, and for the memories that I will keep for a lifetime.

Lastly, to my family. To my parents, for all the efforts you have made to provide me with the best education and opportunities, and to allow me to dream and follow my passions. To my sister, for the complicity and for always giving me the strength to carry on. You all are my permanent support and I can always count on you.

Inês Soares

“ Science is more than a body of knowledge.
It’s a way of thinking.”

— Carl Sagan

Contents

List of Figures

List of Tables

Abbreviations and Symbols

COVID-19	Coronavirus Disease 2019
DOF	Degrees of freedom
L	Length
W	Width
H	Height
3D	Three-Dimensional
LiDAR	Light Detection And Ranging
IMU	Inertial Measurement Unit
DC	Direct Current
CAD	Computer-Aided Design
PID	Proportional-Integral-Derivative
UDP	User Datagram Protocol
XML	Extensible Markup Language
IDE	Integrated Development Environment
GUI	Graphical User Interface

Chapter 1

Introduction

Alongside a brief overview on mobile robotics, and its current role in healthcare, this chapter introduces the work developed.

1.1 Background and Motivation

In a world where it is expected that the number of people aged 65 years or over more than doubles between 2019 and 2050 [1], improving life quality of this ageing population becomes a paramount challenge in the years to come. Therefore, the role of technological innovation in healthcare will undoubtedly be reinforced.

As autonomous systems, and in particular autonomous robotic systems, brace and accelerate these new needs and demands, the medical care paradigm is being transformed.

In the following sections, mobile robots are characterised and their integration in healthcare is highlighted.

1.1.1 Mobile Robotics

Anchored in a broad range of disciplines, robotics aims at developing machines capable of performing tasks to help and assist human activities or even replace humans [2].

Mobile robots have the ability to move autonomously, that is, they can move freely within their environment without human influence [2, 3]. Some are already being commercialised, being developed for a variety of applications across multiple areas, such as industry, exploration, inspection of hazardous environments, personal and domestic services, and medical care [2, 3].

Ensuring correct autonomous operation requires accurate integration of the fundamental systems of a robot. These comprise the locomotion, perception, localisation, navigation, and control systems (Figure 1.1) [3].

Locomotion is a critical issue in robot design that must consider kinematics, dynamics, and control theory [3]. The main locomotion types are based on ground, aquatic and aerial locomotion; ground robots use wheels, legs, tracks or a combination of these (hybrid locomotion) [3].

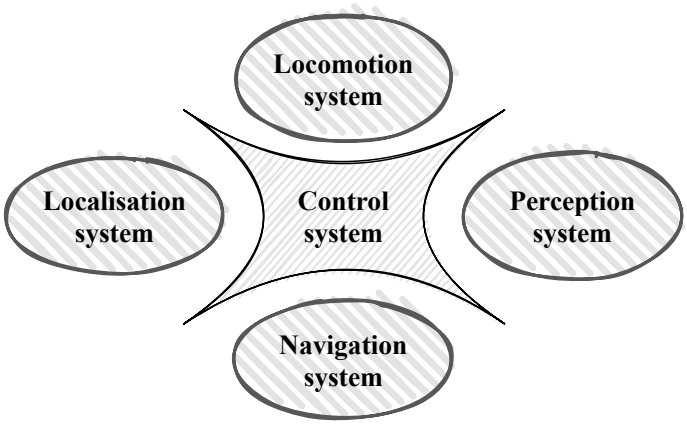


Figure 1.1: Fundamental systems of a mobile robot. The control system is responsible for integrating all the remaining systems.

Robots can have the ability of perceiving the surrounding environment as well as sensing and estimating their own state. Collecting data from sensors and processing it to retrieve meaningful information, and consequently determine the actions to be performed, are crucial tasks in autonomous operation.

Navigation is a fundamental competence of mobile robots. The navigation system mainly relies on perception, localisation (determining the robot's position), cognition (decision-making to achieve the objectives), and motion control (define the motor signals) [3]. The navigation process is realised from information on the robot's location, trajectory planning and obstacle avoidance [3].

Localising a robot might not only include knowing its position within the workspace (relative position in respect to a target), but also its absolute position on Earth [4]. It also requires, to allow mapping the robot's position, the generation of a map of the world [3].

Motion planning – a comprehensive concept comprising both path and trajectory planning, and trajectory tracking – relates to the computation of a trajectory to reach a target position [2, 3]. Path planning and trajectory planning both aim at defining how a robot gets to its destination [3]. The difference between the two is associated to the fact that the latter considers the temporal evolution of the motion and the forces required to achieve that motion [3]. Trajectory tracking enables knowing how a robot moves around its workspace [3].

Obstacle avoidance algorithms involve obstacle detection and obstacle avoidance itself. Collisions can be avoided with the redefinition of the robot's trajectory or even stopping it [3].

Data acquired from robot sensors is processed to be used in the cognition unit. Consistent choices on how to act, and what to do to meet a given purpose, are achieved with a decision-making process that accomplishes and executes high-level objectives [3].

Successful operation of a mobile robot thus requires an architecture that integrates different systems and complex interactions between them, involving knowledge from several scientific fields. This control architecture can be primarily divided into two components: low-level control and high-level control. The first is responsible for the control of the actuation system of the robot. It relies on the acquisition of data related to each motor in order to establish the control signals for

motion of the robot. High-level control is associated with the decision-making processes needed to perform the tasks that result in the robot achieving its goals. Other architectures can be considered for the control system, namely the inclusion of additional hierarchical levels.

1.1.2 Mobile Robotics and Healthcare

Robotics in healthcare is still centred on surgical robots (robotic arms/manipulators). Nevertheless, and although mobile robots are still not mainstreamed across a diversity of sectors, they are starting to be used in medicine as their benefits for patient care and hospital management are realised.

The COVID-19 outbreak has also contributed with use cases for the implementation of mobile robots in medical facilities. During this pandemic, robotic solutions, from disinfection and cleaning robots to hand sanitiser-dispensing and food delivery robots, have been widely adopted [5].

Mobile robots for the healthcare industry are considered *service robots*, as they are developed to assist patients and healthcare personnel [6]. Commercially available solutions can be categorised as (a) indoor transportation robots (namely, courier systems for delivery of medications, laboratory specimens and meals), (b) outdoor delivery robots (usually, unmanned aerial robots), (c) social robots (acting as receptionists to inform and guide patients and visitors, or aiding in nursing to provide therapeutic assistance), and (d) cleaning and disinfection robots.

Since awareness of the advantages that mobile robots can bring to healthcare is rising, the medical robotics market will keep its growing tendency. These systems can redefine current practices within healthcare organisations as they become faster, more reliable and their performance is improve.

Robots do not fall sick, do not get tired and can operate in contaminated spaces, which means they are highly cost-effective and sustainable [6]. However, co-operation with humans is a critical issue that needs to be considered when conceptualising a robot targeted for medical purposes. Thus, safety has to be always prioritised and the design and functional specifications must be validated with the potential users. The system needs to incorporate manual control and must be quiet, low maintenance, robust enough to comply with long-term use, and have a user-friendly interface [6]. Autonomous operation also requires careful selection of the power source to ensure long runtimes with high efficiency [6]. Moreover, the robots need to be equipped with a powerful framework for collision-free pathways and for replanning and redefinition of their objectives, because healthcare facilities are highly dynamic and have low predictability.

Therefore, mobile robots allow increased time for high-value tasks (focused on patient care), relieving health professionals from routine tasks or exposure to dangerous substances, and improve efficiency and safety of healthcare facilities while reducing health expenditures.

1.2 Objectives

The main goal of the developed work is to set up a control architecture for the operation of a hybrid legged-wheeled robot. Intermediate objectives include:

- Development and validation of the simulation model of the robot in the SimTwo simulator.
- Development and implementation of strategies for trajectory control in the SimTwo simulator.
- Case study: simulate robot operation in a hospital environment.

1.3 Structure

Four more chapters follow this introductory chapter. In Chapter 2, the current development landscape of legged-wheeled robots is given and some robotics simulators are briefly described, along with some applications. Chapter 3 presents the hybrid legged-wheeled robotic vehicle in which the Dissertation has been based. Chapter 4 focuses on the work developed: the simulation model of the robot is presented, the control structure described, an approach for trajectory tracking proposed, and, finally, results regarding the developed algorithms are reported. The final chapter (Chapter 5) concludes the Dissertation and presents suggestions for future work.

Chapter 2

Literature Review

Based on a review of the literature, this chapter focuses on the state of the art of hybrid vehicles combining legged and wheeled locomotion. The importance of simulation in robotics is also emphasised, along with the description of some simulators and their applications.

2.1 Introduction

Locomotion mechanisms have been constantly refined throughout the years. Designing hybrid locomotion systems has gained increasing focus in research, as they are able to merge the strengths of each embedded mechanism and outperform their disadvantages.

Wheeled robots face multiple problems when moving on uneven surfaces due to limitations related to wheel geometry (difficulty in passing over obstacles larger than the wheel radius) and the low slip ratio of wheels on steep surfaces. Unlike legged robots, they require simple control mechanisms and are inherently stable (the design ensures wheel contact with the ground at all times). The number of legs in a robot is a critical aspect for stability. Multi-legged robots, such as quadrupeds (four-legged) and hexapods (six-legged), besides their ability to move omnidirectionally and their robustness against external perturbations, have better balancing stability, when compared to bipedal robots, due to their larger support polygon[1].

Therefore, locomotion based on legs and wheels combines the efficiency of wheels with the enhanced obstacle negotiation capability of legs. Legged-wheeled robots thus show better mobility performance in both rough and smooth terrains, as legs easily adapt to uneven surfaces and wheels can achieve higher speed and lower power consumption on flat ones. Section 2.2 describes examples, found in the literature, of these types of robots.

Since robots are quite complex systems, simulation is an essential tool in robotics. Development of realistic simulation models of robotic systems represents an extremely important stage for refinement of a robot, as precise operation can only be assured by finely tuning the variables related to control strategies, dynamics and the robot mechanics itself. Therefore, realistic models not

[1]The support polygon refers to the convex hull of the expected trajectories of the foot-ground contact [7]

only significantly decrease the probability of a robot suffering unforeseen damage, but also reduce development time by speeding up the testing of control algorithms.

Robot simulators are also valuable for testing purposes due to allowing modelling a variety of situations for assessment of the performance of the robot, while avoiding harming the real robot. Realistic simulations and models can thus ensure successful operation in real world environments.

Section 2.3 is dedicated to the description of some simulators for robotics and examples of applications of simulation in the development of legged-wheeled robots.

2.2 Hybrid Vehicles with Legs and Wheels

Hybrid robots combining wheeled and legged locomotion that have been developed for the past decades fall, according to how legs and wheels interact with one another to achieve motion, into the following categories: (a) robots with separate wheels and legs, (b) legged-wheeled robots, (c) rotary-legged robots, and (d) transformable wheel-legged robots. Examples are given in the following sections.

2.2.1 Robots with Separate Wheels and Legs

Robots belonging to this category realise motion through coordination between wheels and legs. This seems to have been the initial approach to hybrid locomotion.

In 1993, Eiji *et al.* [8] described Chariot I and Chariot II. The former has two large wheels, attached to the right and left sides of the robot body, and two legs, one at the front and the other at the rear end. Chariot II [9] and Chariot III [10] (Figure 2.1a) both have four legs (3 DOF – Degrees of Freedom – per leg) and two wheels, arranged in a similar way to that of Chariot I. The wheels are installed with a passive suspension mechanism and each joint has an encoder attached to read the position of each foot.

Wheeleg [11] (Figure 2.1b), which weighs 25 kg and has dimensions of 1.110×0.660×0.400 m (L×W×H), has two pneumatically actuated front legs (3 DOF each) and two actuated rear wheels (to carry most of the robot's weight). Each one of its feet includes a touching sensor (four optical switches) to perceive which side of the leg end is in contact with the ground. Incremental encoders for position feedback and linear potentiometers at each joint are also incorporated in this vehicle.

The robot described by Ottaviano *et al.* [12] (Figure 2.1c) has a similar design concept of Wheeleg, but its legs have only 1 DOF and its wheels are passive. The robot's mass without batteries is 9 kg and its characteristic dimensions are 1.000×0.500×0.400 m (L×W×H). Experimental tests on the developed prototype showed a maximum achievable speed of 0.07 m/s.

A wall climbing robot is described by Fu *et al.* [13]. The robot (Figure 2.1d), with a mass of 9.5 kg, is composed by a base body with a vacuum adhesion mechanism and a 3-DOF mechanical leg to adapt to different wall surfaces. The base body, with dimensions 300×300×85 mm (L×W×H), includes a three-wheeled mechanism with two driving wheels and a castor wheel (tricycle configuration), which improves stability. The authors report a speed in wheeled mode of 0.17 m/s.

The HyTRo-I robot [14] (Figure 2.1e) comprises four 3-DOF legs and four wheels (two passive omnidirectional wheels at the front and back of the robot's body, and two active wheels on the right and left sides). The locomotion modes defined for this robot include the wheeled rolling mode (using the wheels over flat surfaces), the quadrupedal walking mode (using the legs on uneven surfaces), and the leg-wheel hybrid mode (wheels and legs co-operatively moving). Obstacle detection and body posture regulation are achieved with data from sensors — gyroscope, laser radar and GPS — mounted on the robot's body.

Mantis [15] (Figure 2.1f), a four-wheeled robot developed as a vehicle for surveillance and inspection, derived its name from its two rotating legs, which were inspired by the shape of a praying mantis legs. Its dimensions are $0.335 \times 0.298 \times 0.160$ m (L×W×H) and it weighs 3.2 kg (the mass includes a surveillance camera and a 2600 mAh LiPo battery). Mantis has a maximum payload mass of around 1 kg (compatible with typical inspection gear), can achieve a maximum speed on flat ground of 0.64 m/s, and the maximum step it can climb is 200-mm-tall (25% higher than the robot's rest position).

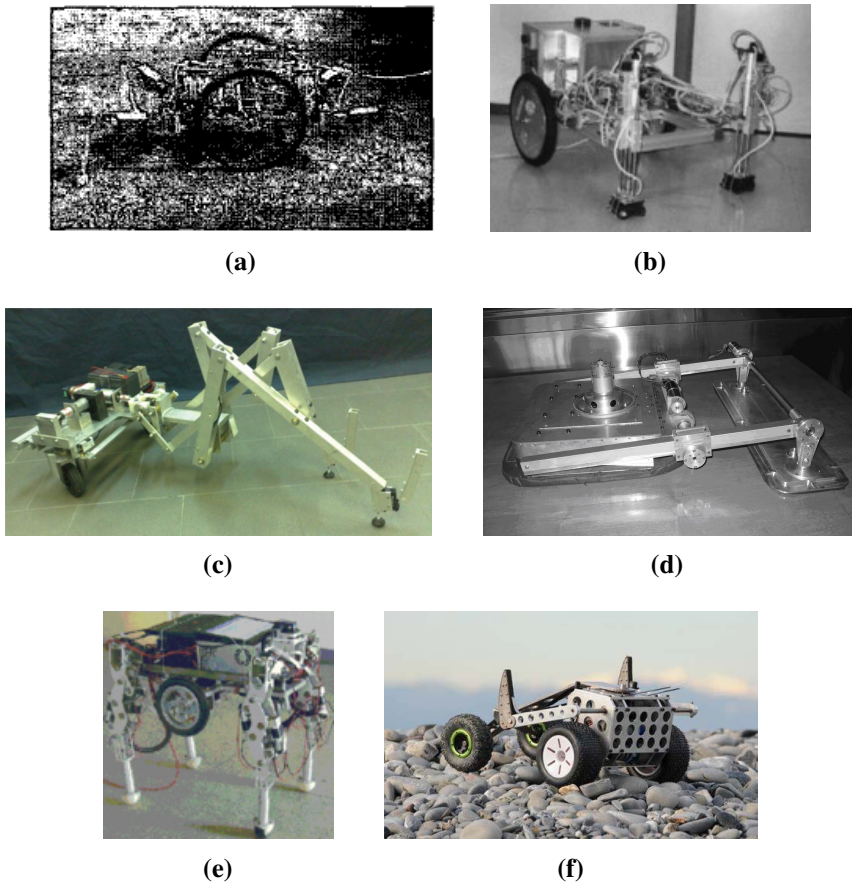


Figure 2.1: Examples of robots with separate wheels and legs. (a) Chariot III [10] ©2004 IEEE. (b) Wheeleg [11]. (c) Legged-wheeled robot described by Ottaviano *et al.* [12]. (d) Wall climbing robot [13] ©2008 IEEE. (e) HyTRo-I [14] ©2013 IEEE. (f) Mantis [15].

2.2.2 Legged-Wheeled Robots

This type of vehicle has wheels mounted at a joint or at the end of a leg.

WorkPartner [16], CENTAURO [17], Momaro [18] are centaur-like vehicles, having an upper body with a two-arm manipulator system sitting on a wheeled quadrupedal lower body.

WorkPartner [16] (Figure 2.2a), developed for urban outdoor environments, is a large scale robot with a weight of about 200 kg and a payload of 40 kg. The hybrid locomotion mode of this robot is called *rolking* (wheels co-operatively work with the legs' joints). A maximum speed of 2 m/s was obtained while moving only by wheels on a flat ground.

The 5-DOF legs of CENTAURO [17] were designed to have large motion range (inward and outward knee arrangements) and combine spider and mammal configurations. CENTAURO, shown in Figure 2.2b, is close to humans in size, 0.610×0.615×1.706 m (L×W×H), and weight, 92 kg. In order to achieve high torque/power density, the authors decided to use the actuators they have designed. The robot includes three colour cameras, an RGB-D sensor and a 3D LiDAR (Light Detection And Ranging). According to the experimental data reported, with a load mounted on the pelvis, CENTAURO showed a payload capacity of 60 kg.

Described by Schwarz *et al.* [18], Momaro (Figure 2.2c) combines its four 4-DOF legs with a pair of steerable wheels (adds 2 DOF to the configuration) at their end. With a base footprint of 0.800×0.700 m (L×W) and a weight of 58 kg, Momaro was built with off-the-shelf components. It includes 3D laser scanners (to provide information to construct a 3D grid map), cameras, a microphone (for auditory feedback), an infrared distance sensor and an IMU (Inertial Measurement Unit). The system is teleoperated, incorporating immersive 3D visualisation. The team of the University of Bonn (Germany) developing Momaro participated in the DARPA (Defense Advanced Research Projects Agency) Robotics Challenge.

Another quadruped with wheels mounted at the end of each leg, which was designed and built at the Laboratoire de Robotique de Paris (France), is HyLoS [19] (Figure 2.2d). The robot incorporates a stereovision system that produces texture information and a digital elevation map. These data are used to decide on the locomotion mode to adopt. Information from force sensors, gyrometers and inclinometers allows dedicated control of the locomotion mode selected. The locomotion modes defined for HyLoS are pure rolling mode (in flat surfaces), rolling mode with reconfiguration (in irregular ground without discontinuities), and peristaltic mode (in non-cohesive soils; wheel traction used to move the leg).

The jumping robot AirHopper [20] (Figure 2.2e) has four legs (widely spread for stability) and four active wheels at foot-end (total of 8 DOF). Legs are driven by pneumatic actuators. This robot measures 1.290×1.200×0.060 m (L×W×H), weighs 34.6 kg, and is able to jump 850 mm and land softly by controlling the in-cylinder pressure.

The BIT-NAZA series includes both wheeled quadrupedal [21] (Figure 2.2f) and wheeled hexapod [22] robots. These heavy payload robots have a maximum payload capacity of the order of 300 kg and a weight of 400 kg. Each leg of the robots is an inverted 6-DOF Stewart platform with a wheel attached at foot-end. The hexapod BIT-NAZA incorporates an attitude sensor, a LiDAR, and

an infrared camera, and reaches maximum speeds in legged and wheeled mode of, respectively, 1.2 and 5.6 m/s.

ANYmal [7] (Figure 2.2g) has four 3-DOF legs terminating with a wheel. It is able to climb steps up to 20% of its leg length and, on flat terrains, can achieve a speed of 2 m/s, with a consumption of 156 W.

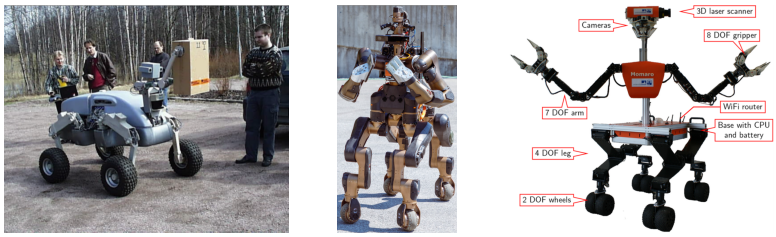
On a slightly different segment are the robots made by Freitas *et al.* [23] and Li *et al.* [24] as, besides ground locomotion, they are able to perform aquatic locomotion. The EHR (Environmental Hybrid Robot) [23] (Figure 2.2h) is targeted to aid in monitoring the Amazon rain forest along the Coari-Manaus pipeline. Each of its four legs terminate in a balloon-like structure, the wheel, that allows floating in water and driving on land. The system is teleoperated. The spherical quadrupedal robot developed by Li *et al.* [24] (Figure 2.2i) is capable of moving underwater. It is divided into two hemispheres (upper one having a diameter of 234 mm and lower one of 250 mm). Its legs have 2 DOF each and its four passive wheels incorporate a braking mechanism to prevent rotation in walking mode. The robot performs a roller-skating motion when using the wheels and the maximum velocity reached was 0.378 m/s.

The robots in Yuk *et al.* [25] and Shi *et al.* [26] were developed considering interaction of the robot with living beings. KaMERo (Kaist Motion Expressive Robot) [25] executes body expressions (emotion and behaviour related) which were evaluated by humans. This robot has two front legs with a passive wheel each and a rear leg with and active wheel. Each leg has 2 DOF. KaMERo (Figure 2.2j) is remotely controlled with a joystick and the three locomotion modes defined are tricycle mode (front wheels forward aligned, while the rear wheel is steering), two-steering mode (front wheels are steerable and rear wheel forward aligned) and synchronous driving (all wheels with same orientation). WR-3 (Waseda Rat No. 3) [26] (Figure 2.2k) is intended for assessing social interactions between robots and rats. This bio-inspired robot resembles, in shape and size, an adult rat. Its structure has a 2-DOF neck, a 1-DOF waist, four 3-DOF legs, and two 1-DOF wheels (mounted at the tip of the rear legs).

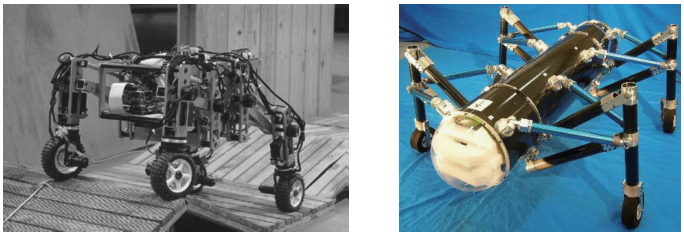
2.2.3 Rotary-Legged Robots

Rotary-legged robots, such as the vehicles in the Whegs series [27] (like DAGSI Whegs™, shown in Figure 2.3a) and ASGUARD [28] (Figure 2.3b), are characterised by their rotating legs/spokes that are driven around an axis. This locomotion mechanism has been considered as an improvement of the traditional wheel structure, simplifying the control mechanisms [29]. Even though stability is compromised, the spokes act similarly to legs and facilitate overcoming obstacles [29].

ASGUARD (Advanced Security Guard) [28] has four wheels, actuated by a 24 V DC (Direct Current) motor each. One wheel includes five compliant legs and the robot was designed at the Robotics Lab of the German Research Center for Artificial Intelligence. Applications envisioned for this robot include security, outdoor surveillance and disaster mitigation missions.

