

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Development of a Quadruped Robot with modular legs

Gustavo Barbosa Cohen

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Paulo Gomes da Costa

Second Supervisor: José Luis Lima

July 22, 2024

Abstract

In the exciting field of robotics and mechatronics, researchers and innovators have constantly shown interest in developing ways to improve the mobility of robots in challenging environments. This focus on enhanced mobility stems from the limitations that many traditional robots face when navigating complex terrains. As a result, there has been a surge of interest in exploring different modes of ground locomotion, with a particular emphasis on the mechanisms of legged movement.

This dissertation addresses the development of a mammal-type quadruped robot with modular legs. To achieve this, each leg is designed with three degrees of freedom, incorporating servo feedback for precise control. Given the torque intensive nature of mammal-type quadruped robot gaits, a key component of this work is the study and comparison of optimization techniques. Initially, single-objective optimization methods were tested, however, recognizing the complexity of balancing torque reduction and maximizing travel distance, the study progressed to evaluate multi-objective genetic algorithms in a simulated environment. An in-depth comparison and study was then conducted to determine which of the algorithms had the best results and, subsequently, the resulting optimized gait was applied to the real robot.

Finally, in addition to the development of the robot and optimization studies, several tests were conducted to validate the robots joint accuracy, torque, and distance traveled when performing the optimized gait. The results from these tests were then recorded and compared with those from the simulator. The findings were satisfactory, showing relatively small discrepancies between the real and simulated robots.

Resumo

No domínio da robótica e da mecatrónica, investigadores e inovadores têm demonstrado interesse em desenvolver métodos para melhorar a mobilidade de robôs em ambientes difíceis. Este foco na mobilidade resulta das limitações que muitos robôs tradicionais enfrentam quando navegam em terrenos complexos. Como resultado, tem havido uma onda de interesse em explorar diferentes modos de locomoção no solo, com particular ênfase nos mecanismos de movimento com pernas.

Esta dissertação aborda o desenvolvimento de um robô quadrúpede do tipo mamífero com pernas modulares. Para tal, cada pata é concebida com três graus de liberdade, incorporando servo feedback para um controlo com maior precisão. Dada a natureza intensiva de binário em gaits de robôs quadrúpedes mamíferos, um componente fundamental deste trabalho é o estudo e comparação de técnicas de otimização. Inicialmente, foram testados métodos de otimização com um único objetivo. No entanto, reconhecendo a complexidade do equilíbrio entre a redução do binário e a maximização da distância percorrida, o estudo avançou para a avaliação de algoritmos genéticos multi-objetivo em ambiente simulado. Foi efectuada uma comparação e um estudo aprofundados para determinar qual dos algoritmos apresentava os melhores resultados e, subsequentemente, a marcha optimizada resultante foi aplicada ao robô real.

Finalmente, para além do desenvolvimento do robô e dos estudos de otimização, foram realizados vários testes para validar a precisão das articulações, o binário e a distância percorrida pelo robô ao executar a marcha optimizada. Os resultados destes testes foram depois registados e comparados com os do simulador. Os resultados foram satisfatórios, mostrando discrepâncias relativamente pequenas entre o robô real e o simulado.

Sustainable Development Goals (SDGs) of the United Nations

In the context of sustainable development, the following Table 1 highlights the specific Sustainable Development Goals that this dissertation aligns with, detailing its contributions and potential impact on achieving these global objectives.

Table 1: Contributions of PLA and Legged Robots to Sustainable Development Goals

SDG	Goal	Contribution	Key Performance Indicator
7 - Affordable and Clean Energy	3 - By 2030, substantially increase the share of renewable energy in the global energy matrix.	Optimized robot movements and efficient servo operation reduce energy consumption, promoting the use of clean and affordable energy sources. This contributes to energy sustainability in industrial applications.	Energy intensity of the economy in primary energy
11 - Sustainable Cities and Communities	3 - By 2030, increase inclusive and sustainable urbanization and capacities for participatory, integrated and sustainable human settlement planning in all countries	In urban settings, legged robots can navigate infrastructure and public spaces without causing damage to pavements, parks, and other communal areas, supporting the development of sustainable and resilient cities.	Proportion of cities with a structure for the direct participation of civil society in urban planning and management that operates regularly and democratically
12 - Responsible Consumption and Production	2 - By 2030, achieve the sustainable management and efficient use of natural resources	PLA is derived from renewable resources, such as corn starch or sugarcane, which reduces the reliance on finite fossil fuels.	Domestic material consumption
13 - Climate Action	2 - Integrate climate change measures into national policies, strategies and planning	PLA production results in lower greenhouse gas emissions compared to traditional petroleum-based plastics.	Countries that adopt and implement national disaster risk reduction strategies in line with the Sendai Framework for Disaster Risk Reduction 2015-2030