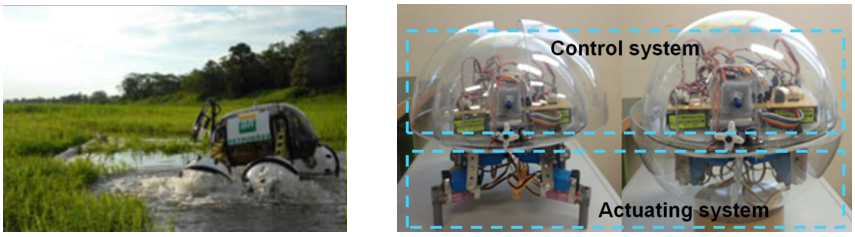


(a) (b) (c)

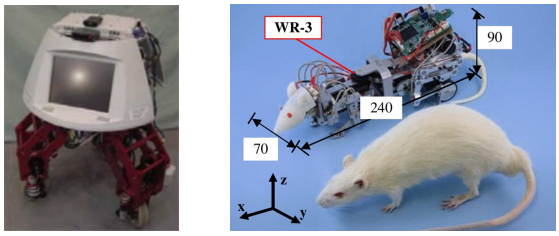


(d) (e)

(f) (g)

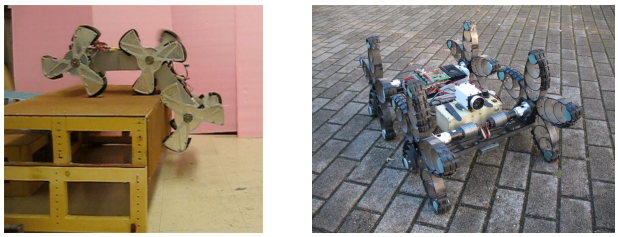


(h) (i)



(j) (k)

Figure 2.2: Examples of legged-wheeled robots. (a) WorkPartner [30]. (b) CENTAURO [17] ©2019 IEEE. (c) Momaro [18]. (d) HyLoS [19]. (e) AirHopper [20] ©2008 IEEE. (f) BIT-NAZA [21]. (g) ANYmal [7] ©2020 IEEE. (h) EHR [23]. (i) Spherical quadrupedal robot [24]. (j) KaMERo [25] ©2008 IEEE. (k) WR-3 (comparison with mature rat) [26] ©2010 IEEE.



(a) (b)

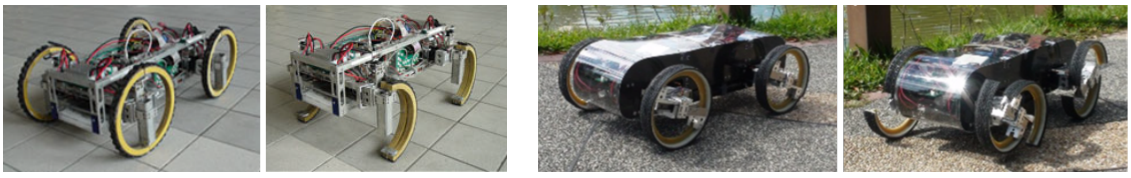
Figure 2.3: Examples of robots with rotating legs. (a) DAGSI Whegs™ [27] ©2008 IEEE. (b) ASGUARD [28].

2.2.4 Transformable Wheel-Legged Robots

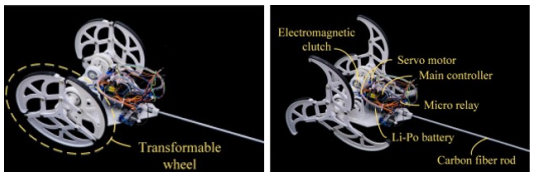
Transformable wheel-legged robots have a single structural assembly that comprises mechanisms to interchange between wheel and leg configurations.

For Quattroped [31] (Figure 2.4a) and its successor, TurboQuad [32] (Figure 2.4b), this is achieved by folding each of their four wheels into two half circles.

In the case of Land Devil Ray [33], shown in Figure 2.4c, passive and active mechanisms are used to switch to legged mode (its two wheels open into a three-spoked structure). This robot is able to climb over obstacles 2.8 times higher than the wheel radius.



(a) (b)



(c)

Figure 2.4: Examples of transformable wheel-legged robots. On the left, the wheeled mode of each robot is presented, while the right side shows the legged mode. (a) Quattroped [31] ©2014 IEEE. (b) TurboQuad [32] ©2017 IEEE. (c) Land Devil Ray [33].

2.3 Simulation in Robotics and Applications

Available 3D simulators with incorporated dynamics (use physics engines) for robotics include Gazebo, Webots and SimTwo [34]. These are all currently free and open source.

Gazebo [35] supports both single and multi-robots' systems and allows simulation of complex indoor and outdoor environments. Also providing access to multiple physics engines, plugins can be enabled in this simulator, expanding the functionalities of the models created (it has, namely, ROS – Robot Operating System – plugins).

Webots [36] is a platform that allows design, testing and validation of different types of robots.

SimTwo [37], which has been used in this work, is an application with multiple windows that are used to visualise or define specific features of the simulation, such as the models of the robots, the environment setup and control algorithms.

Published studies reporting the use of SimTwo (examples of such publications can be found next) are related to the development of models of actuators, sensors, robotic platforms and the environment they operate in, as well as to the implementation and testing of control algorithms.

The model of a brushless motor for actuation of a three-wheel omnidirectional robot was derived and validated in [38] (later improved by the authors in [39]). In [40] and [41], the actuators of a differential robot were modelled (the latter corresponds to a geared motor). Servo motors have been modelled in [42] and [43].

In [40], Gonçalves *et al.* have described the model of a light sensor, simulating it on a differential robot. Gonçalves *et al.* [38] and Campos *et al.* [44] modelled an infrared distance sensor of the Sharp family and the laser scanner of a hacked Neato XV-11, respectively, which were then embedded in an omnidirectional robot for simulation.

The simulation of a Lego Mindstorms NXT based robot has been presented by Gonçalves *et al.* [40]; it was compared to the real robot with an approach for path following based on a light sensor.

Communication between SimTwo and a remote application via UDP packets has been described by Gonçalves *et al.* [39] and Pinho *et al.* [45]. In [39], localisation and navigation algorithms are executed remotely for an omnidirectional robot in a Fire Fighting Robot Contest arena, while in [45] a SimTwo-ROS framework was tested in the model of a MSL (Middle Size League) soccer robot.

The model of a bipedal humanoid robot and a joint trajectory controller were proposed by Lima *et al.* in [42]. Subsequent work described an optimised gait planning for the robot [43].

The work of Costa *et al.* [46] and Eckert *et al.* [47] address simulation in the context of robotics competitions. The first describes the control architecture, localisation system and trajectory planning methods for an omnidirectional robot operating in an industrial environment. The simulation model was developed and tested both in the simulated and real scenarios of the Robot@Factory competition[2]. The second presents the simulation and control of a robot for the Micromouse competition[3]. The maze (competition scenario) is generated in SimTwo from a text file and the robot maps it through the sensors it has, which measure the distance to lateral and front walls.

[2]To answer the challenges of an AGV (Automated Guided Vehicle) operating in a factory plant, robots in the Robot@Factory competition must solve tasks related to manipulation and transportation of objects, while navigating in the environment and avoiding obstacles and collisions [46].

[3]The objective in the Micromouse competition is to use a small autonomous robot to travel an unknown maze and find its centre in the shortest time possible [47].

Specifically for legged-wheeled robots, several publications report the use of simulation to test the robots (not limited to platforms providing the possibility to realistically represent the robot as a whole).

Kamedula *et al.* [48] developed and used a Gazebo-ROS framework to validate the wheeled-legged motion control scheme for CENTAURO [17].

In the development of ANYmal [49] (corresponding to its legged version only), different simulations have been performed. Gazebo has been used to test an online motion plan generator for dynamic gaits [50] and to also evaluate the performance of a contact planner algorithm [51]. The RBDL library (Rigid Body Dynamics Library) was applied for computation of articulated-body dynamics in [52].

A 3D dynamic simulation was conducted by Suzumura *et al.* [53] to validate the developed methodologies, namely for posture control of the robot.

Chen *et al.* [22] evaluated their proposed control strategy for stable walking with co-simulation.

Chapter 3

Hybrid Legged-Wheeled Robotic Vehicle

Throughout this chapter, a description of the robotic platform in which the Dissertation work has been based is provided. The vehicle is being developed by a group of researchers affiliated with FEUP (Faculty of Engineering of the University of Porto) and CRIIS (Centre for Robotics in Industry and Intelligent Systems) at INESC TEC (Institute for Systems and Computer Engineering, Technology and Science). The design, the strategy for control of the joints and the kinematic model of the robot are detailed.

3.1 Introduction

The robot that has been developed (introduced by Pinto *et al.* in [54]) was conceptualised to comply with three key features: (1) *affordable* to prevent that necessary changes greatly influence its prototyping and maintenance budget; (2) *expandable* to allow adaptability, without complicated mechanical alterations, to different tasks or the inclusion of new sensors and actuators; and (3) *modular* to provide reconfigurability, so that changes can be performed locally without modifying the whole system. It was designed to also conform with unstructured environments and different types of terrains.

To fulfil the prime requirements above mentioned, fabrication of the prototype resorted to 3D printing as the main source for the development of the limbs' mechanical parts (PLA – polylactic acid – was the material used), and to aluminum profile.

The vehicle is a quadrupedal robot. Its locomotion system combines a 3-DOF leg with a driving wheel (rubber wheel) and a support (omnidirectional) wheel at foot-end. Figure 3.1 shows the assembled prototype.

3.2 Locomotion System

The work in [54] has described the robot's design, modelling and control of the locomotion system.

Since its publication, changes were performed as issues affecting the robot's mechanical robustness, communications and control were identified. The current version of the locomotion system, as published in [55], is described below.

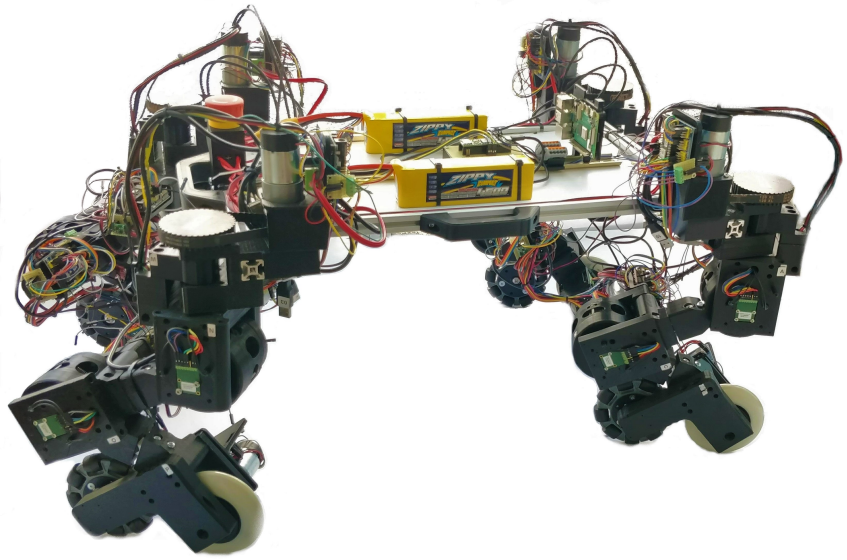


Figure 3.1: The hybrid legged-wheeled robotic vehicle: prototype.

3.2.1 Mechanical Design

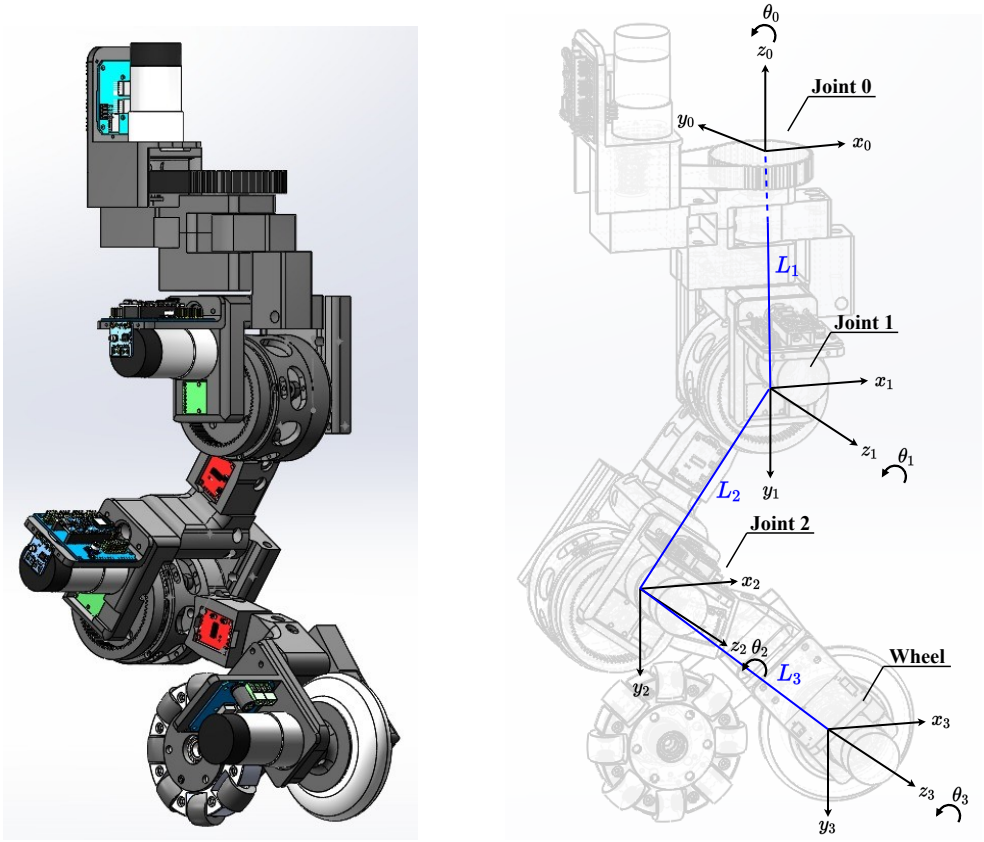
Each limb was designed to have an approximate length of one-third the average human height (1700 mm) and to be compatible, in both size and weight, with a regular car trunk (ease of transportation).

The leg (Figure 3.2) embodies three joints, one rigid (joint 0 in Figure 3.2b) and two non-rigid (joints 1 and 2 in Figure 3.2b), and two wheels, one driving wheel and one free wheel. All joints follow a circular configuration and are equal in size.

A leg configuration combining both rigid and non-rigid joints is beneficial to increase compliance of the leg and improve the vehicle's robustness. The elastic components of a non-rigid joint allow to accommodate impacts and floor irregularities, hence improving the robot's obstacle negotiation capabilities and navigation in rough terrains. It is also noteworthy that leg compliance is associated with the natural dynamics of legged animals. The use of series elastic actuators[1] thus mimics a biological behaviour.

The rigid joint (connects directly to the motor shaft) is the uppermost joint and is thus considered the hip joint. It has a C structure and allows the limb to rotate from $-90°$ to $+90°$, being responsible for steering the vehicle. The in-line reduction gear (4:1 ratio) connected to this joint through a geared belt decreases both gearbox backlash and the motor's effort (as it is the only vertical axis in the system, this joint is liable to impacts).

[1] Series elastic actuators (SEA) designate a type of passive compliant actuators. Compliant actuators are influenced by external forces, which cause changes in their equilibrium position. SEA consist of a spring connected to a stiff actuator.



(a) (b)

Figure 3.2: (a) CAD (Computer-Aided Design) view of the leg configuration. (b) Axis, links and joints numbering of the leg (established according to the leg's forward kinematics).

Each of the non-rigid joints (Figure 3.3) comprises eight steel traction springs attached to a circular support and a rotary damper to reduce high frequency oscillations (caused by the springs). These purely passive elastic joints (non-rigidity established without any energy consumption) have been described in detail by Pinto *et al.* in [56]. As these joints allow to position the vehicle along its z-axis, to achieve the required torque for this task, a planetary gearbox (3.2:1 ratio) was connected to their motor shaft (this gearbox – not included in the design presented in the referred publication – also reduces the joint backlash).

Figure 3.4 shows the conceptual representation of the developed non-rigid joint, as published by Pinto *et al.* in [54], and Equations (3.1) and (3.2) establish the linear differential equations that model the joint (the superscript j designates the joint number, which, according to Figure 3.2b, is 1 or 2).

$$J_1\,\dot{\omega}_1^j = \overbrace{k\,i}^{T_m} - B_1\,\omega_1^j - k_s\left(\theta_1^j - \theta_2^j\right) - B_d\left(\omega_1^j - \omega_2^j\right) \tag{3.1}$$

$$J_2\,\dot{\omega}_2^j = -B_2\,\omega_2^j + k_s\left(\theta_1^j - \theta_2^j\right) + B_d\left(\omega_1^j - \omega_2^j\right) - d\,m\,g\,\sin\left(\theta_2^j\right) \tag{3.2}$$

The parameters R, u and i are associated with the motor model defined by Equation (3.3); u is

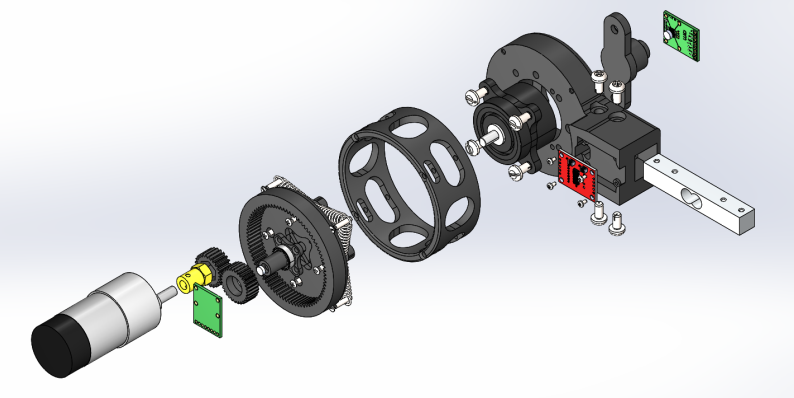


Figure 3.3: CAD of the non-rigid joint (exploded view).

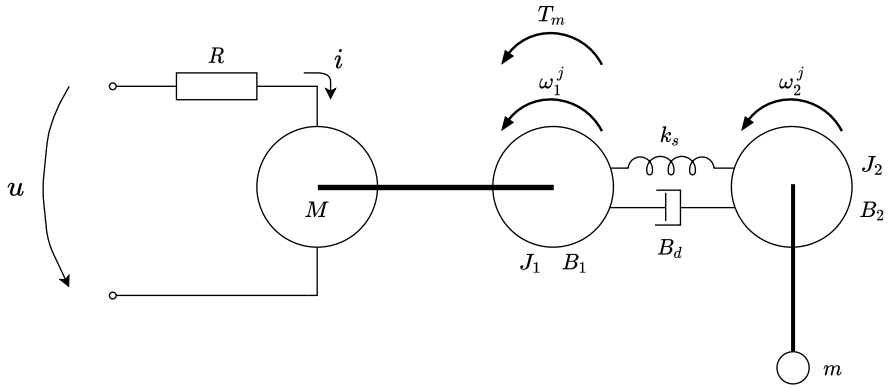


Figure 3.4: Conceptual representation of the non-rigid joint.

the voltage applied to the motor (V), R the motor resistance (Ω), i the current flow (A), and k the motor constant $\left(\text{N·m}/\sqrt{\text{W}}\right)$. As mentioned in [54], the motor's static friction torque T_q (N·m) is given by $k \times i$, and T_q and T_m (motor's torque, N·m) are assumed to be equal when $T_m < T_{q_{max}}$.

$$u = R\,i + k\,\omega_1^j \tag{3.3}$$

In regards to the remaining parameters: k_s is the spring constant (N/m), and B_d the damping coefficient (N·m·s); B_1 and B_2 are the viscous friction coefficients (N·m·s); J_1 and J_2 correspond to the moments of inertia (kg·m^2), and ω_1^j and ω_2^j, and $\dot{\omega}_1^j$ and $\dot{\omega}_2^j$ are the respective angular speeds (rad/s) and accelerations (rad/s^2); θ_1^j and θ_2^j are the angular positions (rad); d (m) is the distance between the joint and a mass m (kg), and g corresponds to the acceleration of gravity (m/s^2). The parameters with the subscript *1* are associated to the joint input, while those with the subscript *2* to the joint output.

The spring induced torque τ_{k_s} and the viscous induced torque τ_{B_d} are represented, respectively, by Equations (3.4) and (3.5).

$$\tau_{k_s} = k_s\left(\theta_1^j - \theta_2^j\right) \tag{3.4}$$

$$\tau_{B_d} = B_d\left(\omega_1^j - \omega_2^j\right) \tag{3.5}$$

By combining Equation (3.1) with Equation (3.3) and considering $\omega_1^j = \dot{\theta}_1^j$ and $\omega_2^j = \dot{\theta}_2^j$, the state-space model of the non-rigid joint, with respect to the state vector X in Equation (3.6), is obtained. The model follows the state-space description in Equation (3.7) and is represented by Equation (3.8). In Equation (3.7), u is the system input (in this case, the motor voltage), while y the system output; A, B and C are, respectively, the $n \times n$ system matrix, the $n \times 1$ input vector, and the $1 \times n$ output vector (n corresponds to the system order, being 4 for the joint). Detailed steps on how the model was obtained, as well as description of the system's transfer function and estimation of the parameters, are found in [54].

$$X = \begin{bmatrix} \theta_1^j \\ \omega_1^j \\ \theta_2^j \\ \omega_2^j \end{bmatrix} \tag{3.6}$$

$$\begin{aligned} \dot{X} &= AX + Bu \\ y &= CX \end{aligned} \tag{3.7}$$

$$\begin{bmatrix} \dot{\theta}_1^j \\ \dot{\omega}_1^j \\ \dot{\theta}_2^j \\ \dot{\omega}_2^j \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_s}{J_1} & -\left(\frac{k^2}{RJ_1} + \frac{B_1 + B_d}{J_1}\right) & \frac{k_s}{J_1} & \frac{B_d}{J_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{J_2} & \frac{B_d}{J_2} & -\frac{k_s + d\,m\,g}{J_2} & -\frac{B_2 + B_d}{J_2} \end{bmatrix} \begin{bmatrix} \theta_1^j \\ \omega_1^j \\ \theta_2^j \\ \omega_2^j \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{RJ_1} \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_1^j \\ \omega_1^j \\ \theta_2^j \\ \omega_2^j \end{bmatrix} \tag{3.8}$$

As aforementioned, the leg connects to a driving and support wheels at foot-end. Advantages of this arrangement are related to the reduction of energy consumption to a minimum as the robot is endowed with a low-power configuration, in which it is capable to move simply using the wheels (this concern is due to the fact that legged systems require continuous actuation of the joints to maintain their stability, resulting in higher energy consumption in comparison to traditional wheeled systems).

The leg configuration developed potentially reduces energy consumption even further due to the driving wheel, which is attached to the last link of the limb, being motorised for rotation only. Despite the wheels have a fixed axle, owing to rotation of the legs, omnidirectional motion is possible.

The role of the single omnidirectional wheel in each leg is associated to the promotion of a stable platform while in wheeled mode, as it is an additional contact point with the ground that allows the driving wheel to stay aligned with the hip joint.

3.2.2 Electrical Design

The three joints and the rubber wheel of each leg are actuated using brushed DC motors with in-line gearbox and built-in incremental encoder to determine the angular position. The motors use their own driver for motor control. A pair of rotary position sensors (magnetic rotary encoders) in each of the three joints provides information on absolute angular positioning (one measures the angular position of the joint and the other the extension of the springs).

A load cell couples both halves of the links, connecting the non-rigid joints and the bottom non-rigid joint to the wheel. These links were divided to get accurate and real-time sensing of the force in the leg; the load cells will thus yield complementary values of the force exerted on the leg.