SDG	Goal	Contribution	Key Performance Indicator
15 - Life on Land	1 - By 2020, ensure the conservation, restoration and sustainable use of terrestrial and inland freshwater ecosystems and their services, in particular forests, wetlands, mountains and drylands, in line with obligations under international agreements.	Unlike wheeled or tracked robots that can compact soil and damage vegetation, legged robots distribute their weight more evenly and delicately, causing minimal disturbance to the ground and surrounding flora	Proportion of important sites for terrestrial and freshwater biodiversity that are covered by protected areas, by ecosystem type

Acknowledgments

I would like to express my sincere gratitude to the following individuals for their incredible support and contributions during the development of this dissertation.

First and foremost, I am immensely grateful to my supervisors, Professor Paulo Costa and Professor José Lima, for their guidance, encouragement, and the laughs we shared during post-dinner meetings. Their support and knowledge was crucial to the development of this work as well as to my growth as an academic and professional.

Additionally, I would like to thank the friends and colleagues I made throughout my academic journey. A special thank you to André Barata, Diogo Vilela, and José Mendes for the moments we spent together. I will truly cherish these memories for the rest of my life.

I would also like to extend a heartfelt thank you to my girlfriend, Alice, for her patience, understanding, and unwavering encouragement during this stressful time. Your love and belief in me have been a constant source of strength and support. Thank you for always being there for me.

With all my heart I could never truly put into words how grateful I am to my family for their love, and support throughout my whole life. To my parents, Paula and Ilídio, and sister, Rebeca, thank you. You are the driving force behind all my achievements and my refuge in times of failure. And to my grandparents, living and departed, this work is a testament to your enduring love and support. I am profoundly grateful for everything you have done for me.

Gustavo Cohen.

*“I’m an optimistic guy.
One of those big dreamers.
I’m one of those kids with an annoying imagination.”*

Tyler Gregory Okonma

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Objectives	2
1.3	Document Structure	2
2	Background and State-of-the-art	5
2.1	Legged Robots	5
2.1.1	Humanoid/Bipedal Robots	5
2.1.2	Hexapod/Hexopedal Robots	6
2.1.3	Octapod Robots	7
2.1.4	Quadruped robots	8
2.2	Degrees of Freedom	12
2.3	Servo Motor Control Methods	13
2.3.1	UART	14
2.3.2	PWM	14
2.3.3	Comparison	15
2.4	Transmission Control Protocol	16
2.4.1	Header Format	17
2.5	Conclusion	17
3	Quadruped Robot	19
3.1	Initial Model	19
3.2	3D Model	20
3.2.1	Link and Joint Modifications	20
3.2.2	Enhancements and Further Modifications for Improved Performance . . .	22
3.3	Waveshare ST3215 Servo Comparisons	23
3.3.1	Comparison to other servos	23
3.4	Kinematics	24
3.4.1	Inverse Kinematics	24
3.4.2	Forward Kinematics	27
4	Simulation	31
4.1	Overview	31
4.2	Simulation Loop	32
4.3	Torque Study	34
4.4	Torque Peak Optimization	35
4.4.1	Leg Model Configuration	36
4.5	Further SHGO Optimization	37