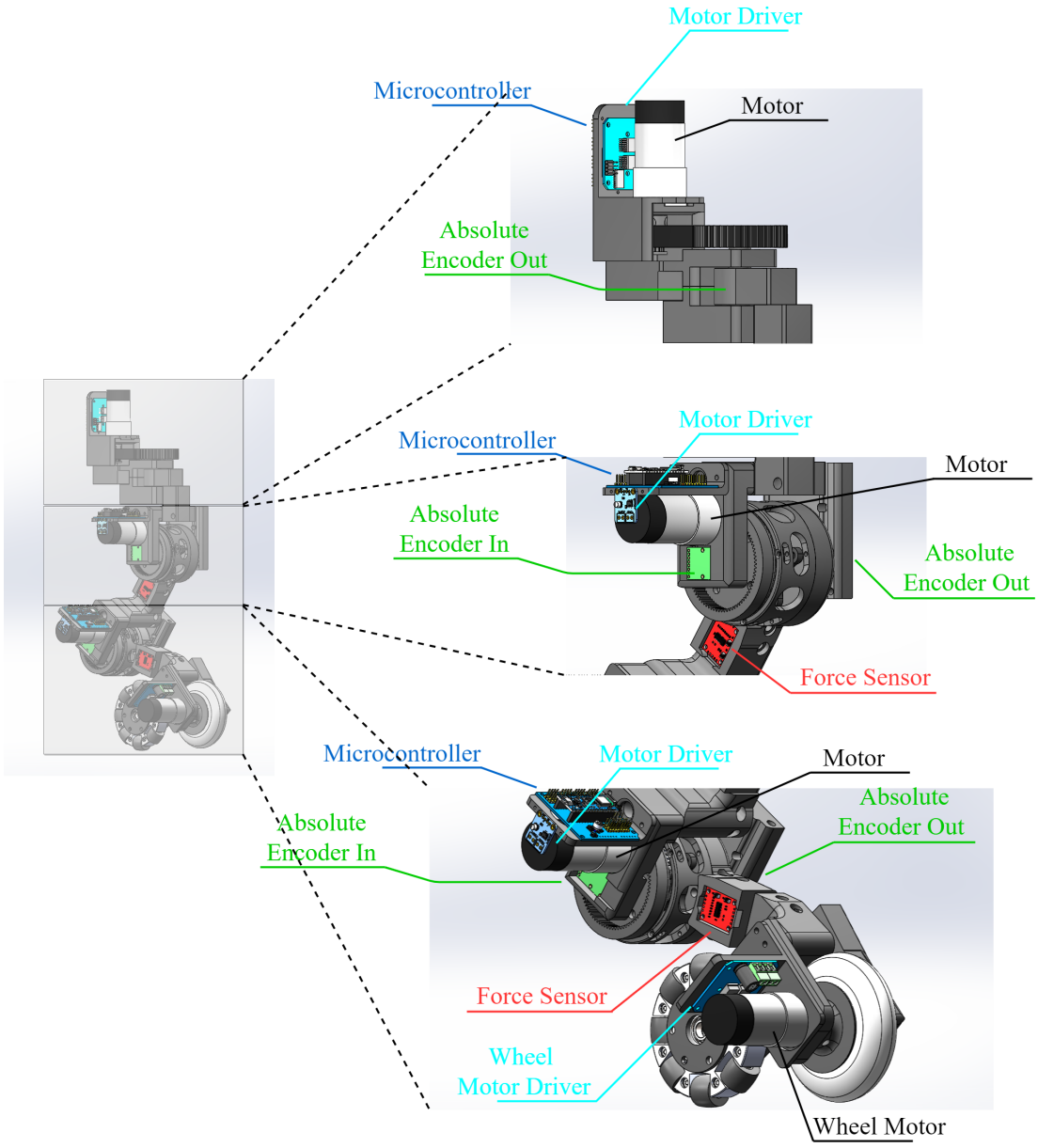


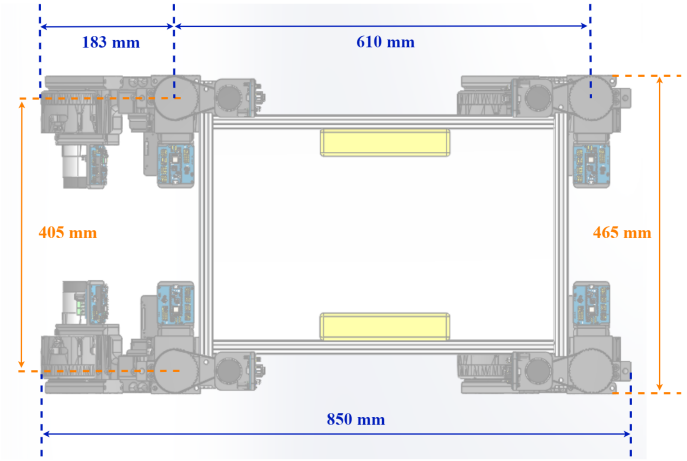
Figure 3.5: Hardware diagram of a single leg.

Three Arduino Nano 33 IOT receive data from the sensors and control the motor drives of the joints (the reference signals given to each microprocessor and the controllers uploaded in each of them, enabling control of the joints, can be consulted in the diagram of Figure 3.8).

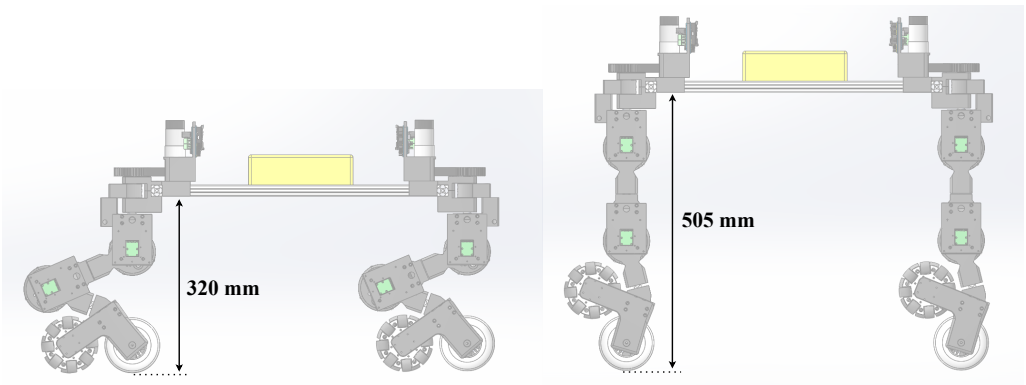
Figure 3.5 shows the position of the aforementioned hardware in the leg.

3.3 Robotic Platform

As shown in Figure 3.6, the built prototype of the vehicle has characteristic dimensions of 610×405 mm (L×W) and, taking into account mechanical constraints, its height can vary between 320 and 505 mm. Its approximate mass is 18 kg. Except for the hip joint, which connects each leg to the main body of the robot, the four legs follow the same position configuration.



(a)



(b)

Figure 3.6: Dimensions — (a) size, and (b) height — of the robot.

Given that the robot combines legs with wheels, three locomotion modes are defined to exploit the advantages of both mechanisms: purely legged, purely wheeled, and hybrid. In purely legged mode, wheels are not used and a gait pattern must be generated to allow the robot to walk (which requires definition and implementation of a gait planning framework); walking enables, for example,

stair climbing. In purely wheeled mode, even though motion relies solely on wheels, the robot's height can be varied to adapt to the environment it is operating in. Hybrid mode will combine both gait and wheeled motion to allow surpassing or avoiding different types of obstacles.

The robot performs in a low-power state when in purely wheeled mode and at rest position. The rest position is defined as the robot's lowest achievable height, allowing deactivation of the controllers of the non-rigid joints.

3.3.1 Kinematic Model

In purely wheeled mode, the x-axis of the wheels is kept aligned with the rotation axis of the hip joint. To derive the robot kinematics, this consideration was taken into account.

Figure 3.7 shows and describes the variables of the kinematic model of the vehicle (also including the numbering of the legs). The variables to be controlled, allowing modulation of the robot motion across the x and y axes, are the angular position of each hip joint, which defines the wheel's angle (θ_{wheel}), and the speed of the wheel (v_{wheel}).

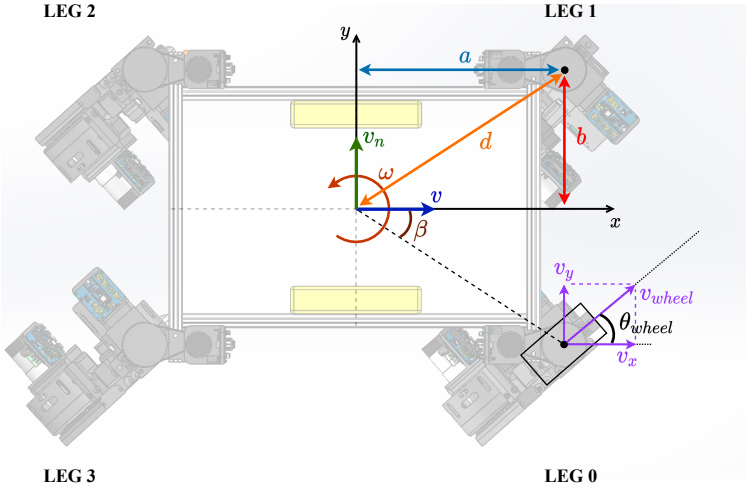


Figure 3.7: Kinematic variables of the robot.

Considering the trigonometric relations that can be established between the angle β and the variables a, b and d, the orthogonal projections of v_{wheel} can be written as presented in Equations (3.9) and (3.10) (this simplification allows to reduce the computational resources used and improve algorithmic efficiency, since it would otherwise be necessary to apply the trigonometric functions sine and cosine). Equations (3.11) and (3.12) allow computing v_{wheel} and θ_{wheel} according to the v, v_n and ω speeds.

The mentioned equations are associated to the contribution of a generic leg to v, v_n and ω; for each leg, the respective wheel speed can be calculated using the equations in Table 3.1.

$$v_x = v - \omega \cdot y \tag{3.9}$$

$$v_y = v_n + \omega \cdot x \tag{3.10}$$

$$v_{wheel} = \sqrt{v_x^2 + v_y^2} \tag{3.11}$$

$$\theta_{wheel} = \text{atan2}\,(v_y,\ v_x) \tag{3.12}$$

Table 3.1: Equations for calculation of the orthogonal projections of the linear speed of each leg's wheel.

	Leg 0	**Leg 1**	**Leg 2**	**Leg 3**
$\boldsymbol{x}$	a	a	$-a$	$-a$
$\boldsymbol{y}$	$-b$	b	b	$-b$
$\boldsymbol{v_x}$	$v+\omega\cdot b$	$v-\omega\cdot b$	$v-\omega\cdot b$	$v+\omega\cdot b$
$\boldsymbol{v_y}$	$v_n+\omega\cdot a$	$v_n+\omega\cdot a$	$v_n-\omega\cdot a$	$v_n-\omega\cdot a$

Therefore, x-y control of the robot is achieved by setting the v, v_n and ω speeds. These are then converted to θ_{wheel} and ω_{wheel}, which are the references fed to the controllers of the hip joints and the wheels, respectively (as outlined in Figure 3.8).

3.3.2 Control

The vehicle is controlled along the x, y and z axes and can thus perform omnidirectional motion. Simultaneous control of multiple axes is also possible.

Each leg of the robot is controlled individually. To integrate the data from the microprocessors in each leg and execute the necessary calculations to generate the references for control of each joint and wheel, a Teensy 4.1 board is used as the main microprocessor. Data transfer in the system is implemented through serial communication, as represented in the schematic of Figure 3.8.

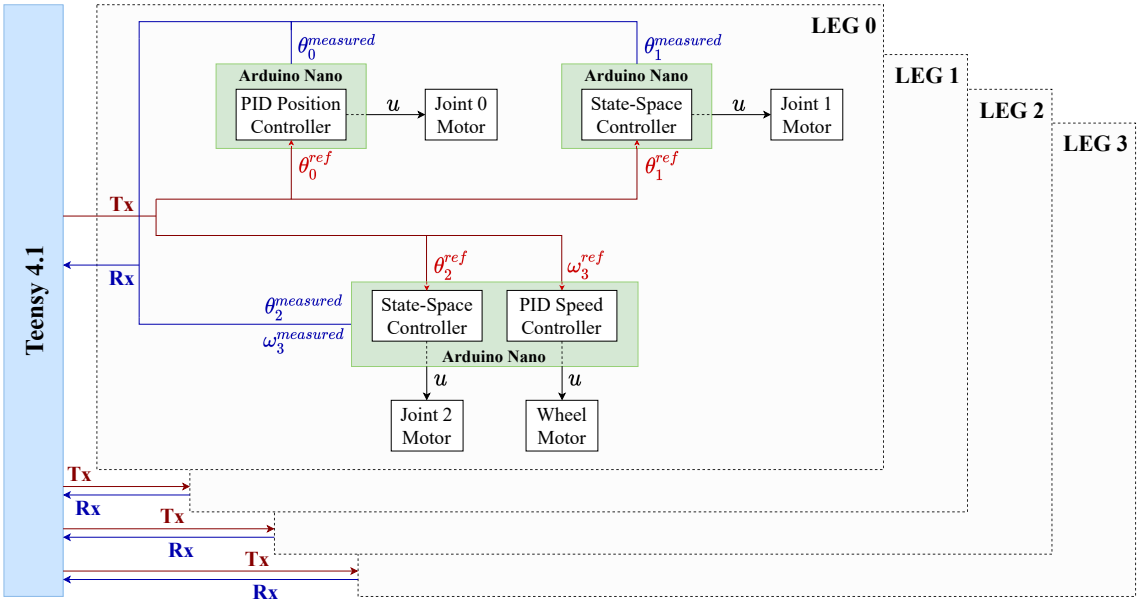


Figure 3.8: Macro diagram of the hardware of the robot for control and communication.

Regarding control of the joints, as can be inferred from Figure 3.8, state-space controllers are applied to each of the non-rigid joints, while PID controllers (Proportional-Integral-Derivative controllers) are used for position control of the hip joint and for speed control of the wheel.

A state-space approach has been adopted to control the position of the non-rigid joints since the control strategy presented in [56], which was based on PID controllers, rapidly exhibited instabilities. The state-space controller has been described in [54].

3.4 Simulation Model

The simulation model of the robot has been developed in SimTwo. The model and its validation, as well as the high and low-level control architectures implemented in simulation are described in the following chapter.

Chapter 4

Realistic Simulation of a Robotic Vehicle with Hybrid Locomotion

As mentioned, a realistic model of the robotic platform described throughout Chapter 3 has been developed in the SimTwo simulator software to allow the definition and testing of a high-level control architecture for this robot. This chapter presents the designed simulation model, its validation, and the developed framework for trajectory tracking, which was used for navigation inside a simulated inpatient unit.

4.1 Introduction

SimTwo [37] provides a 3D environment with incorporated dynamics for realistic simulation of different types of robots, such as manipulators, wheeled and legged mobile robots, and even robots with aquatic locomotion. This high versatility is a result of how robots are described in the simulator: the robotic system must be decomposed into an arrangement of rigid bodies and joints, which are optionally powered with motors.

Therefore, considering all the features of the SimTwo simulator, it seemed feasible to create a realistic simulation model of the vehicle through this software. The developed model (Figure 4.1) is a valuable asset to study, test, evaluate and improve the mechanics and control architecture of the robot, without being dependent on the real vehicle.

4.2 Simulation Model Design

SimTwo uses different XML files (with an extended XML syntax) for characterisation of the physical simulation environment, allowing definition of the properties of the system to be simulated.

The simulation model of the legged-wheeled robot was designed by decomposing it into a set of rigid bodies having mass and size equal to the corresponding parts of the real robot. The characteristic dimensions of the model, as well as the minimum and maximum height are depicted, respectively, in Figure 4.2a and Figure 4.2b.

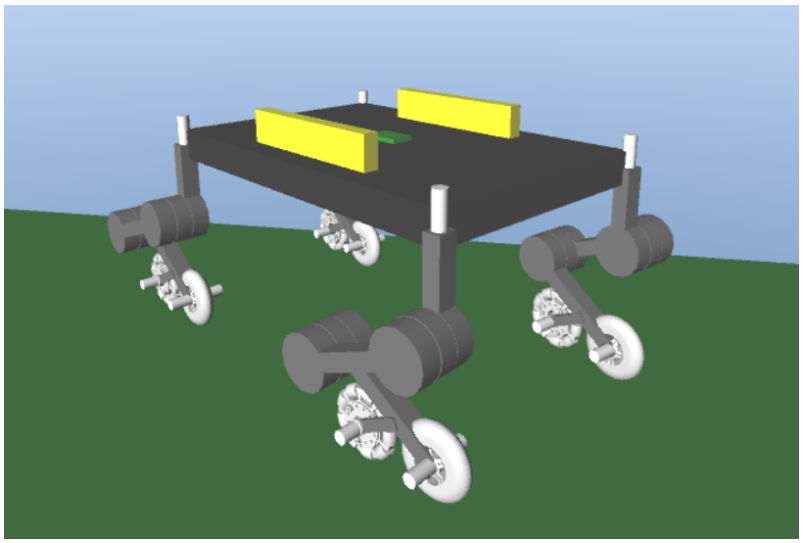
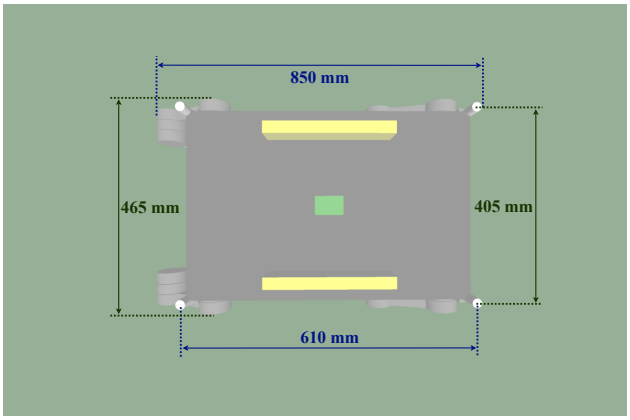
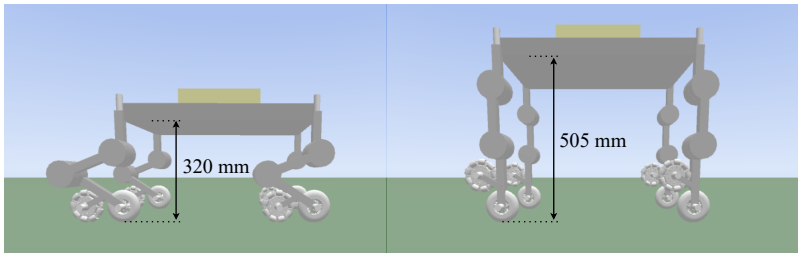


Figure 4.1: The hybrid legged-wheeled robotic vehicle: simulation model.



(a)



(b)

Figure 4.2: Dimensions — (a) size, and (b) height — of the robot's simulation model.

Leg links and the main body were represented as cuboids, while joints and wheels as cylinders (joints allow connection between different rigid bodies). Except for the joint for rotation of the omnidirectional wheel, all the others are actuated. Thus, and as shown in Figure 4.3, the model comprises four legs, that connect to the main body on their uppermost joint (joint 0), and eight

wheels. Both joints' and legs' numbering in simulation follow the numbering defined for the real robot, which has been represented in Figure 3.2b and Figure 3.7, respectively.

The locomotion system of the robot (detail in Figure 4.3) was modelled by combining, per leg, three links with five joints (includes the wheels' joint). The z-axis is the actuation axis of the hip joint, while the y-axis is that of the remaining ones.

Non-rigidity of joints 1 and 2 was modelled by assembling a compound joint: two separate joints connect the leg links to an intermediate cylindrical link; one of them is actuated and the other has a spring (simulate the elastic component of the joint). This configuration considered the the non-rigid joint conceptual representation presented in Section 3.2.1.

Omnidirectionality of each non-actuated wheel was established by defining two coefficients of friction onto their surface: one equal to less than one to allow lateral motion (the value was set to 0.02), and the other equal to one to hinder forward/backward motion.

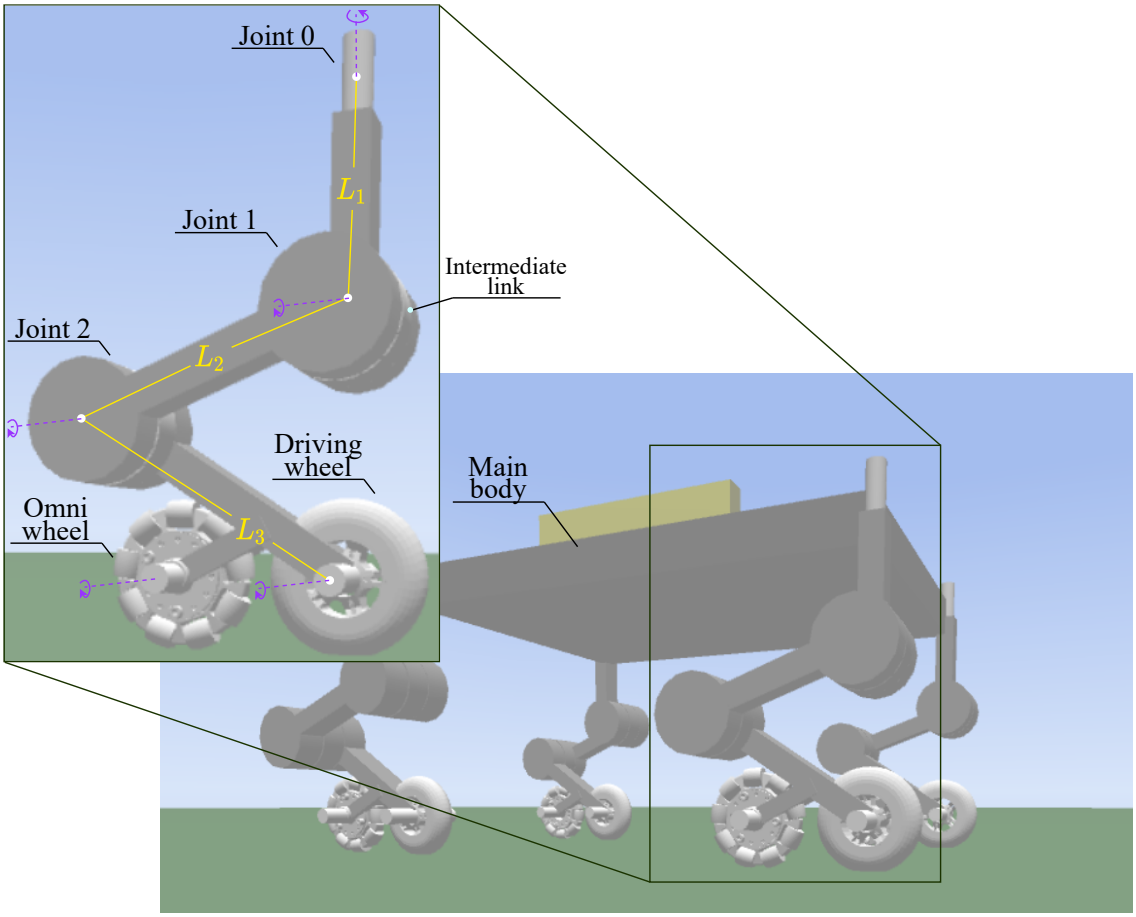


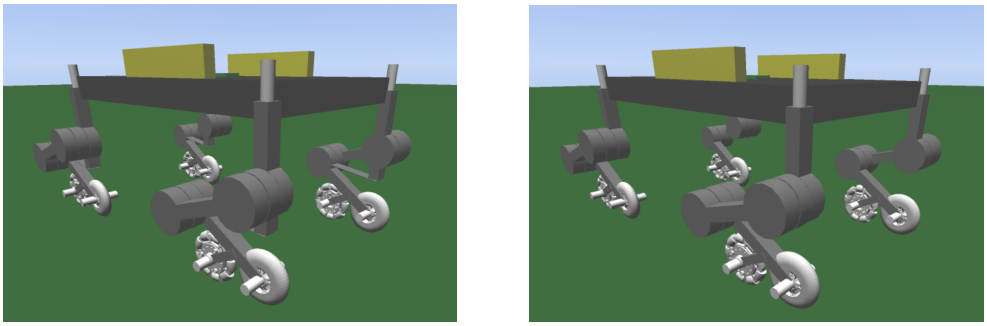
Figure 4.3: Simulation model: leg detail with links and joints numbering, and actuation axis of each joint.

Table 4.1 displays the values of the parameters of the rigid bodies used to model the robot. As can be noticed, only the mass of the wheels is mentioned; these values were looked up in the datasheet of the wheels used in the real robot. An estimate of 5 kg and 0.5 kg was assigned to the mass of the main body and links, respectively.

Table 4.1: Parameters of the rigid bodies of the simulation model of the legged-wheeled robot.

Main body	Length 0.610 m	Width 0.405 m	Height 0.050 m
Links (length)	L_1 0.186 m	L_2 0.168 m	L_3 0.130 m
Links (size)	0.025 × 0.030 m		
Driving wheel	Diameter 0.100 m	Thickness 0.024 m	Mass 0.132 kg
Omni wheel	Diameter 0.100 m	Thickness 0.019 m	Mass 0.175 kg

While in purely wheeled mode and performing in low power, the real vehicle keeps its position at its lowest height due to mechanical constraints. Modelling this behaviour required placing cuboid shells (shown in Figure 4.4a) that restrict the robot position to remain at the lowest height as, upon collision, they stay fixed, avoiding that rigid bodies plunge into one another.



(a) **(b)**

Figure 4.4: Visualisation of the strategy adopted to constrain the robot's position at lowest height. (a) Cuboid shells for position restriction visible. (b) Cuboid shells for position restriction hidden.

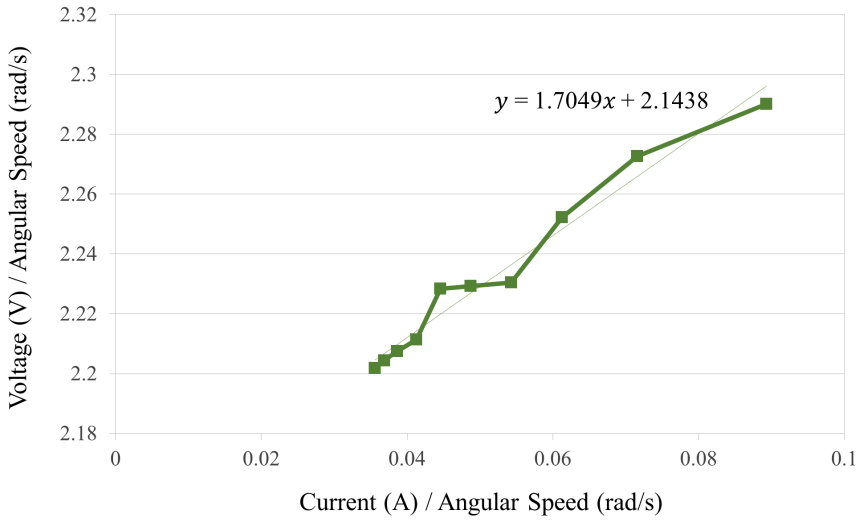
4.3 Simulation Model Actuation System

Although described in Chapter 3, the additional reduction achieved with the planetary gearbox in each non-rigid joint is yet to be considered in the developed simulation model. The current actuation system of the model is explained below.

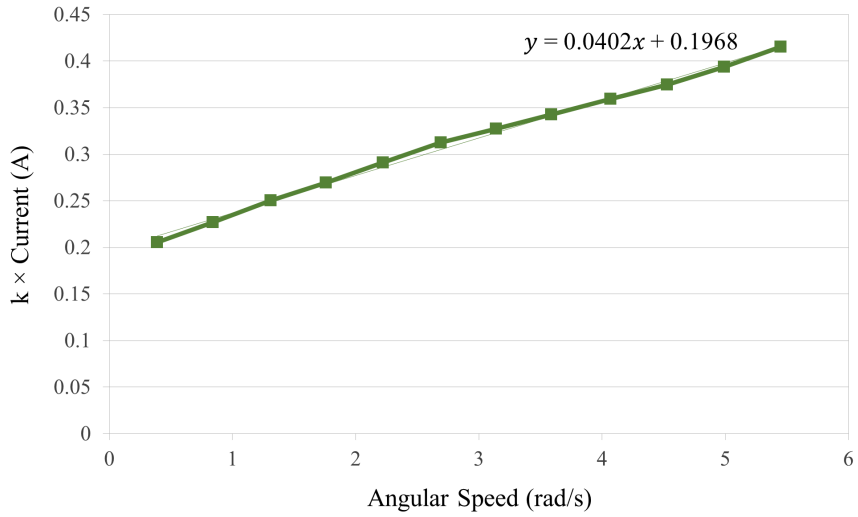
Table 4.2: Properties of the spring of the non-rigid joints.

k_s	B_v	fc
11.92 N/m	0.477 N·m·s	0.010 N·m

The elasticity of the non-rigid joints is characterised by the spring constant k_s, the viscous friction coefficient B_v of the rotary damper, and the static Coulomb friction f_C. The values of these parameters are presented in Table 4.2.



(a)



(b)

Figure 4.5: Estimation of the motor model parameters. (a) Plot of $\frac{u\ (\mathrm{V})}{\omega\ (\mathrm{rad/s})}$ vs. $\frac{i\ (\mathrm{V})}{\omega\ (\mathrm{rad/s})}$. (b) Plot of $k \cdot i$ (A) vs. ω (rad/s).

As referred in [54], the motor-gearbox system has been considered equivalent to the generic model for a DC motor, which is represented in Equation (4.1). In the equation, u is the voltage applied to the system (V), R the motor resistance (Ω), i the current flow (A), and k the motor constant $\left(\mathrm{N{\cdot}m}/\sqrt{\mathrm{W}}\right)$. The parameters that model a DC motor also include the viscous friction coefficient B_v (N·m·s) and the static friction torque T_q (N·m). The latter is given by the product

between k and i.

$$u = R\,i + k\,\omega \tag{4.1}$$

The hardware tests performed to derive the motor model are described in [57]. The motor of the non-rigid joints was used. The results are plotted in Figure 4.5 and the parameters estimated from these data are summarised in Table 4.3.

Table 4.3: Estimated parameters of the motor.

R	k	B_v	T_q
1.705 Ω	2.144 N·m/$\sqrt{\text{W}}$	0.040 N·m·s	0.197 N·m

Since it was intended to set, in the simulation, the gear ratio of the motors equal to one, the model for the motor without gearbox was deduced from the parameters estimated by assuming that R and B_v are the same, and k and T_q are divided by the value of the reduction. Taking into consideration that the motors used in the real vehicle are from the same manufacturer, the parameters for the motors of the hip joints and wheels were computed, following the rationale just described (which means that, considering the values of the system without gearbox, k and T_q are multiplied by the reduction). Table 4.4 presents the calculated parameters for the simulation of each joint's motor (v_{max} and i_{max} correspond to the maximum values, defined by the manufacturer, for the motor's voltage and current, respectively).

Table 4.4: Parameters of the motors powering each joint of the simulation model of the legged-wheeled robot.

	Motor (no gearbox)	**Joint 0**[a]	**Joint 1**	**Joint 2**	**Wheel**
Gear ratio	—	70:1 + 4:1	210:1	210:1	43.8:1
R (Ω)	1.700	1.700	1.700	1.700	1.700
k $\left(\text{N·m}/\sqrt{\text{W}}\right)$	0.0102	2.853	2.140	2.140	0.446
B_v (N·m·s)	0.0402	0.0402	0.0402	0.0402	0.0402
T_q[b] (N·m)	9.371×10^{-4}	0.262	0.197	0.197	0.0410
v_{max} (V)	12	12	12	12	12
i_{max} (A)	6.5	6.5	6.5	6.5	6.5
PPR[c] (encoder)	—	4480	3360	3360	700

[a] Parameters were calculated considering the reduction provided by both the motor's gearbox and the in-line gear.
[b] In SimTwo, T_q is defined by the property f_C.
[c] Pulses per revolution.

The values of PPR in Table 4.4 refer to the built-in encoders of the motors; these were extracted from the CPR (counts per revolution) value provided by the motors' manufacturer (CPR = 64).

Equation (4.2) shows the relation between these entities.

$$\text{PPR} = \frac{1}{4} \times \text{CPR} \tag{4.2}$$

For a motor-gearbox system, its PPR corresponds to the product between the motor's encoder PPR and the value of the reduction.

4.4 Simulation Model Control

Control of a system ensures its stability, producing the required behaviours to achieve its goals. Controlling its processes (e.g., speed of a motor) allows keeping a process variable close to the desired value (reference), despite disturbances or other variations that impact the system.

The control architecture of the robot implemented in simulation is described hereafter.

4.4.1 Low-Level Control

As mentioned in Section 3.3.2 in regards to the robot, the simulation also implements PID controllers for position control of each hip joint and for speed control of the wheels, while state-space controllers were developed for control of the non-rigid joints. A 10-ms period was defined for low-level control of the robot.

4.4.1.1 PID Controller

PID controllers have been widely adopted in industrial settings as they deliver consistent performance in control tasks [58]. They represent an important tool in control, owing to their relatively simple implementation, which has thus cemented their role as a standard for feedback control [58].

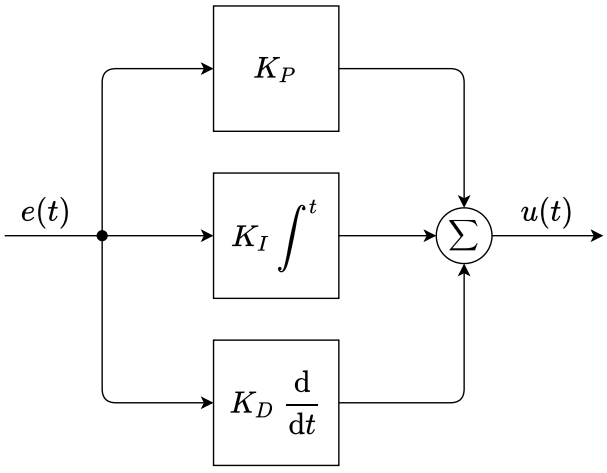


Figure 4.6: Generic block diagram of a PID controller.

A PID controller is a three-term controller applying the following control actions [58, 59]:

- Proportional control (P term): controller action proportional to the current error signal.
- Integral control (I term): controller action proportional to the integral of the error (relate past error values); used to correct steady offsets from a constant reference signal value.
- Derivative control (D term): controller action proportional to the derivative of the error; it introduces a prediction factor into the control action as it uses the rate of change of the error.

The generic time domain representation of a PID controller is shown in Figure 4.6.

In Figure 4.6, K_P, K_I and K_D (controller parameters) denote the proportional, integral and derivative gains, respectively, and e corresponds to the system error (controller input), while u represents the control signal.

Considering the simplified closed-loop controller in Figure 4.7, where r corresponds to the reference signal and y to the process output (controlled variable), e is computed as the difference between r and y.

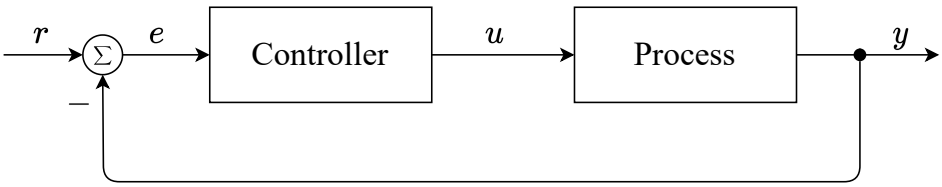


Figure 4.7: Schematic block diagram of a feedback control system.

The PID control law in both time and Laplace domains is given by Equations (4.3) and (4.4), respectively.

$$u(t) = K_P\, e(t) + K_I \int_{-\infty}^{t} e(t)\,\mathrm{d}t + K_D\, \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{4.3}$$

$$u(s) = \left(K_P + \frac{K_I}{s} + k_D\, s\right) e(s) \tag{4.4}$$

This short introduction on PID control is followed by the description of the specific PID controllers used.

PD Position Controller

The diagram of the PD controller for position control of each hip joint is presented in Figure 4.8. The angular position of the joint, denoted in the diagram as θ_0 (according to the numbering in Figure 3.2b), corresponds to the wheel's angle θ_{wheel} of the kinematic model of the robot (Section 3.3.1). The control signal u_0 is the voltage applied to the hip joint's motor. The saturation block introduced in the controller limits the motor's voltage between its minimum and maximum values, which, as referred in Table 4.4, correspond to ± 12 V.

The controller includes P and D terms only. The value of each of the controller gains is given in Table 4.5. These values correspond to the ones that have been implemented in the real robot.

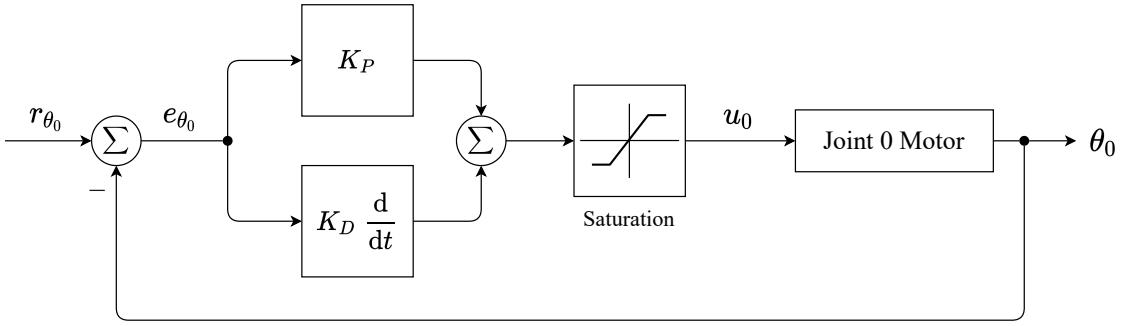


Figure 4.8: Block diagram of the PD controller for position control of each hip joint.

Table 4.5: Gains of the PD controller of the hip joint position.

K_P	K_D
4.00	3.00

PI Speed Controller

Figure 4.9 depicts the diagram of the PI controller for speed control of each wheel. The angular speed of the wheel, denoted in the diagram as ω_3 (according to the numbering in Figure 3.2b), relates to the wheel's linear speed v_{wheel} of the kinematic model of the robot (Section 3.3.1). The voltage applied to the wheel's motor is the control signal, u_3, of the controller.

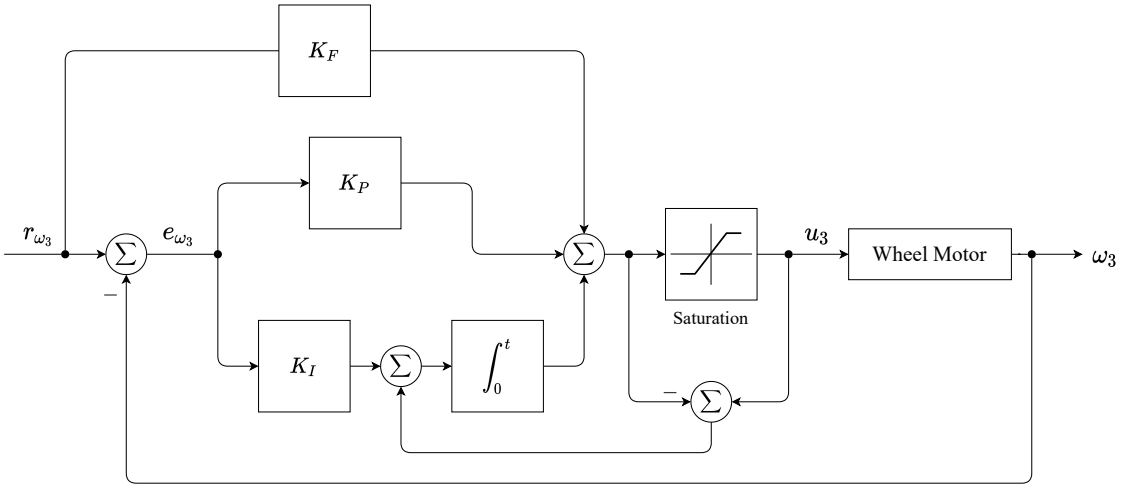


Figure 4.9: Block diagram of the PI controller for speed control of each wheel.

Besides the feedback control loop, a feedforward control loop was incorporated. The feedforward of the reference signal allows speeding up the response (K_F denotes the feedforward gain). The controller also includes a saturation block, which limits the motor's voltage between its minimum and maximum values (as presented in Table 4.4, these correspond to ± 12 V).

In cases where saturation limits are applied, saturation of the controller output leads to an effect denominated *integrator windup effect*, which causes degradation of the controller response [60].

This situation occurs when, upon changing the reference signal (step input), the control variable reaches the saturation limits during the transient response. Consequently, the error decreases slowly and accumulates, resulting in the growth of the integral term [60]. The response will likely exhibit a large overshoot and settling time [60]. A number of techniques can be applied to avoid this phenomenon.

As P and I terms characterise the controller, an anti-windup strategy is included to consider the integral term only when the control signal outputted by the controller is not saturated.

Table 4.6 contains the values of the controller gains. These have been adjusted in simulation and are thus not the same as the ones implemented in the real robot.

Table 4.6: Gains of the PI controller of the wheel speed.

K_P	K_I	K_F
0.20	0.04	0.55

4.4.1.2 State-Space Controller

The angular position of both non-rigid joints is controlled through a state-space control technique. Figure 4.10 shows the block diagram of the applied controller, where the subscript j designates the joint number (following Figure 3.2b, j is either 1 or 2). The control signal u_j is the voltage applied to the motor, and the reference signal r_{θ_j} is the desired (absolute) angular position of the joint.

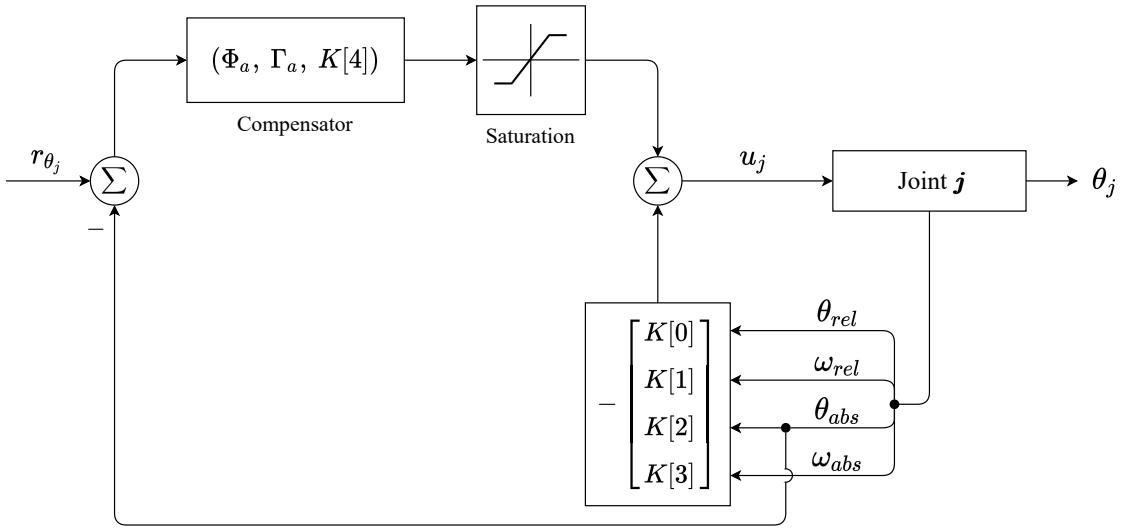


Figure 4.10: Block diagram of the state-space controller for a non-rigid joint.

Both relative and absolute positions of a non-rigid joint are used by the controller. Thus, θ_{rel}, ω_{rel}, θ_{abs}, and ω_{abs} correspond to the state variables. According to the joint model presented in Section 3.2.1, the controller state variables relate to the joint state variables through Equations (4.5) to (4.8).

$$\theta_{rel} = \theta_1^j \tag{4.5}$$
$$\omega_{rel} = \omega_1^j \tag{4.6}$$
$$\theta_{abs} = \theta_2^j + \theta_1^j \tag{4.7}$$
$$\omega_{abs} = \omega_2^j + \omega_1^j \tag{4.8}$$

Considering that the controller is applied to a discrete-time system and the joint model (described in Section 3.2.1) is continuous-time, the methods proposed by Vaccaro [61] to discretise the system were followed. The discretisation and estimation of the feedback vector K are detailed by Pinto *et al.* in [54] (K corresponds to the controller gains). The feedback vector implemented in simulation is presented below (both joints share the same K).

$$K = \begin{bmatrix} K[0] \\ K[1] \\ K[2] \\ K[3] \\ K[4] \end{bmatrix} = \begin{bmatrix} 10.7110 \\ 0.5160 \\ 2.3263 \\ 0.4410 \\ 0.3118 \end{bmatrix} \tag{4.9}$$

As can be observed in Figure 4.10, the controller includes a compensator that implements an additional dynamics block. This additional dynamics is introduced to drive the system to a nonzero equilibrium state, so that it responds to step inputs; it also provides the capability to deal with unit step disturbances. Φ_a and Γ_a are both equal to one.

Unlike the controller that has been implemented in the real robot, the saturation block, in simulation, acts directly on the state variable of the additional dynamics, consequently limiting the voltage applied to the joint's motor (improved controller performance was observed with this choice).

Another difference between the controller in the robot and in the simulation is the state observer, which has not been considered. Observers are used to estimate variables; they are useful for reduction of the number of state variables to be measured in high-order systems, or when the measurement of a particular state variable is difficult [61]. In the case of the real non-rigid joints, the state observer not only allows a better estimation of the variables that are not directly known, but also provides a reduction of the noise from sensor data. In SimTwo, all the variables can be computed reading the data from the physical simulation environment, eliminating the need for implementation of the observer.

4.4.1.3 Implementation in Simulation

SimTwo has built-in position and speed PID controllers, which were thus used to control the hip joints and wheels of the model. The controllers are set upon definition of the joint in the XML file that describes the robot, with the elements presented in Listing 4.1 and Listing 4.2.

```
<controller mode='pidposition' kp='4' ki='0' kd='3' kf='0.0'
  active='1' period='10'/>
```

Listing 4.1: Definition of the PID position controller.

```
<controller mode='pidspeed' kp='0.2' ki='0.04' kd='0' kf='0.55'
  active='1' period='10'/>
```

Listing 4.2: Definition of the PID speed controller.

Update of the references given to these controllers is made in the SimTwo IDE using the functions `SetAxisPosRef` and `SetAxisSpeedRef` for each of the hip joints and wheels, respectively.

The state-space controllers were also fully implemented within SimTwo using its IDE. This implementation required definition of custom data types (Listing 4.3) to characterise the controller gains and each of the non-rigid joints, and initialisation, for each joint, of Φ_a and Γ_a.

```
TGains = array[0..4] of double;

TJoint = record
   dx: integer; //index
   K: array[0..4] of double;
   theta_abs, theta_rel, w_abs, w_rel: double;
   theta_ref, u: double;
   phi_a, gamma_a, x_a: double;
end;
```

Listing 4.3: Custom data types for implementation of the state-space controller.

Regarding the `TJoint` type, `dx` is used to identify the non-rigid joint within the model, according to its automatically indexed value upon definition.

The procedure `set_k`, shown in Listing 4.4, is used to initialise the matrix of the controller gains, while `joint_control` (Listing 4.5) implements the controller of Figure 4.10, being responsible for sending the respective motor voltage according to the state update calculated.

```
procedure set_k(var joint: TJoint; K: TGains);
var i: integer;
begin
   for i := 0 to 4 do joint.K[i] := K[i];
end;
```

Listing 4.4: Procedure to set the gains of the state-space controller.

```
procedure joint_control(var J: TJoint);
var maxV: double;
begin
   { Joint data : state variables measurement }
   J.theta_rel := GetAxisPos(0, J.dx);
   J.theta_abs := GetAxisPos(0, J.dx + 1) + J.theta_rel;
   J.w_rel := GetAxisSpeed(0, J.dx);
   J.w_abs := GetAxisSpeed(0, J.dx + 1) + J.w_rel;

   J.x_a := J.phi_a * J.x_a + J.gamma_a * (J.theta_ref -
      J.theta_abs); //state update

   { Saturation block }
   maxV := 24;
   if J.x_a > maxV / J.K[4] then begin
       J.x_a := maxV / J.K[4];
   end else if J.x_a < -maxV / J.K[4] then begin
       J.x_a := -maxV / J.K[4]
   end;

   J.u := J.K[0] * J.theta_rel + J.K[1] * J.w_rel + J.K[2] *
      J.theta_abs + J.K[3] * J.w_abs;
   J.u := -J.u + J.x_a * J.K[4]; //system output

   SetAxisVoltageRef(0, J.dx, J.u * 1);
end;
```

Listing 4.5: State-space controller implementation in SimTwo.

4.4.2 High-Level Control

Autonomous operation of the robot requires implementation of control algorithms that rule its motion in the environment, so that it can succeed in the tasks it has to perform. For the simulation model assembled, a Lazarus application that runs its high-level control architecture was developed. This control architecture includes a trajectory tracking algorithm for motion control.

4.4.2.1 Application: Interaction with SimTwo

The Lazarus application that has been developed intends to replicate the role of the Teensy 4.1 board as the main microprocessor of the robotic system that has been modelled. This decision resulted from the fact that SimTwo is capable of communicating with remote programs using the UDP protocol (User Datagram Protocol) or through serial port. The former has been used.

Communication between the application, named as *LWRsimulation*, and the simulator allows information exchange. The message flow is outlined in Figure 4.11. The UDP packets are sent to SimTwo every 40 milliseconds.

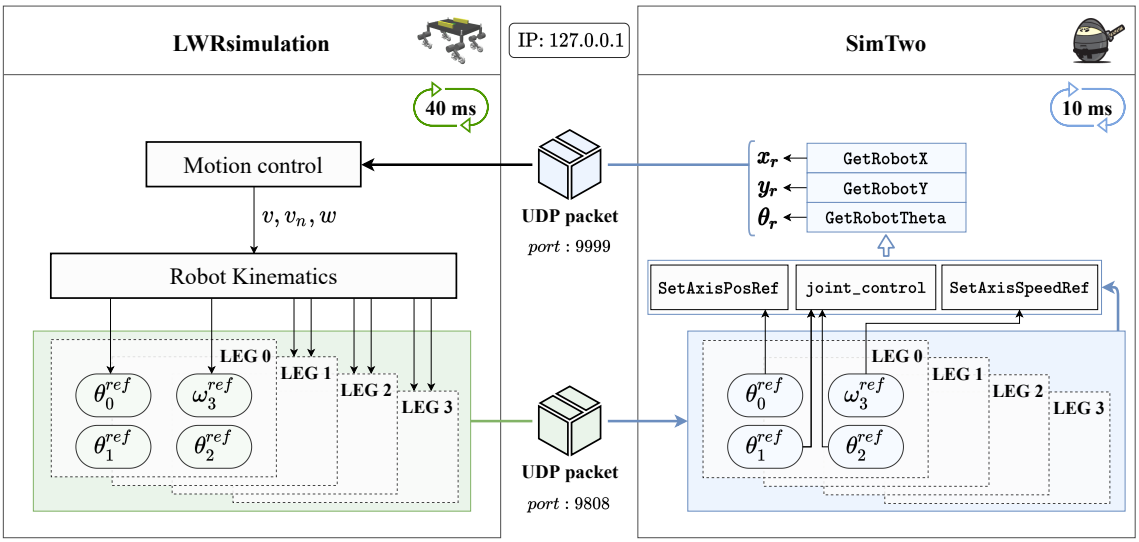


Figure 4.11: Interaction diagram of the developed application and SimTwo.

As can be interpreted from Figure 4.11, the application is responsible for generating the references of the angular position (rad) of the joints (θ_0^{ref}, θ_1^{ref}, and θ_2^{ref}) and the speed (rad/s) of the wheels (ω_3^{ref}) of each leg. These are sent to SimTwo, which in turn applies them to the controllers, and transmits the robot position back to the application. The robot position (in relation to the global frame) is characterised by the x_r (m) and y_r (m) coordinates and the robot's angle θ_r (rad), being directly acquired from the simulator environment using the functions `GetRobotX`, `GetRobotY` and `GetRobotTheta`.

Therefore, the application directs the execution of the algorithms for motion control of the robot, which compute the v (m/s), v_n (m/s) and ω (rad/s) speeds.

The kinematic model of the robot described in Section 3.3.1 was implemented in LWRsimulation to convert the speeds of the robot to the respective references and thus allow control in the x and y axes. Note shall be taken on the fact that, to maintain the realism of the model, restrictions are applied to the values of θ_0^{ref} and ω_3^{ref}, due to mechanical constraints of the real robot (as it has been mentioned in Section 3.2.1, the hip joint rotates from $-90°$ to $+90°$). If the calculated θ_0^{ref} does not belong to the interval $[-\pi/2,\ \pi/2]$ rad, it is normalised back to this range, while ω_3^{ref} is updated to its opposite.

Control in the z-axis is yet to be implemented. Nevertheless, as the current motion control techniques (details in Section 4.4.2.3) were developed for the robot to operate in purely wheeled mode, the reference for the angular position of joints 1 and 2 of each leg is, respectively, $\theta_1^{ref} = -65°$ and $\theta_2^{ref} = 130°$. These correspond to the values that allow keeping the robot at its lowest height. The controllers of these joints run for a period of 10 seconds after data transfer is detected and

initiated, guaranteeing that this position has been reached. They are then deactivated in the simulator, and the robot will thus operate in a low-power state.

4.4.2.2 Application Interface

The interface of the developed application is shown in Figure 4.12.

Figure 4.12: GUI of the application that communicates with the SimTwo simulator.

Motion control is activated only when *RUN* is clicked; otherwise the robot is stopped (null speed).

4.4.2.3 Motion Control

Two motion control modes are defined: *Manual* and *Auto*; the former refers to controlling the robot in simulation through keyboard inputs, while the latter to the (automatic) execution of motion control algorithms. These are associated with the "Motion Control" block of Figure 4.11.

Manual Control

The robot in simulation can be manually controlled through keyboard inputs. v is set by the `up` and `down` arrow keys, while v_n by the `left` and `right` arrows; ω is set with the `tab` and `back space` keys. Figure 4.13 shows the GUI state that allows performing manual control.

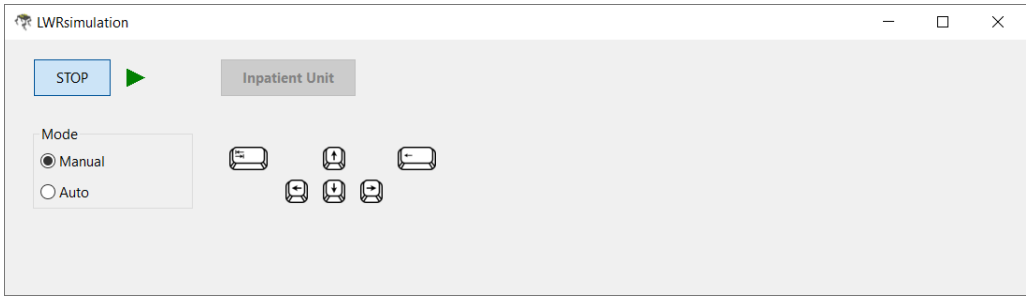


Figure 4.13: GUI state for manual control of the robot.

Motion Control Algorithms: Trajectory Tracking

The purpose of a trajectory tracking algorithm is for the robot to follow a well-characterised trajectory.

The methods that have been implemented require the robot to follow a line or a curve from an initial point, with coordinates $(x_i,\ y_i)$, to a target point, with coordinates $(x_f,\ y_f)$.

The developed trajectory tracking algorithm can be represented with a state machine comprising two states, as shown in Figure 4.14.

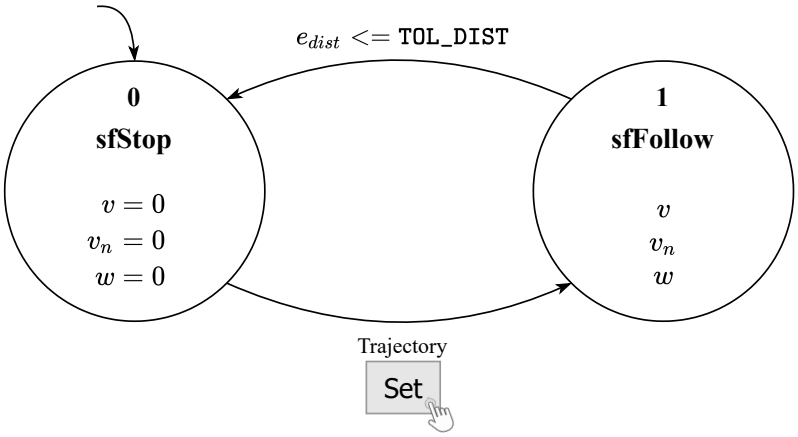


Figure 4.14: State machine diagram of the trajectory tracking framework.

To attain its goal, the robot must thus travel along the trajectory, which is performed when the state corresponds to `sfFollow`. Execution of this task involves the use of controllers that correct the robot's position. Taking advantage of the omnidirectional motion ability of this robot, these controllers adjust the position with both lateral and angular motion.

Transitioning from state 1 to state 0 indicates that the robot has reached the target point. `TOL_DIST` is a constant that sets the admissible tolerance for how close the robot must be to the target to stop; it was defined to be equal to 3 cm.

The variable e_{dist} (m) evaluates how close the robot is to the target point and is calculated according to Equation (4.10).

$$e_{dist} = \sqrt{\left(x_r^2 - x_f^2\right) + \left(y_r^2 - y_f^2\right)} \tag{4.10}$$

The robot follows the trajectory with constant linear speed v_{nom} (m/s) up to a defined distance to the target, where it initiates a smooth linear deceleration motion to reach and stop at the final position. This behaviour is represented in Figure 4.15.

The values of v_{nom} and $v_{f_{de\_accel}}$ (which corresponds to the robot's linear speed, in m/s, when it gets to the target) were defined to be 0.70 m/s and 0.20 m/s, respectively.

In Figure 4.15, the x-axis variable l (m) relates to e_{dist} through Equation (4.11).

$$l = -e_{dist} \tag{4.11}$$

Hence, l_{max} (m) is the opposite value of the maximum distance to the target, which means it is associated with e_{dist} at the robot's start position. l_t (m) is a constant that defines where the transition occurs, so that deceleration is initiated; its value depends on the type of trajectory the robot is to follow.

Equation (4.12) defines v as represented in Figure 4.15.

$$v = \begin{cases} v_{nom} & l \leq l_t \\ v_{de\_accel} & l > l_t \end{cases} \tag{4.12}$$

Taking into consideration the relation in Equation (4.11), the robot's linear speed is determined by Equation (4.13).

$$v = \begin{cases} v_{nom} & e_{dist} \geq -l_t \\ \left(\frac{v_{f_{de\_accel}} - v_{nom}}{l_t} \right) e_{dist} + v_{f_{de\_accel}} & e_{dist} < -l_t \end{cases} \tag{4.13}$$

The characterisation of the trajectory and the specific controllers used for line tracking and curve tracking, as well as the GUI state to set the trajectories, are described below.

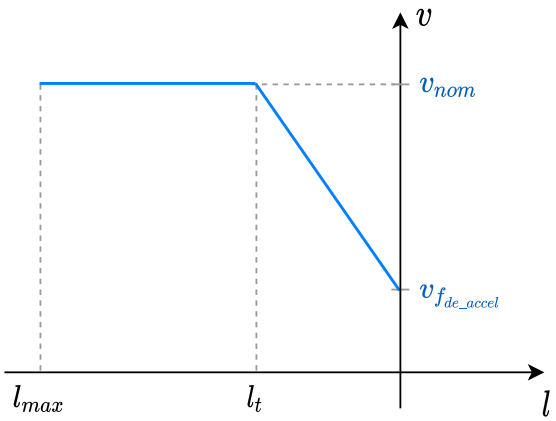


Figure 4.15: Graphical representation of the robot's linear speed v along the trajectory.

Line Tracking A line is defined by a set of two points, which thus correspond to the initial and target points. As shown in Figure 4.16, the user inputs these points in the GUI and, when *Set* is pressed, the robot will begin to move and follow the trajectory defined. The charts plot, as the robot moves, both the robot's position and the trajectory itself, so that its motion can be visualised in the application.

Figure 4.17 pictures the typical problem a line tracking algorithm must solve: keep the robot on the line. The position of the robot thus needs to be corrected, so that it converges to the trajectory.

The inclination of the line, α (rad), is defined in Equation (4.14).

$$\alpha = \text{atan2}\,(y_f - y_i,\ x_f - x_i) \tag{4.14}$$

The vector equation in Equation (4.15) characterises the line.

$$(x,\ y) = (x_i,\ y_i) + k_{line}\ \hat{u} \tag{4.15}$$

The vector $\hat{u}$ is a unit vector, with components u_x and u_y, indicated in Equation (4.16), pointing to the target position.

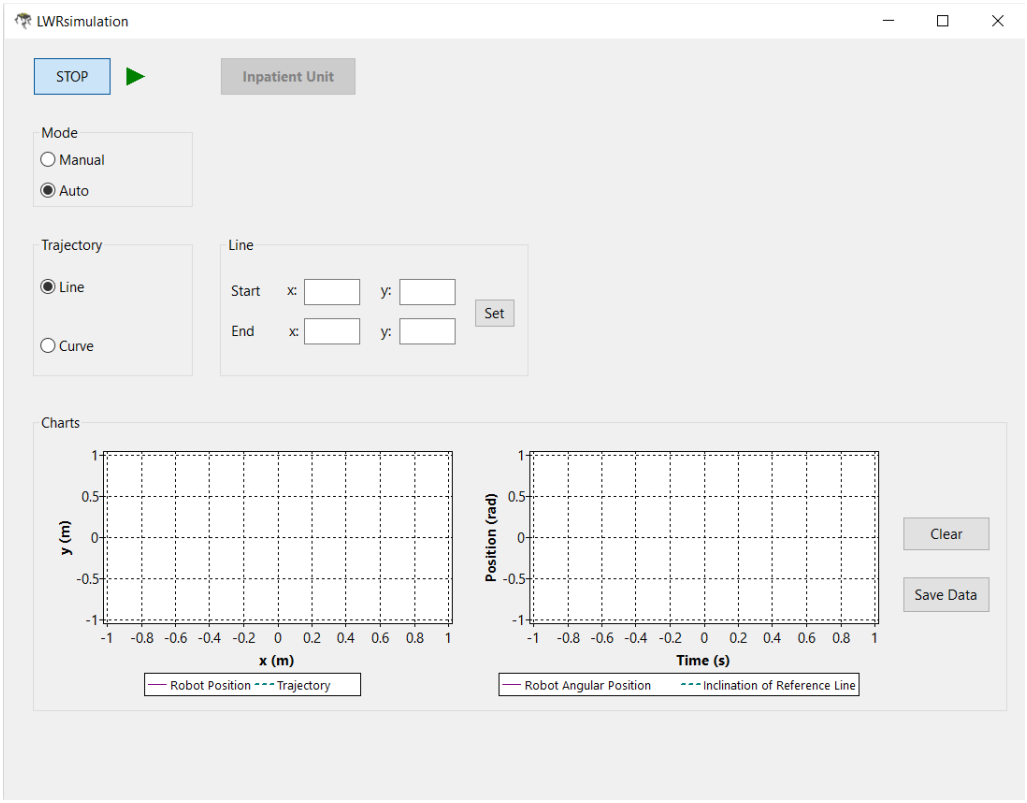


Figure 4.16: GUI state for defining the line that is to be followed by the robot.

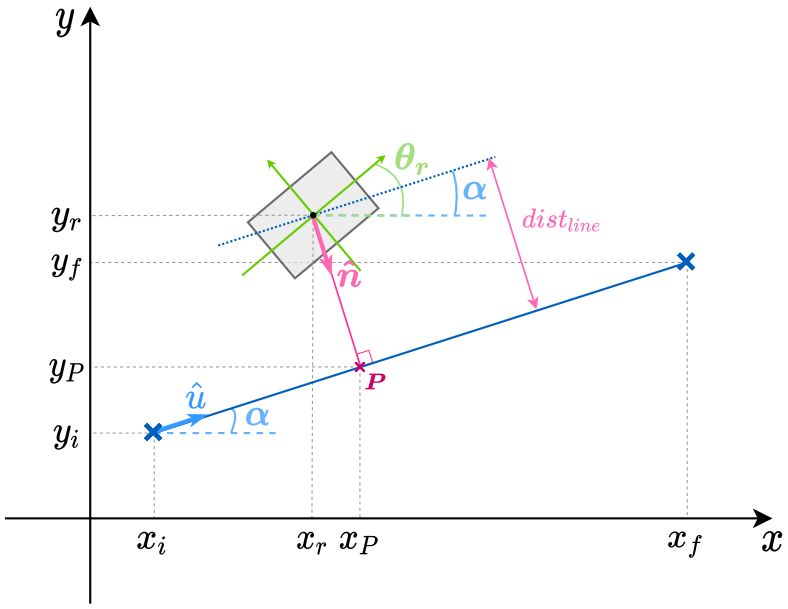


Figure 4.17: Representation of the situation that must be answered by a line tracking algorithm to keep the robot on the trajectory (the rectangle represents the main body of the robot, and the trajectory is the solid blue line; the global frame corresponds to the $x-y$ frame, while the robot's frame is represented in green).

$$
\begin{aligned}
u_x &= \frac{x_f - x_i}{\sqrt{\left(x_f^2 - x_i^2\right) + \left(y_f^2 - y_i^2\right)}} \\
u_y &= \frac{y_f - y_i}{\sqrt{\left(x_f^2 - x_i^2\right) + \left(y_f^2 - y_i^2\right)}}
\end{aligned}
\tag{4.16}
$$

Precise line tracking required the implementation of P controllers that adjust v_n and ω, allowing correct positioning of the robot on the line. Equations (4.17) and (4.18) define the implemented control laws for v_n and w, respectively.

$$v_n = K_{v_n} \cdot dist_{line} \tag{4.17}$$

$$\omega = K_{\omega} \cdot e_{\theta} \tag{4.18}$$

These controllers take characteristics of the trajectory itself as reference signals. The error signal of the controller in v_n, $dist_{line}$ (m), is related to the shortest distance between the robot and the line, while that of the controller in ω, e_{θ} (rad), measures the difference between the angular position of the robot and the inclination of the line. The error signals are calculated using Equations (4.19) and (4.20).

$$dist_{line} = \frac{(y_i - y_r)\,u_x - (x_i - x_r)\,u_y}{u_x^2 + u_y^2} \tag{4.19}$$

$$e_{\theta} = \alpha - \theta_r \tag{4.20}$$

The value of e_{θ} is always normalised to the range $[-\pi, \pi]$ rad.

Deriving Equation (4.19) requires considering the system of equations in Equation (4.21), which allows determining the coordinates of point P (Figure 4.17). P is the nearest point to the robot on the trajectory, and thus $\hat{n}$ is a unit vector perpendicular to $\hat{u}$ (pointing to the trajectory) that defines the line passing through the robot and P. The components of $\hat{n}$ can be written in relation to $\hat{u}$: the x-component is $-u_y$ and the y-component u_x.

$$
\begin{cases}
(x_P,\, y_P) = (x_i,\, y_i) + k_{line}\,\hat{u} \\
(x_P,\, y_P) = (x_r,\, y_r) + dist_{line}\,\hat{n}
\end{cases}
=
\begin{cases}
(x_P,\, y_P) = (x_i,\, y_i) + k_{line}\,(u_x,\, u_y) \\
(x_P,\, y_P) = (x_r,\, y_r) + dist_{line}\,(-u_y,\, u_x)
\end{cases}
\tag{4.21}
$$

To obtain Equation (4.19) and calculate $dist_{line}$, Equation (4.21) is rearranged into Equation (4.22).

$$
\begin{cases}
x_i + k_{line}\,u_x = x_r + dist_{line}\,(-u_y) \\
y_i + k_{line}\,u_y = y_r + dist_{line}\,(u_x)
\end{cases}
\tag{4.22}
$$

The value of the gain of each controller, as well as the value of the previously mentioned l_t, is given in Table 4.7.

Table 4.7: Gains of the P controllers and the l_t value of the line tracking algorithm.

K_{v_n}	K_ω	l_t
7.00	0.80	−0.20

Curve Tracking To define the curve the robot must follow, a set of three points is used. These correspond to the initial and target points, and to a third point between the two, with coordinates $(x_{int},\, y_{int})$. The points are inserted in the GUI, as presented in Figure 4.18, setting the robot movement to track the trajectory defined. As referred for the line tracking GUI, the robot's motion is visualised in the application through the charts that display its position and the trajectory.

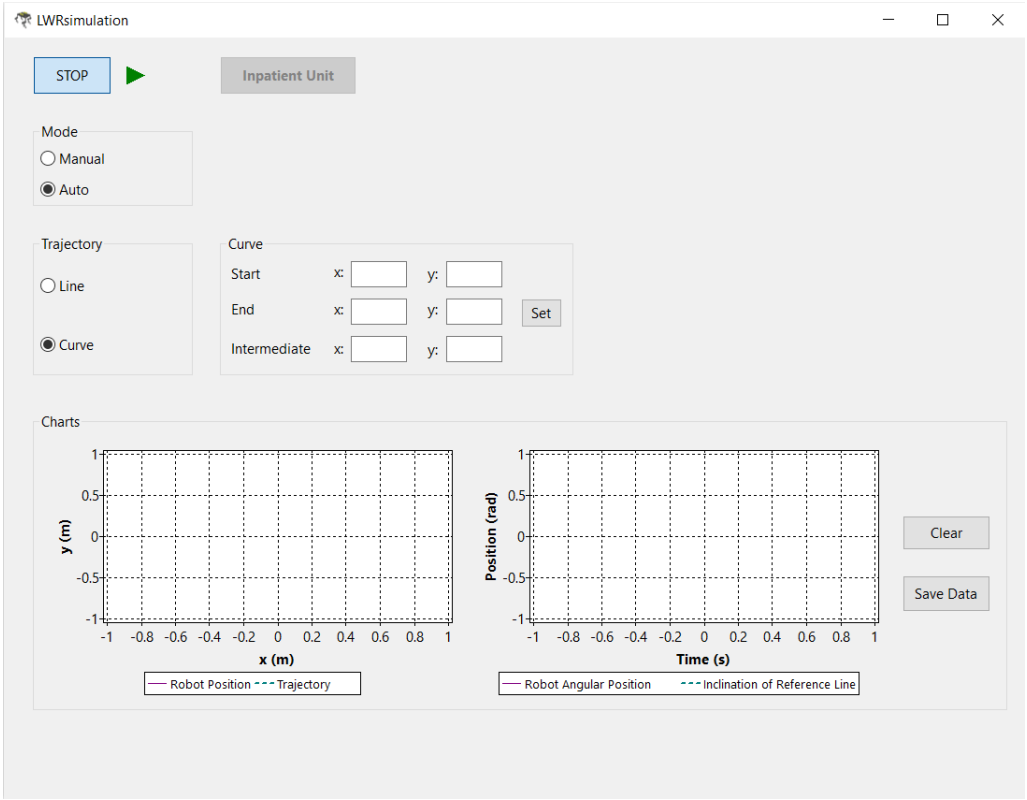


Figure 4.18: GUI state for defining the curve that is to be followed by the robot.

The defined curve-shaped trajectory is an arc of a circle between the initial and final points. To characterise the trajectory, the centre and the radius of the circle thus have to be calculated.

To simplify the equations written hereafter, the subscripts identifying each point of the trajectory are substituted by the subscripts *1*, *2* and *3*, according to the relations (i) to (iii).

$$(x_i,\, y_i) \longrightarrow (x_1,\, y_1) \tag{i}$$

$$(x_{int},\, y_{int}) \longrightarrow (x_2,\, y_2) \tag{ii}$$

$$(x_f,\, y_f) \longrightarrow (x_3,\, y_3) \tag{iii}$$

The standard form of the equation of a circle with centre $(x_c,\ y_c)$ and radius r (m) is given in Equation (4.23), and the general form in Equation (4.24).

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{4.23}$$

$$A\left(x^2 + y^2\right) + B\,x + C\,y + D = 0 \tag{4.24}$$

The relation between the constants A, B, C and D, and the circle's centre and radius is presented in Equations (4.25) to (4.27).

$$x_c = -\frac{B}{2A} \tag{4.25}$$

$$y_c = -\frac{C}{2A} \tag{4.26}$$

$$r = \sqrt{\frac{B^2 + C^2 - 4AD}{4A^2}} \tag{4.27}$$

Substituting the coordinates of the three points that lie on the circle into Equation (4.24), the constants A, B, C and D are computed from Equations (4.28) to (4.31).

$$A = x_1\,(y_2 - y_3) - y_1\,(x_2 - x_3) + x_2\,y_3 - x_3\,y_2 \tag{4.28}$$

$$B = \left(x_1^2 + y_1^2\right)(y_3 - y_2) + \left(x_2^2 + y_2^2\right)(y_1 - y_3) + \left(x_3^2 + y_3^2\right)(y_2 - y_1) \tag{4.29}$$

$$C = \left(x_1^2 + y_1^2\right)(x_2 - x_3) + \left(x_2^2 + y_2^2\right)(x_3 - x_1) + \left(x_3^2 + y_3^2\right)(x_1 - x_2) \tag{4.30}$$

$$D = \left(x_1^2 + y_1^2\right)(x_3\,y_2 - x_2\,y_3) + \left(x_2^2 + y_2^2\right)(x_1\,y_3 - x_3\,y_1) + \left(x_3^2 + y_3^2\right)(x_2\,y_1 - x_1\,y_2) \tag{4.31}$$

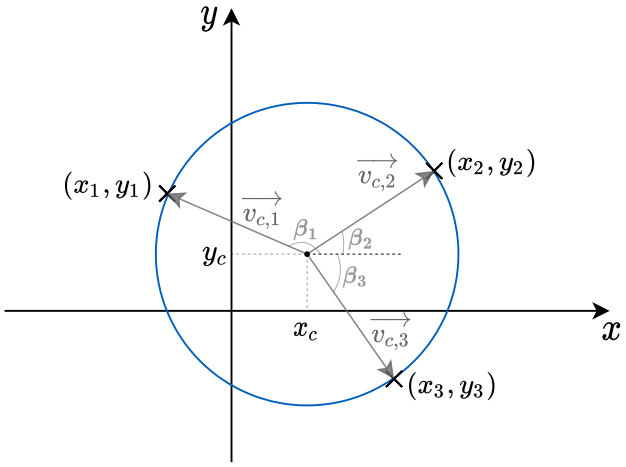


Figure 4.19: Representation of the vectors defined from the circle's centre to each point on the circle and respective directions.

Since the robot aims to travel from an initial point to a target point and pass an intermediate point, the direction of rotation must be defined. For that purpose, the components of the vectors $\overrightarrow{v_{c,1}}$, $\overrightarrow{v_{c,2}}$ and $\overrightarrow{v_{c,3}}$ (vectors between the circle's centre and each point) are calculated, as well as their

directions β_1, β_2 and β_3 (in radians), applying the *atan2* function to the components. Figure 4.19 depicts the entities just mentioned.

The intended direction of rotation d_R is set from evaluating the angles presented in Equations (4.32) and (4.33) (γ_1 and γ_2 are normalised to the range $[0,\ 2\pi]$ rad). If $\gamma_2 > \gamma_1$, the robot travels clockwise ($d_R = 1$); otherwise, it will be counterclockwise ($d_R = -1$).

$$\gamma_1 = \beta_2 - \beta_1 \tag{4.32}$$

$$\gamma_2 = \beta_3 - \beta_1 \tag{4.33}$$

Figure 4.20 represents the typical problem the curve tracking algorithm must solve.

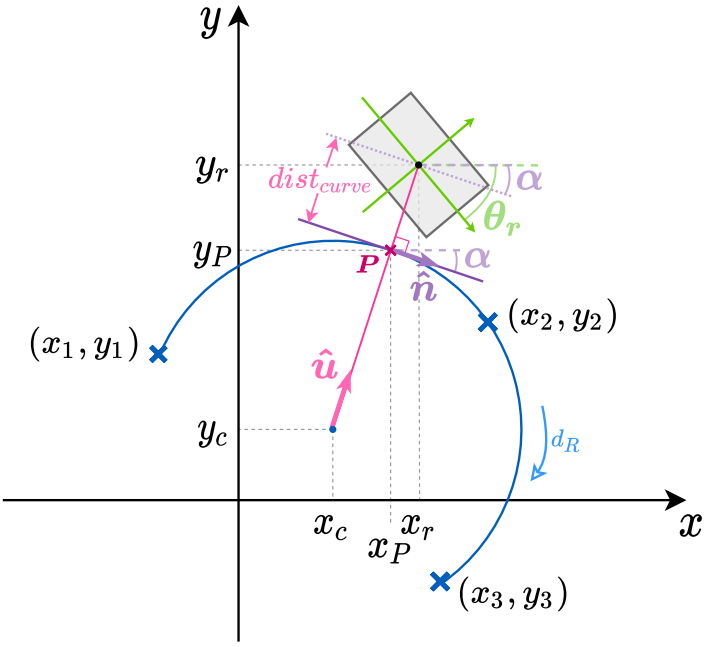


Figure 4.20: Representation of the situation that must be answered by a curve tracking algorithm to keep the robot on the trajectory (the rectangle represents the main body of the robot, and the trajectory is the solid blue curve; the global frame corresponds to the $x-y$ frame, while the robot's frame is represented in green).

Provided that the robot must accurately follow the trajectory, PD controllers were implemented to adjust v_n and ω. The latter also includes a feedforward term, ω_{nom} (rad/s) that allows the robot to accompany the curvature of the circle; ω_{nom} and v are related through Equation (4.34).

$$\omega_{nom} = \left(\frac{v}{r}\right) \cdot d_R \tag{4.34}$$

The control laws for v_n and ω are given in Equations (4.35) and (4.36), respectively, where T_d corresponds to the derivative time (tuning parameter of the controller).

$$v_n = K_{v_n}\left(dist_{curve} + T_{d_{v_n}}\frac{\mathrm{d}\, dist_{curve}}{\mathrm{d}t}\right) \cdot (-d_R) \tag{4.35}$$

$$\omega = \omega_{nom} + K_\omega\left(e_\theta + T_{d_\omega}\frac{\mathrm{d}\, e_\theta}{\mathrm{d}t}\right) \tag{4.36}$$

The line passing through the circle's centre and the robot (solid pink line in Figure 4.20), which contains P (nearest point to the robot on the curve), is characterised by the unit vector $\hat{u}$, whose components u_x and u_y are extracted from Equation (4.37) ($\hat{u}$ points to the robot's position).

$$\hat{u} = \left(\frac{x_r - x_c}{\sqrt{(x_r^2 - x_c^2) + (y_r^2 - y_c^2)}}, \frac{y_r - y_c}{\sqrt{(x_r^2 - x_c^2) + (y_r^2 - y_c^2)}} \right) \tag{4.37}$$

Equations (4.38) and (4.39) present two possible ways of representing the vector equation of the referred line.

$$(x_P, y_P) = (x_c, y_c) + r\,\hat{u} \tag{4.38}$$

$$(x_P, y_P) = (x_r, y_r) + dist_{curve}\,\hat{u} \tag{4.39}$$

The coordinates of P are computed from Equation (4.38), allowing to determine the error signal of the controller in v_n, $dist_{curve}$ (m). This variable is related to the distance between the robot and the curve, and can be calculated either from Equation (4.40) or Equation (4.41) (the choice lies in the one that does not give rise to a division by zero).

$$dist_{curve} = \frac{x_P - x_r}{u_x} \tag{4.40}$$

$$dist_{curve} = \frac{y_P - y_r}{u_y} \tag{4.41}$$

The error signal of the controller in ω, e_θ (rad), measures the difference between the angular position of the robot and the inclination of the tangent to the curve at point P (solid purple line in Figure 4.20), α (rad). The tangent is characterised by the unit vector $\hat{n}$, perpendicular to $\hat{u}$. The direction of $\hat{n}$, set by Equation (4.42), depends on the direction of rotation.

$$\hat{n} = (n_x, n_y) = \begin{cases} (-u_y, u_x) & d_R = 1 \\ (u_y, -u_x) & d_R = -1 \end{cases} \tag{4.42}$$

The inclination of the tangent is defined by Equation (4.43).

$$\alpha = \text{atan2}\,(n_y, n_x) \tag{4.43}$$

The error signal e_θ is calculated using Equation (4.44).

$$e_\theta = \alpha - \theta_r \tag{4.44}$$

The parameters of the PD controllers implemented, as well as the value of the previously mentioned l_t, is given in Table 4.8.

Table 4.8: Parameters of the PD controllers and the l_t value of the curve tracking algorithm.

K_{v_n}	$T_{d_{v_n}}$	K_ω	T_{d_ω}	l_t
2.40	1.70	3.50	1.15	−0.45

4.5 Simulation Model Validation

Validation of the developed simulation model required comparing it to data acquired from the real robotic system. Namely, the elastic behaviour of the non-rigid joints was evaluated.

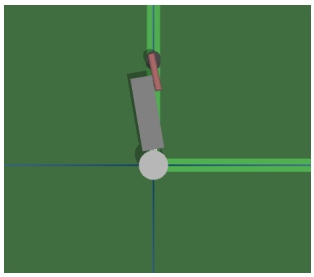
A simple setup (Figure 4.21a) to replicate the test described in [56] was built in the simulator. This comprised a single (non-actuated) joint connecting two links, with one of them fixed; this joint was set as a spring.

Table 4.9: Parameters of the simulated spring-mass system.

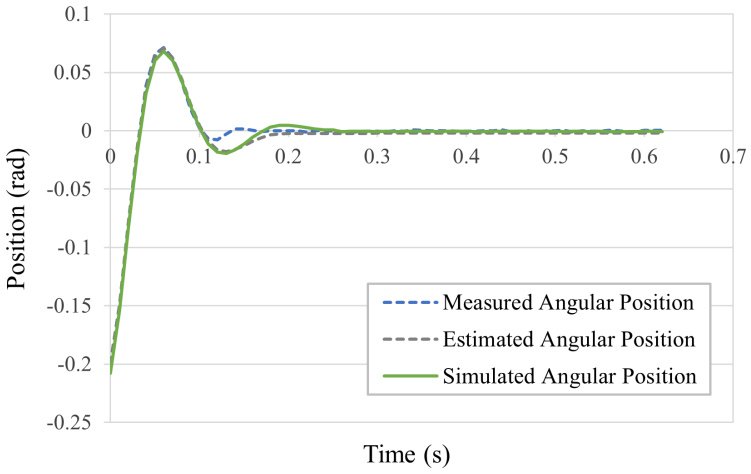
Mass	B_v	f_C	k_s
0.505 kg	0.239 N·m·s	0.018 N·m	16.92 N/m

For this spring-mass system, the spring constant k_s determined in [56] was used. B_v was defined as the viscous friction coefficient of the rotary damper included in the real joint tested in the referred publication. Table 4.9 summarises the parameters of the system.

The oscillatory response of the system was observed by pulling the free link up to a defined position and releasing it (the pull motion was performed by another link driven by a generic motor). The obtained results are shown in Figure 4.21b.



(a)



(b)

Figure 4.21: (a) Simulator setup for observation of the oscillatory response of the non-rigid joint. (b) Angular position (real vs. estimated vs. simulated).

The settling time of the system is approximately 200 ms (considering 2% range) and the 98% rise time around 30 ms. The data collected in the simulation (shown in green) was compared against the results of the test performed on the real non-rigid joint and its modelled response (data published in [56]). The mean absolute errors of the position in simulation compared to the estimated and real positions are, respectively, 0.0025 and 0.0023 rad, while the maximum absolute errors are 0.0084 and 0.0181 rad. The reason the obtained response in simulation is closer to the modelled

one, instead of the real one, is related to the uncertainties that derive from data acquisition in real settings, which might not be modelled in a simulated environment (e.g., load cell noise).

4.6 Results

This section is dedicated to the results associated with the simulation of the robot.

4.6.1 State-Space Controller

The controller of the non-rigid joints (described in Section 4.4.1.2) was evaluated, while keeping the main body of the robot fixed, by changing the reference position of the joint from 0 to 1 rad.

The values of the controller gains in Section 4.4.1.2 refer to a tuning of the controller, which is explained hereafter.

In [54], Pinto *et al.* presented the estimated values for the controller gains (Equation (4.45)), which were thus first used in the controller implemented in simulation.

$$K = \begin{bmatrix} 10.7110 \\ 0.5160 \\ -2.3263 \\ 0.4410 \\ 0.3118 \end{bmatrix} \tag{4.45}$$

The steady state response was registered and is shown in Figure 4.22, which plots the angular position of one of the upper non-rigid joints (joint 1) over time. Considering a 2% range, the joint exhibits an overshoot of approximately 14.7%.

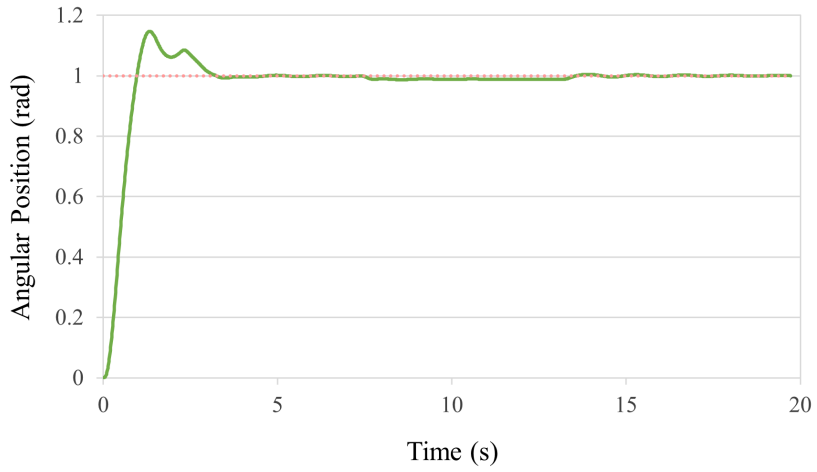


Figure 4.22: State-space controller (K of Equation (4.45)), with step reference (dotted line indicates the reference position).

To decrease the overshoot and settling time of the joint and hence increase its stability to allow smooth transitions between positions, the controller gains were adjusted to the values presented in Equation (4.9). The obtained response is shown in Figure 4.23.

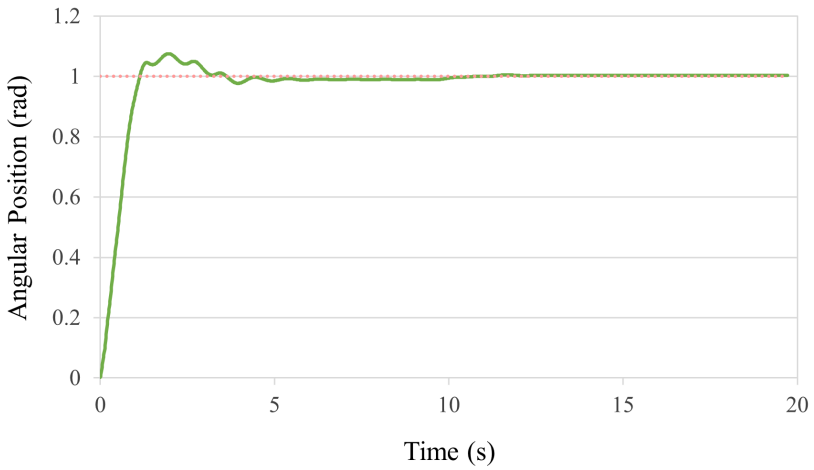


Figure 4.23: State-space controller (K of Equation (4.9)), with step reference (dotted line indicates the reference position).

The overshoot decreased to approximately 7.53%. As the joint has not been updated to the improved version and the response is satisfactory, the controller was not further refined.

4.6.2 Reactive Obstacle Surpassing

The reactive obstacle surpassing capability of the robot is realised through combination of wheel motion and elasticity of the non-rigid joints. Therefore, while in wheeled mode, this feature was analysed, allowing to observe the compliant leg behaviour achieved with passive non-rigid joints.

The setup built in the simulator, shown in Figure 4.24, comprised two cuboid obstacles with equal size (0.200×0.400 m, L×W); the respective heights of obstacles *1* and *2* are 4 cm and 6 cm.

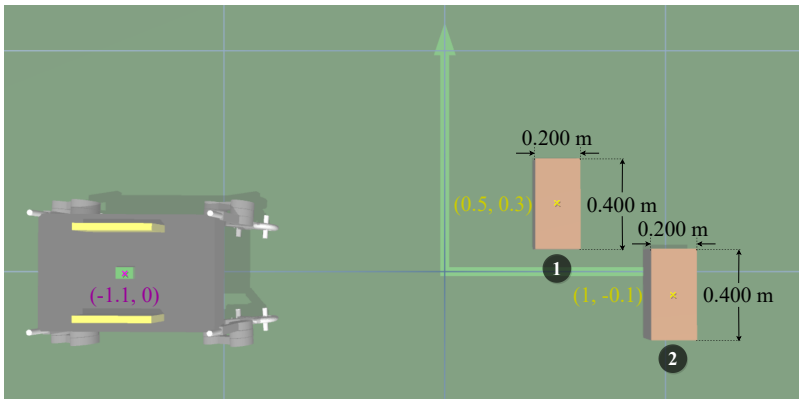
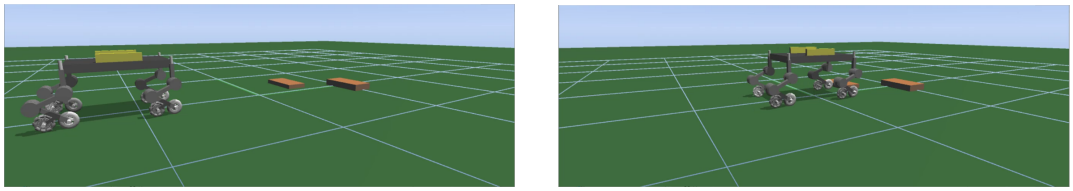
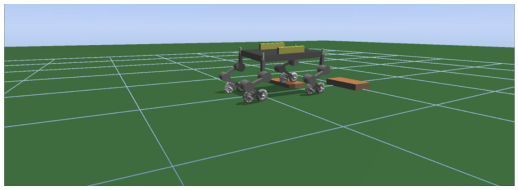


Figure 4.24: Simulator setup for assessment of the obstacle climbing ability of the robot.

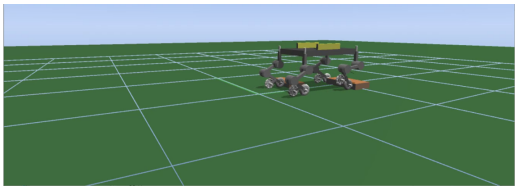


(a) $t = 0$ s

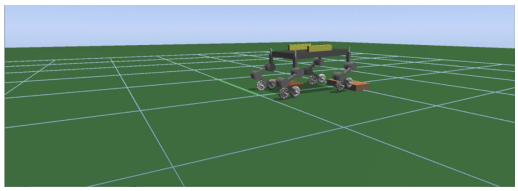
(b) $t = 2.44$ s



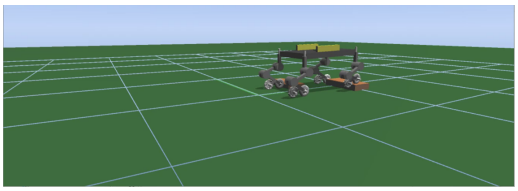
(c) $t = 2.88$ s



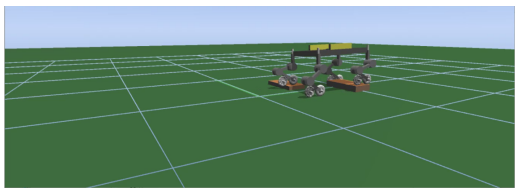
(d) $t = 3.82$ s



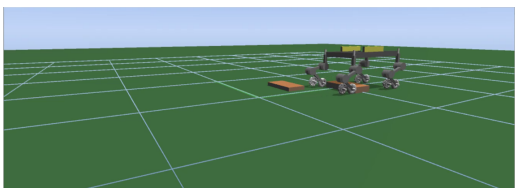
(e) $t = 4.96$ s



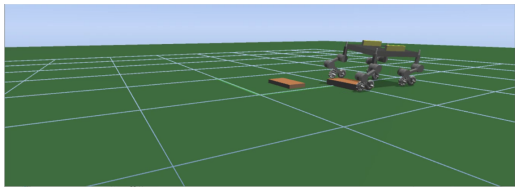
(f) $t = 5.62$ s



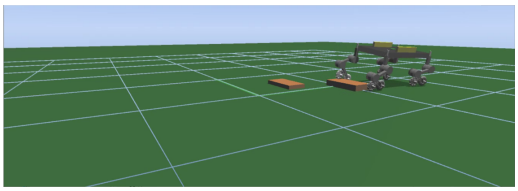
(g) $t = 6.28$ s



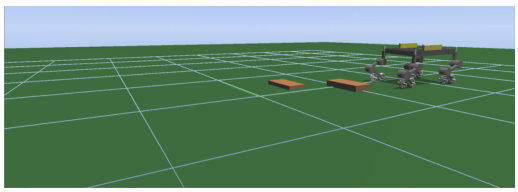
(h) $t = 6.84$ s



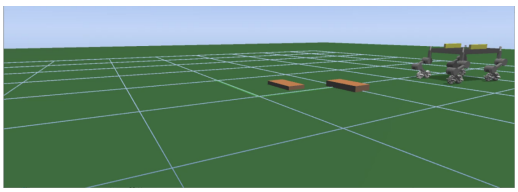
(i) $t = 8.98$ s



(j) $t = 9.22$ s



(k) $t = 9.80$ s



(l) $t = 10.8$ s

Figure 4.25: Sequence of snapshots (at timestamp t) of obstacle climbing.

Besides the size of the obstacles, Figure 4.24 identifies the position of each obstacle and the initial position of the robot in the environment (x and y coordinates).

Within a timeframe of 10.8 seconds, the robot was able to overcome both obstacles. Figure 4.25 contains a series of snapshots showing the robot performance.

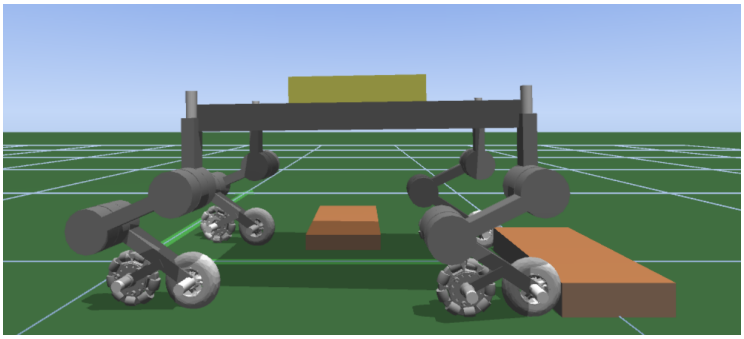
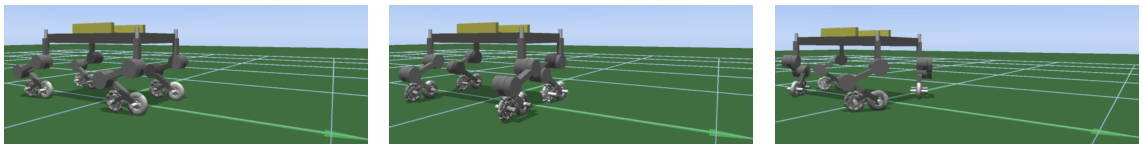


Figure 4.26: Robot colliding with obstacle *2*, illustrating the proportion between the wheel radius and the obstacle's height.

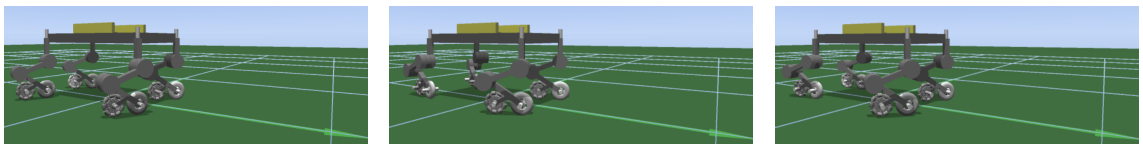
It is thus validated that the robot is able to climb up obstacles that are up to 1.2 times higher than the wheels' radius (Figure 4.26), exceeding the capabilities of a traditional wheeled vehicle.

4.6.3 Control in the x and y Axes

The implementation of $x-y$ control of the robot was first assessed by inputting a set of values for v, v_n and w (this test required fixing the main body in the simulation environment). Figure 4.27 shows the obtained positions of the robot legs. It can be visually determined that, according to the speed references given to the robot, the legs are positioned as expected.



(a) v=0.5 v_n=0 w=0 **(b)** v=0 v_n=0.5 w=0 **(c)** v=0 v_n=0 w=0.5



(d) v=0.5 v_n=0.5 w=0 **(e)** v=0.5 v_n=0 w=1.0 **(f)** v=0.5 v_n=0.5 w=1.0

Figure 4.27: Control in the x and y axes: setting the speeds v (m/s), v_n (m/s) and w (rad/s) of the robot (all images were taken from the same perspective; the axis in the foreground corresponds to the x-axis).

4.6.4 Trajectory Tracking

To analyse the performance of the developed trajectory tracking technique, described throughout Section 4.4.2.3, data regarding two possible trajectories (one of each type) was acquired. In both

cases, the robot started from the same position, with coordinates $(-0.601, 0.266)$, and with the same angle, $-10°$.

The characteristics of the line defined are presented in Table 4.10, while those of the curve in Table 4.11.

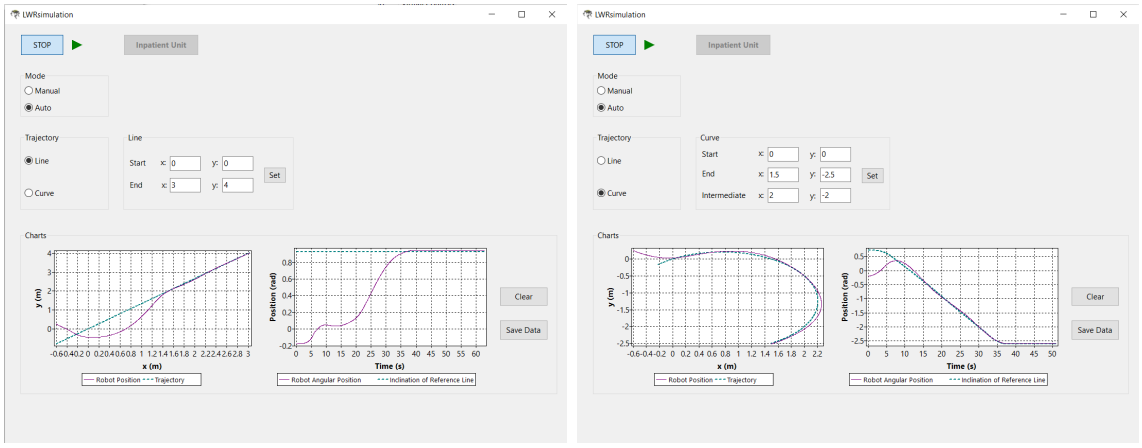
Table 4.10: Characteristics of the simulated trajectory: line.

Line		
Initial position (m)	Target position (m)	Inclination (rad)
(0, 0)	(3.0, 4.0)	0.927

Table 4.11: Characteristics of the simulated trajectory: curve.

Curve			
Initial position (m)	Target position (m)	Centre (m)	Radius (m)
(0, 0)	(1.5, −2.5)	(0.75, −1.25)	1.458

Figure 4.28a and Figure 4.28b show the state of the application interface when the robot stops at the target position. The charts in these figures, which are displayed while the trajectory tracking algorithm is executing, are replicated in Figures 4.29(a–b) and 4.29(c–d).



(a) **(b)**

Figure 4.28: Display of the results of the trajectory tracking algorithm in the GUI. (a) Line tracking. (b) Curve tracking.

Considering the data plotted in Figure 4.29, it is concluded that the robot is able to converge to the trajectory it is given. The goal of stopping at the target was also achieved.

For the line and curve that have been tested, the variation in time of the error in the robot's distance to the trajectory, $dist_{line}$ and $dist_{curve}$, respectively, and the error in its angular position, e_θ, are presented in Figure 4.30.

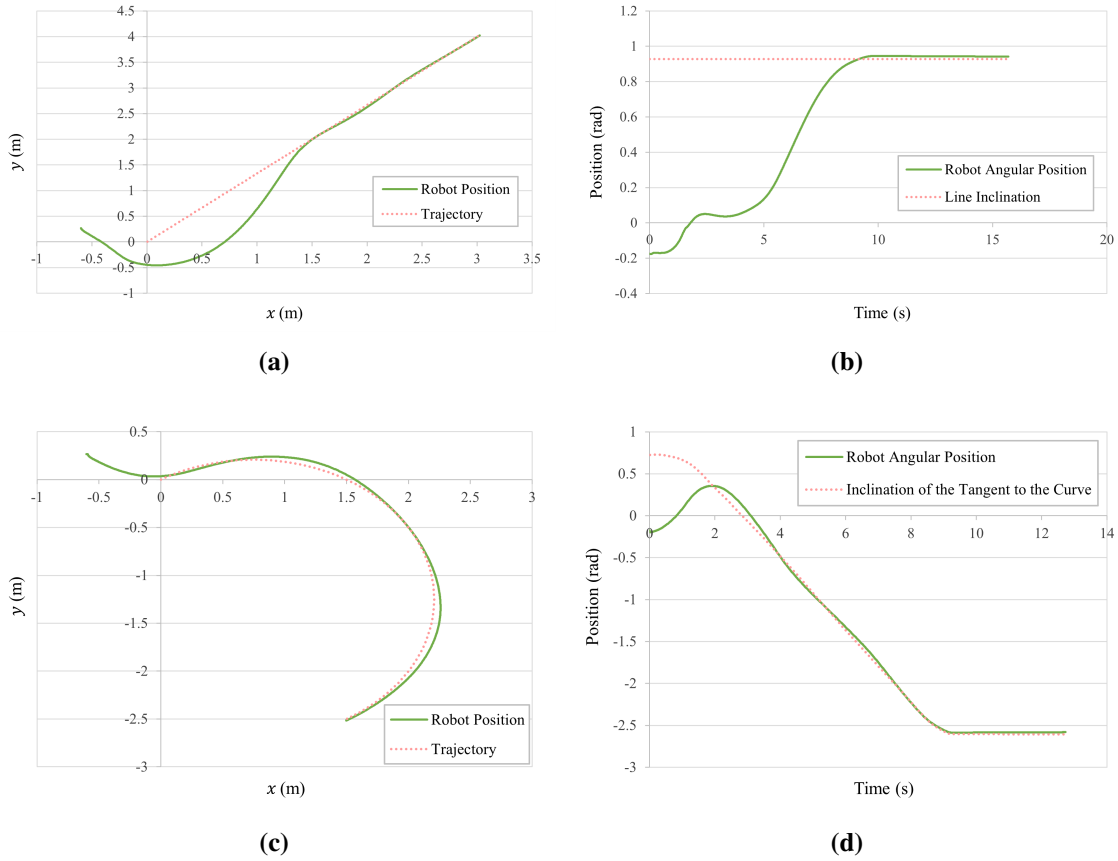


Figure 4.29: Charts displaying the results of the trajectory tracking algorithm executed with a line and a curve. Line tracking: plots of the (a) 2D position (m), and (b) angular position (rad) of the robot. Curve tracking: plots of the (c) 2D position (m), and (d) angular position (rad) of the robot.

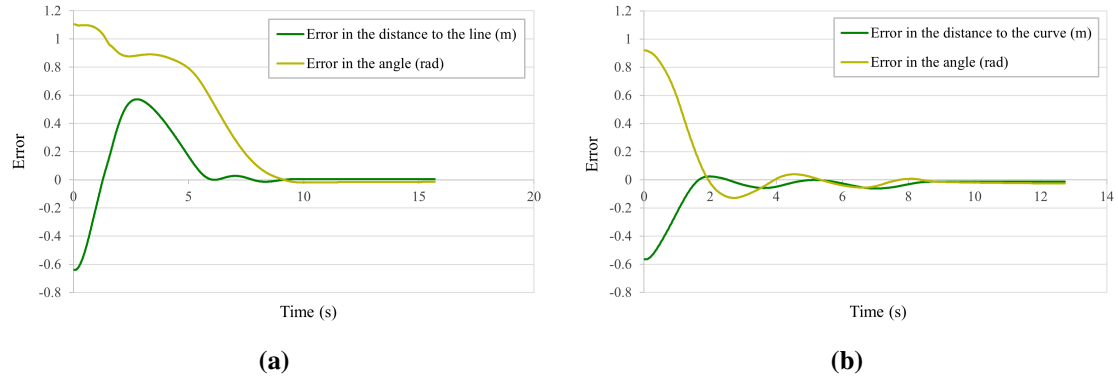


Figure 4.30: Error variation with time, associated with the (a) line and b) curve tracking algorithms, of the error in the distance to the trajectory ($dist_{line}$ and $dist_{curve}$, respectively) and in the angular position (e_θ).

At the final position, the errors in the distance to the target (e_{dist}) and in the angular position (e_θ), for the trajectories tested, are listed in Table 4.12.

Taking into account the complete motion of the robot (from its start to end position), the

maximum absolute error to the trajectory, in the distance, was 0.639 and 0.565 m, respectively for the line and the curve defined. These errors decrease to 0.572 and 0.062 m, when only the motion from the moment the robot passes the initial position of the trajectory is considered.

Table 4.12: Absolute errors in the distance and angular position, associated with the target point.

	Target position (m)	e_{dist} (m)	**Target angle** (rad)	e_{θ} (rad)
Line	(3.0, 4.0)	0.009	0.927	0.015
Curve	(1.5, −2.5)	0.012	−2.608	0.026

The average values of the error in the distance to the trajectory ($dist_{line}$ and $dist_{curve}$) and in the robot's angular position (e_{θ}), once the robot converges to the trajectory, are presented in Table 4.13.

Table 4.13: Average errors of the robot's 2D and angular positions in relation to the trajectory, after convergence.

	Average		
	$dist_{line}$ (m)	$dist_{curve}$ (m)	e_{θ} (rad)
Line	0.004	—	−0.013
Curve	—	−0.025	−0.024

The small errors reported allow to conclude that the methods developed are a viable strategy for the motion of the vehicle in its workspace.

4.6.4.1 Case Study

A real-scale inpatient unit was projected in the simulator to evaluate the approach developed for trajectory tracking. An inpatient unit is a department that provides care to patients that need to be hospitalised [62].

The dimensions of the corridors, doors, rooms and beds of this simulated inpatient unit, shown in Figure 4.31, are based on guidelines and documents provided by the International Health Facility Guidelines [62] and the Department of Health of the United Kingdom [63].

According to the room identification in Figure 4.32, the dimensions considered to build the department are presented in Table 4.14.

As represented in Figure 4.33, the clearance (free space) around each bed was a factor taken into consideration to design the simulated department.

Analysis of the performance of the developed trajectory tracking algorithm required defining a path to be followed by the robot. The path is characterised by a set of points, plotted in Figure 4.34 that identify each trajectory type. The sequence of trajectories is presented in Table 4.15. Since a path is being inputted, the admissible tolerance for how close the robot must be to the target to stop was adjusted to 21 cm.

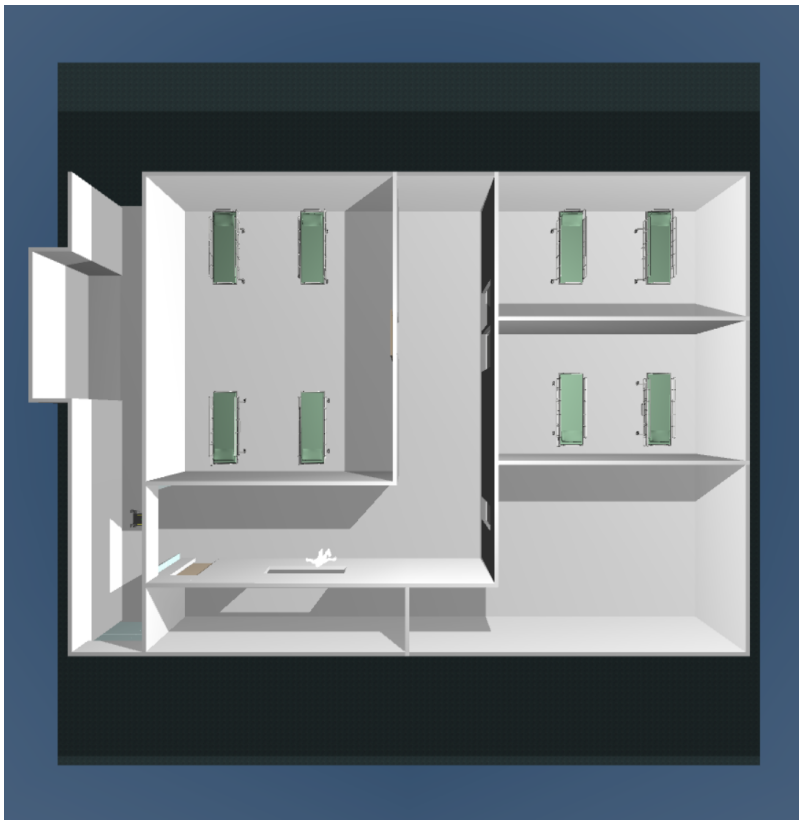


Figure 4.31: Real-scale inpatient unit simulated in SimTwo.

The robot thus has to follow a path between (2.635, 10.430) and (0.710, 4.555). The start position of the robot was at (2.604, 10.444), with a $-90°$-angle.

Figure 4.35 shows the results obtained for the robot's position, while Figure 4.36 presents a sequence of snapshots of the robot moving in the inpatient unit along the preset path. The robot travelled along the path and reached the target position in approximately 60 seconds. At the final position, the errors in the distance to the target (e_{dist}) and in the angular position (e_θ) were, respectively, 0.005 m and -0.027 rad.

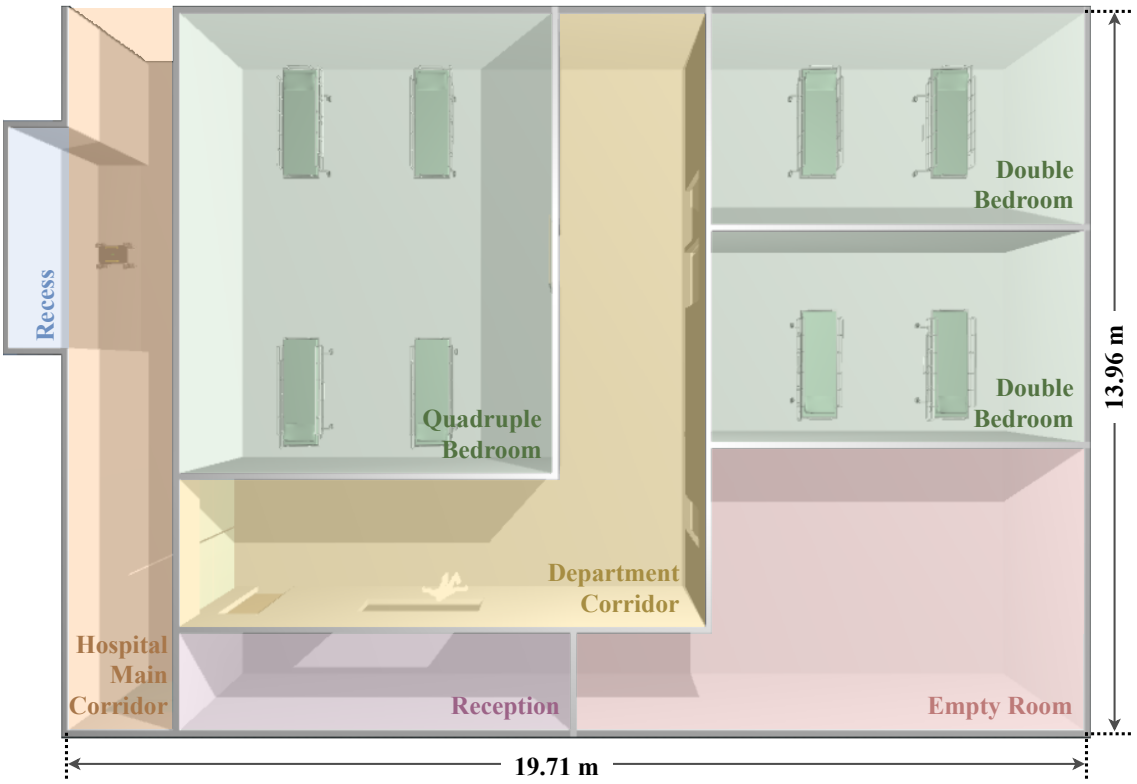


Figure 4.32: Identification of the rooms of the simulated inpatient unit.

Table 4.14: Dimensions set in the simulator to build the inpatient unit.

Ceiling (height)	2.700 m	**Walls** (thickness)	0.120 m
Recess[a]	7.130 × 1.150 m	**Reception**[b]	7.660 × 2.000 m
Hospital Main Corridor (width)	2.150 m	**Hospital Main Door** (width)	(2×) 1.110 m
Department Corridor (width)	2.960 m	**Department Door** (width)	(2×) 1.540 m
Double Bedrooms (size)	7.300 × 4.200 m	**Quadruple Bedroom**[c] (size)	7.300 × 9.000 m

[a] As the main corridor does not allow simultaneous passing of two patient beds/trolleys side by side (contrary to the department corridor), the recess can be used to pull one of them out and wait for the other to pass.
[b] Reception of the inpatient unit.
[c] In [62], it is indicated that the maximum bedroom capacity shall be four patients.

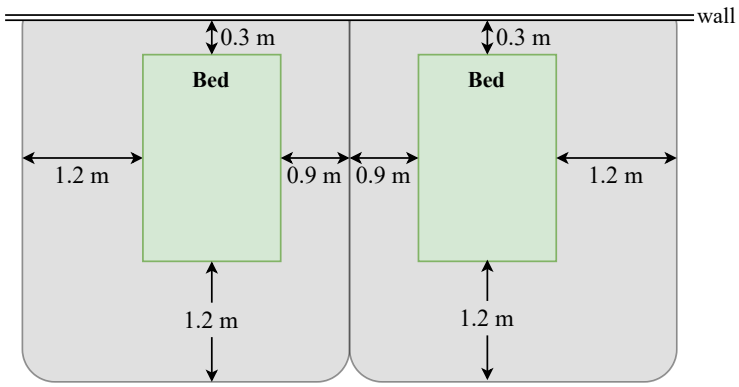


Figure 4.33: Diagram of the clearance (coloured in gray) required around each bed (the values indicated correspond to minimum requirements).

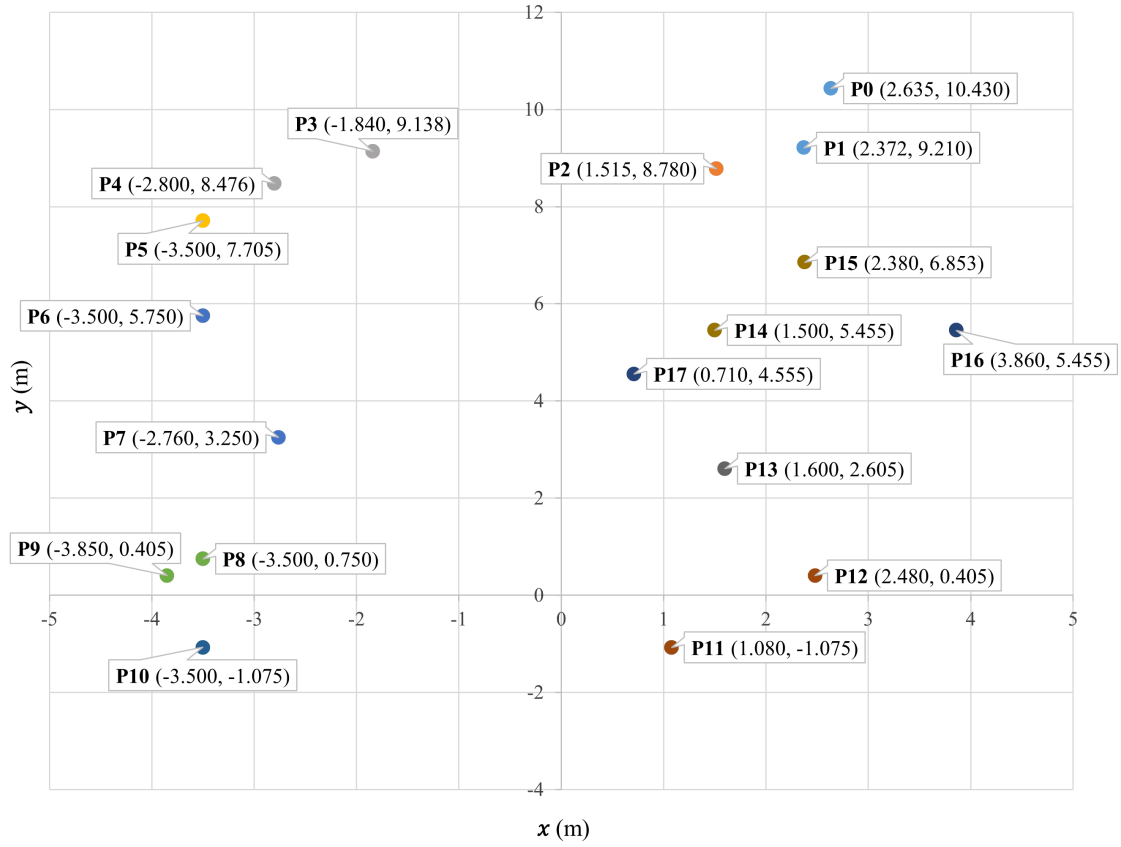


Figure 4.34: Points that define the path followed by the robot in the simulated department.

Table 4.15: Sequence of trajectories and respective points that characterise the path followed by the robot in the simulated department.

Type	**Start point** (m)	**End point** (m)	**Intermediate point** (m)
Curve	(2.635, 10.430)	(1.515, 8.780)	(2.372, 9.210)
Line	(1.515, 8.780)	(−1.840, 9.138)	—
Curve	(−1.840, 9.138)	(−3.500, 7.705)	(−2.800, 8.476)
Line	(−3.500, 7.705)	(−3.500, 5.750)	—
Curve	(−3.500, 5.750)	(−3.500, 0.750)	(−2.760, 3.259)
Curve	(−3.500, 0.750)	(−3.500, −1.075)	(−3.850, 0.405)
Line	(−3.500, −1.075)	(1.080, −1.075)	—
Curve	(1.080, −1.075)	(1.600, 2.605)	(2.480, 0.405)
Line	(1.600, 2.605)	(1.500, 5.455)	—
Curve	(1.500, 5.455)	(3.860, 5.455)	(2.380, 6.853)
Line	(3.860, 5.455)	(0.710, 4.555)	—

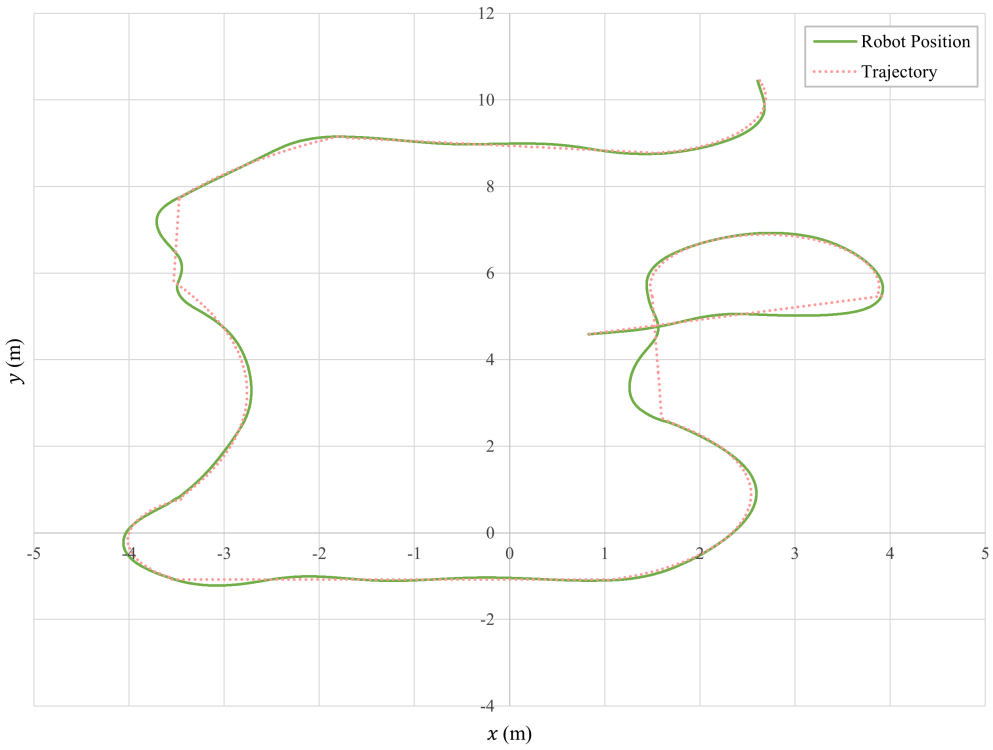
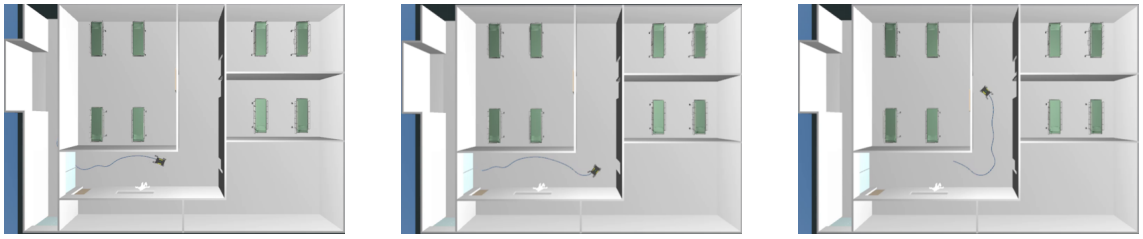
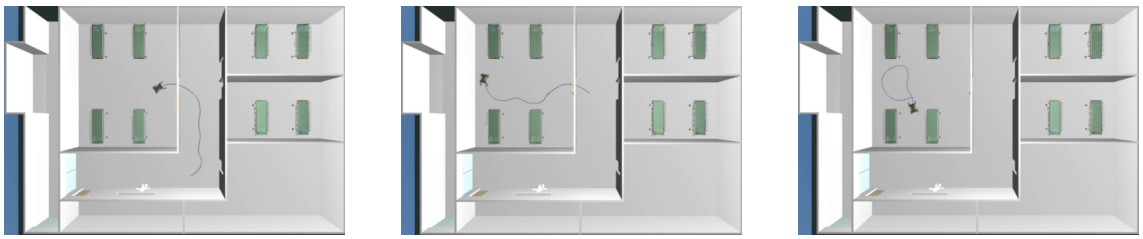


Figure 4.35: Chart of the robot position in the simulated inpatient unit, applying the developed trajectory tracking algorithm with a path.

(a) $t = 0$ s **(b)** $t = 10.00$ s **(c)** $t = 16.36$ s



(d) $t = 23.14$ s **(e)** $t = 28.26$ s **(f)** $t = 36.94$ s



(g) $t = 41.02$ s **(h)** $t = 50.84$ s **(i)** $t = 59.82$ s

Figure 4.36: Sequence of snapshots (at timestamp t) of the robot moving along the path (defined in Table 4.15) in the simulated inpatient unit.

Chapter 5

Conclusions and Future Work

The following chapter concludes the Dissertation work.

5.1 Conclusions

This Dissertation has contributed to the development of a hybrid legged-wheeled vehicle by establishing its simulation model.

The legged-wheeled robot has four 3-DOF legs with wheels attached at foot-end. Each leg combines rigid and non-rigid rotational joints (these non-rigid joints are purely passive).

Modelling the robot is helpful for refinement and testing purposes. Simulation can also ensure accurate operation in real world environments, which is particularly important in medical settings. Healthcare facilities are unstructured and highly dynamic environments, presenting varying ground conditions and obstacles. Therefore, integrating a robot with hybrid locomotion and its simulation serves the purpose of this specific application.

The simulation of the robot was developed in the SimTwo simulator. Its feasibility to realistically represent the vehicle was analysed and confirmed due to the high similarity observed between the simulated and real data.

Regarding control of the model, low-level control was implemented within SimTwo, while an application communicating with the simulator via UDP packets was responsible for the execution of the high-level control.

A trajectory tracking algorithm has been developed and implemented. From the results obtained, it can be determined that it is a strategy that can be used in the real robot. The algorithm was further tested in an environment built in the simulator that reproduces a real-scale inpatient unit.

This Dissertation has included the preparation of two papers.

The first, titled “Realistic 3D Simulation of a Hybrid Legged-Wheeled Robot”, has been accepted for presentation at the CLAWAR 2021 conference (24^{th} issue of the International Conference Series on Climbing and Walking Robots and the Support Technologies for Mobile Machines). The paper elaborates on the simulation model of the robot, describing its development and validation in SimTwo.

The second has been published:

> Vítor H. Pinto, Inês N. Soares, Marco Rocha, José Lima, José Gonçalves, and Paulo Costa. "Design, Modeling, and Control of an Autonomous Legged-Wheeled Hybrid Robotic Vehicle with Non-Rigid Joints". In: *Applied Sciences* 11.13 (2021). DOI: 10.3390/app11136116.

This article regards the design, modelling and control of the vehicle in which the Dissertation has been based, as well as the design of its simulation model and description of the implemented motion control algorithms.

5.2 Future Work

In the future, as the simulation model has been validated and proved to be realistic, further work can continue in order to include control of the robot in the z-axis and to develop and test algorithms for trajectory control and planning.

Motion control with more complex algorithms, namely for obstacle avoidance, will require including sensors (such as LiDAR sensors) in the robot or in its surrounding environment to allow the robot to perceive its workspace.

The model can also be used to develop and tune the gait planning framework of the robot.

Furthermore, proceeding with the replication of different real world environments in the simulator and assessing the robot performance within them can confirm the capability of a legged-wheeled system to outperform platforms with traditional locomotion mechanisms.

Bibliography

[1] Department of Economic and Social Affairs, Population Division. *World Population Prospects 2019: Highlights*. ST/ESA/SER.A/423. United Nations, 2019.

[2] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. "Introduction". In: *Robotics: Modelling, Planning and Control*. London: Springer London, 2009, pp. 1–37. DOI: 10.1007/978-1-84628-642-1_1.

[3] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. "A review of mobile robots: Concepts, methods, theoretical framework, and applications". In: *International Journal of Advanced Robotic Systems* 16.2 (2019). DOI: 10.1177/1729881419839596.

[4] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[5] Kathrin Cresswell and Aziz Sheikh. "Can Disinfection Robots Reduce the Risk of Transmission of SARS-CoV-2 in Health Care and Educational Settings?" In: *Journal of Medical Internet Research* 22.9 (2020), e20896. DOI: 10.2196/20896.

[6] Nikos Katevas. *Mobile robotics in healthcare*. Vol. 7. IOS Press, 2001.

[7] Marko Bjelonic, Prajish K. Sankar, C. Dario Bellicoso, Heike Vallery, and Marco Hutter. "Rolling in the Deep–Hybrid Locomotion for Wheeled-Legged Robots using Online Trajectory Optimization". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3626–3633. DOI: 10.1109/LRA.2020.2979661.

[8] N. Eiji and N. Sei. "Leg-Wheel Robot: A Futuristic Mobile Platform for Forestry Industry". In: *Proceedings of 1993 IEEE/Tsukuba International Workshop on Advanced Robotics*. IEEE. 1993, pp. 109–112. DOI: 10.1109/ICAR.1993.337208.

[9] Yu-Jie Dai, Eiji Nakano, Takayuki Takahashi, and Hiroki Ookubo. "Motion Control of Leg-Wheel Robot for an Unexplored Outdoor Environment". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*. Vol. 2. IEEE. 1996, pp. 402–409. DOI: 10.1109/IROS.1996.570803.

[10] Shuro Nakajima, Eiji Nakano, and Takayuki Takahashi. "Motion Control Technique for Practical Use of a Leg-Wheel Robot on Unknown Outdoor Rough Terrains". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 2. IEEE. 2004, pp. 1353–1358. DOI: 10.1109/IROS.2004.1389584.

[11] Michele Lacagnina, Giovanni Muscato, and Rosario Sinatra. “Kinematics, dynamics and control of a hybrid robot Wheeleg”. In: *Robotics and Autonomous Systems* 45.3-4 (2003), pp. 161–180. DOI: https://doi.org/10.1016/j.robot.2003.09.006.

[12] Erika Ottaviano and Pierluigi Rea. “Design and operation of a 2-DOF leg–wheel hybrid robot”. In: *Robotica* 31.8 (2013), pp. 1319–1325. DOI: 10.1017/S0263574713000556.

[13] Yili Fu, Zhihai Li, and Shuguo Wang. “A Wheel-leg Hybrid Wall Climbing Robot with Multi-surface Locomotion Ability”. In: *2008 IEEE International Conference on Mechatronics and Automation*. IEEE. 2008, pp. 627–632. DOI: 10.1109/ICMA.2008.4798829.

[14] Dongping Lu, Erbao Dong, Chunshan Liu, Min Xu, and Jie Yang. “Design and Development of a Leg-Wheel Hybrid Robot “HyTRo-I””. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 6031–6036. DOI: 10.1109/IROS.2013.6697232.

[15] Luca Bruzzone and Pietro Fanghella. “Mantis: hybrid leg-wheel ground mobile robot”. In: *Industrial Robot: An International Journal* (2014). DOI: 10.1108/IR-02-2013-330.

[16] Aarne Halme, Ilkka Leppänen, Jussi Suomela, Sami Ylönen, and Ilkka Kettunen. “WorkPartner: Interactive Human-Like Service Robot for Outdoor Applications”. In: *The International Journal of Robotics Research* 22.7-8 (2003), pp. 627–640. DOI: 10.1177/02783649030227011.

[17] Navvab Kashiri *et al.* “CENTAURO: A hybrid locomotion and high power resilient manipulation platform”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1595–1602. DOI: 10.1109/LRA.2019.2896758.

[18] Max Schwarz *et al.* “NimbRo Rescue: Solving disaster-response tasks with the mobile manipulation robot Momaro”. In: *Journal of Field Robotics* 34.2 (2017), pp. 400–425. DOI: https://doi.org/10.1002/rob.21677.

[19] G. Besseron, Ch. Grand, F. Ben Amar, F. Plumet, and Ph. Bidaud. “Locomotion Modes of an Hybrid Wheel-Legged Robot”. In: *Climbing and Walking Robots*. Springer, 2005, pp. 825–833.

[20] Takahiro Tanaka and Shigeo Hirose. “Development of Leg-wheel Hybrid Quadruped "AirHopper” Design of Powerful Light-weight Leg with Wheel”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 3890–3895. DOI: 10.1109/IROS.2008.4650880.

[21] Hui Peng, Junzheng Wang, Wei Shen, and Dawei Shi. “Cooperative attitude control for a wheel-legged robot”. In: *Peer-to-Peer Networking and Applications* 12.6 (2019), pp. 1741–1752. DOI: 10.1007/s12083-019-00747-x.

[22] Zhihua Chen *et al.* “Control strategy of stable walking for a hexapod wheel-legged robot”. In: *ISA transactions* (2020). DOI: https://doi.org/10.1016/j.isatra.2020.08.033.

[23] Gustavo Freitas, Fernando Lizarralde, Liu Hsu, Vitor Paranhos, Ney Reis, Dos Reis, and Marcel Bergerman. "Design, Modeling, and Control of a Wheel-Legged Locomotion System for the Environmental Hybrid Robot". In: *Proceedings of the IASTED International Conference, Pittsburgh, Penn*. 2011, pp. 7–9. DOI: 10.2316/P.2011.752-011.

[24] Maoxun Li, Shuxiang Guo, Hideyuki Hirata, and Hidenori Ishihara. "A roller-skating/walking mode-based amphibious robot". In: *Robotics and Computer-Integrated Manufacturing* 44 (2017), pp. 17–29. DOI: https://doi.org/10.1016/j.rcim.2016.06.005.

[25] Nam-Su Yuk and Dong-Soo Kwon. "Realization of Expressive Body Motion Using Leg-Wheel Hybrid Mobile Robot: KaMERo". In: *2008 International Conference on Control, Automation and Systems*. IEEE. 2008, pp. 2350–2355. DOI: 10.1109/ICCAS.2008.4694198.

[26] Qing Shi *et al.* "Development of the Hybrid Wheel-legged Mobile Robot WR-3 Designed to Interact with Rats". In: *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*. IEEE. 2010, pp. 887–892. DOI: 10.1109/BIOROB.2010.5627719.

[27] Alexander S. Boxerbaum, Julio Oro, Gilbert Peterson, and Roger D. Quinn. "The latest generation Whegs™ robot features a passive-compliant body joint". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 1636–1641. DOI: 10.1109/IROS.2008.4650997.

[28] Markus Eich, Felix Grimminger, Stefan Bosse, Dirk Spenneberg, and Frank Kirchner. "Asguard: A hybrid-wheel security and SAR-robot using bio-inspired locomotion for rough terrain". In: (2007).

[29] Tao Sun, Xu Xiang, Weihua Su, Hang Wu, and Yimin Song. "A transformable wheel-legged mobile robot: design, analysis and experiment". In: *Robotics and Autonomous Systems* 98 (2017), pp. 30–41. DOI: https://doi.org/10.1016/j.robot.2017.09.008.

[30] Aarne Halme, Ilkka Leppänen, Miso Montonen, and Sami Ylönen. "Robot motion by simultaneously wheel and leg propulsion". In: *Proc. CLAWAR*. 2001, pp. 1013–1020.

[31] Shen-Chiang Chen, Ke-Jung Huang, Wei-Hsi Chen, Shuan-Yu Shen, Cheng-Hsin Li, and Pei-Chun Lin. "Quattroped: a leg–wheel transformable robot". In: *IEEE/ASME Transactions On Mechatronics* 19.2 (2014), pp. 730–742. DOI: 10.1109/TMECH.2013.2253615. Available online: https://ieeexplore.ieee.org/abstract/document/6508894.

[32] Wei-Hsi Chen, Hung-Sheng Lin, Yun-Meng Lin, and Pei-Chun Lin. "TurboQuad: A novel leg–wheel transformable robot with smooth and fast behavioral transitions". In: *IEEE Transactions on robotics* 33.5 (2017), pp. 1025–1040. DOI: 10.1109/TRO.2017.2696022.

[33] Long Bai, Jian Guan, Xiaohong Chen, Junzhan Hou, and Wenbo Duan. "An optional passive/active transformable wheel-legged mobility concept for search and rescue robots". In: *Robotics and Autonomous Systems* 107 (2018), pp. 145–155. DOI: https://doi.org/10.1016/j.robot.2018.06.005.

[34] Caio Camargo, José Gonçalves, Miguel Á. Conde, Francisco J. Rodríguez-Sedano, Paulo Costa, and Francisco J. García-Peñalvo. "Systematic Literature Review of Realistic Simulators Applied in Educational Robotics Context". In: *Sensors* 21.12 (2021). DOI: 10.3390/s21124031.

[35] Zandra B. Rivera, Marco C. De Simone, and Domenico Guida. "Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments". In: *Machines* 7.2 (2019). DOI: 10.3390/machines7020042.

[36] Olivier Michel. "Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation". In: *International Journal of Advanced Robotic Systems* 1.1 (2004), p. 5. DOI: 10.5772/5618.

[37] Costa Paulo, Gonçalves José, Lima José, and Malheiros Paulo. "SimTwo Realistic Simulator: A Tool for the Development and Validation of Robot Software". In: *Theory and Applications of Mathematics & Computer Science* 1.1 (Apr. 2011), Pages: 17–33.

[38] José Gonçalves, José Lima, Hélder Oliveira, and Paulo Costa. "Sensor and actuator modeling of a realistic wheeled mobile robot simulator". In: *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. 2008, pp. 980–985. DOI: 10.1109/ETFA.2008.4638513.

[39] J. Gonçalves, J. Lima, P. Malheiros, and P. Costa. "Fostering Advances in Mechatronics and Robotics Resorting to Simulation". In: *IFAC Proceedings Volumes* 43.4 (2010). 10th IFAC Workshop on Intelligent Manufacturing Systems, pp. 326–331. DOI: https://doi.org/10.3182/20100701-2-PT-4011.00056.

[40] José Gonçalves, José Lima, Paulo Malheiros, and Paulo Costa. "Realistic simulation of a Lego Mindstorms NXT based robot". In: *2009 IEEE Control Applications, (CCA) Intelligent Control, (ISIC)*. 2009, pp. 1242–1247. DOI: 10.1109/CCA.2009.5280986.

[41] José Gonçalves, José Lima, Paulo J. Costa, and A. Paulo Moreira. "Modeling and Simulation of the EMG30 Geared Motor with Encoder Resorting to SimTwo: The Official Robot@Factory Simulator". In: *Advances in Sustainable and Competitive Manufacturing Systems*. Ed. by Américo Azevedo. Heidelberg: Springer International Publishing, 2013, pp. 307–314. DOI: https://doi.org/10.1007/978-3-319-00557-7_25.

[42] José L. Lima, José C. Gonçalves, Paulo G. Costa, and A. Paulo Moreira. "Humanoid Robot Simulation with a Joint Trajectory Optimized Controller". In: *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. 2008, pp. 986–993. DOI: 10.1109/ETFA.2008.4638514.

[43] José L. Lima, José A. Gonçalves, Paulo G. Costa, and A. Paulo Moreira. "Humanoid Gait Optimization Resorting to an Improved Simulation Model". In: *International Journal of Advanced Robotic Systems* 10.1 (2013), p. 67. DOI: 10.5772/54766. eprint: https://doi.org/10.5772/54766.

[44] Daniel Campos, Joana Santos, José Gonçalves, and Paulo Costa. "Modeling and Simulation of a Hacked Neato XV-11 Laser Scanner". In: *Robot 2015: Second Iberian Robotics Conference*. Ed. by Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-Martinez. Cham: Springer International Publishing, 2016, pp. 425–436. DOI: https://doi.org/10.1007/978-3-319-27146-0_33.

[45] Tatiana Pinho, António Paulo Moreira, and José Boaventura-Cunha. "Framework Using ROS and SimTwo Simulator for Realistic Test of Mobile Robot Controllers". In: *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*. Ed. by António Paulo Moreira, Aníbal Matos, and Germano Veiga. Cham: Springer International Publishing, 2015, pp. 751–759. DOI: https://doi.org/10.1007/978-3-319-10380-8_72.

[46] Paulo José Costa, Nuno Moreira, Daniel Campos, José Gonçalves, José Lima, and Pedro Luís Costa. "Localization and Navigation of an Omnidirectional Mobile Robot: The Robot@Factory Case Study". In: *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje* 11.1 (2016), pp. 1–9. DOI: 10.1109/RITA.2016.2518420.

[47] Lucas Eckert, Luis Piardi, José Lima, Paulo Costa, Antonio Valente, and Alberto Nakano. "3D Simulator Based on SimTwo to Evaluate Algorithms in Micromouse Competition". In: *New Knowledge in Information Systems and Technologies*. Ed. by Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo. Cham: Springer International Publishing, 2019, pp. 896–903. DOI: https://doi.org/10.1007/978-3-030-16181-1_84.

[48] Malgorzata Kamedula, Navvab Kashiri, and Nikos G. Tsagarakis. "On the Kinematics of Wheeled Motion Control of a Hybrid Wheeled-Legged CENTAURO robot". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2426–2433.

[49] Marco Hutter *et al.* "ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 38–44. DOI: 10.1109/IROS.2016.7758092.

[50] C. Dario Bellicoso, Fabian Jenelten, Christian Gehring, and Marco Hutter. "Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2261–2268. DOI: 10.1109/LRA.2018.2794620.

[51] Mathieu Geisert, Thomas Yates, Asil Orgen, Pierre Fernbach, and Ioannis Havoutis. "Contact Planning for the ANYmal Quadruped Robot Using an Acyclic Reachability-Based Planner". In: *Towards Autonomous Robotic Systems*. Ed. by Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang. Cham: Springer International Publishing, 2019, pp. 275–287.

[52] Jemin Hwangbo, Joonho Lee, and Marco Hutter. "Per-Contact Iteration Method for Solving Contact Dynamics". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 895–902. DOI: 10.1109/LRA.2018.2792536.

[53] Akihiro Suzumura and Yasutaka Fujimoto. "Real-Time Motion Generation and Control Systems for High Wheel-Legged Robot Mobility". In: *IEEE Transactions on Industrial Electronics* 61.7 (2013), pp. 3648–3659. DOI: 10.1109/TIE.2013.2286071.

[54] Vítor H. Pinto, José Gonçalves, and Paulo Costa. "Design, Modeling, and Control of a Single Leg for a Legged-Wheeled Locomotion System with Non-Rigid Joint". In: *Actuators* 10.2 (2021). DOI: 10.3390/act10020029.

[55] Vítor H. Pinto, Inês N. Soares, Marco Rocha, José Lima, José Gonçalves, and Paulo Costa. "Design, Modeling, and Control of an Autonomous Legged-Wheeled Hybrid Robotic Vehicle with Non-Rigid Joints". In: *Applied Sciences* 11.13 (2021). DOI: 10.3390/app11136116.

[56] Vítor H. Pinto, José Gonçalves, and Paulo Costa. "Towards a More Robust Non-Rigid Robotic Joint". In: *Applied System Innovation* 3.4 (2020). DOI: 10.3390/asi3040045.

[57] Vítor H. Pinto, José Gonçalves, and Paulo Costa. "Modeling and Control of a DC Motor Coupled to a Non-Rigid Joint". In: *Applied System Innovation* 3.2 (2020). DOI: 10.3390/asi3020024.

[58] Antonio Visioli. "Basics of PID Control". In: *Practical PID control*. Springer-Verlag London, 2006, pp. 1–18. DOI: 10.1007/1-84628-586-0_1.

[59] M. A. Johnson. "PID Control Technology". In: *PID Control: New Identification and Design Methods*. Ed. by Michael A. Johnson and Mohammad H. Moradi. Springer-Verlag London, 2005, pp. 1–46. DOI: 10.1007/1-84628-148-2_1.

[60] Antonio Visioli. "Anti-windup Strategies". In: *Practical PID control*. Springer-Verlag London, 2006, pp. 35–60. DOI: 10.1007/1-84628-586-0_3.

[61] Richard J. Vaccaro. *Digital Control: A State-Space Approach*. McGraw-Hill Higher Education, 1995.

[62] International Health Facility Guidelines. *Part B – Health Facility Briefing & Design, including Functional Planning Units*. International Health Facility Guidelines. 2017. Available online: https://healthfacilityguidelines.com/ViewPDF/ViewIndexPDF/iHFG_part_b_complete. Accessed on: June 15, 2021.

[63] Department of Health. *Health Building Note 00-04: Circulation and communication spaces. Core elements*. Department of Health. 2013. Available online: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/187026/Health_Building_Note_00-04_-_Circulation_and_communication_spaces_-_updated_April_2013.pdf. Accessed on: June 15, 2021.