4.6	Multi-Objective Optimization	39
4.6.1	NSGA-II	40
4.6.2	NSGA-III	40
4.6.3	U-NSGA-III	41
4.6.4	Problem Formulation and Evaluation System	42
4.6.5	NSGA Setup	42
4.6.6	Hypervolume Setup	44
4.6.7	NSGA Optimization Results	44
4.7	Final Optimization using NSGA-III	49
5	Robot Prototype	51
5.1	Gait Generation	51
5.2	Communication	52
5.2.1	Data Format	52
5.3	Microcontroller Software Structure	55
5.3.1	SpotServo	56
5.3.2	Spot	56
5.4	Gait Execution	56
5.5	Graphic User Interface (GUI)	57
6	Tests and Results	59
6.1	Gait Accuracy	59
6.2	Leg Torque	63
6.2.1	Servo Load Feedback Calibration	63
6.2.2	Null Acceleration Test	63
6.2.3	Acceleration Test	64
6.2.4	Resulting Correction	66
6.2.5	Leg Torque Gait Results	67
6.3	Distance Traveled	68
7	Conclusion and Future Work	69
	References	73

List of Figures

2.1	First generation ASIMO robot [1].	6
2.2	LAURON V in the field [2].	6
2.3	SILO-6 [3].	7
2.4	T8X Robot Spider [4].	7
2.5	Mammal-type quadruped robot [5].	8
2.6	Sprawling-type quadruped robot [5].	8
2.7	Boston Dynamics' Spot [6].	9
2.8	Boston Dynamics' Spot equipped with a manipulator arm [6].	9
2.9	MIT's Mini Cheetah [7].	9
2.10	Tokyo Tech - Suzumori Endo Lab's TITAN-XIII [5].	10
2.11	Creep gait step-by-step [8].	10
2.12	Horse in a trot gait [9].	11
2.13	Horse in a pace gait [9].	11
2.14	Horse in a gallop gait [9].	11
2.15	Design drawing of a sprawling-type quadruped leg with three degrees of freedom.	13
2.16	Design drawing of a mammal-type quadruped leg with three degrees of freedom.	13
2.17	UART frame format [10].	14
2.18	Pulse-width modulation and resulting average voltage [11].	15
2.19	TCP Sequence diagram	16
2.20	TCP Header Frame	17
3.1	DOGGO 3D Model by Jakub Plywaczewski [12].	19
3.2	Shoulder joint modifications.	21
3.3	Leg modifications.	21
3.4	Full robot modifications	22
3.5	Full robot modifications	22
3.6	Representation of mammal-type quadruped robot.	24
3.7	Front perspective of the mammal leg.	25
3.8	Rotation of referential.	26
3.9	Side perspective of mammal leg.	26
3.10	Front perspective of the mammal leg	27
3.11	Side perspective for direct kinematics.	28
4.1	Simulated quadruped robot sitting.	32
4.2	Resulting joint angles comparison for the same end-effector position.	32
4.3	Simulation loop UML sequence diagram.	33
4.4	Rear Right leg torque evolution of a non-optimized model.	34
4.5	Rear Right leg torque evolution of an optimized model.	35

4.6	Rear Right leg torque evolution of a SHGO optimized model with shortened links.	37
4.7	Distance traveled comparison between optimization algorithms.	38
4.8	Torque evolution of the SHGO distance optimized gait	39
4.9	Combined NSGA-II Pareto fronts.	45
4.10	Combined U-NSGA-III Pareto fronts.	45
4.11	Combined NSGA-III Pareto fronts.	45
4.12	Average hypervolume evolution for NSGA-II.	46
4.13	Average hypervolume evolution for NSGA-III.	46
4.14	Average hypervolume evolution for U-NSGA-III	46
4.15	Average hypervolume evolution for the three algorithms.	46
4.16	Median function values for the null overdrive torque solutions for each algorithm - Averaged over 8 runs.	48
4.17	Front Left Leg.	49
4.18	Front Right Leg.	49
4.19	Rear Left Leg	49
4.20	Rear Right Leg.	49
4.21	Leg torque evolution for the NSGA-III optimal gait solution.	49
5.1	Gait Generation activity diagram.	52
5.2	Forward gait load message frame example for a Front Left leg movement.	54
5.3	Forward gait loading communication sequence.	54
5.4	ESP32 Class Diagram	55
5.5	Gait Execution loop diagram.	57
5.6	Graphic User Interface (GUI).	57
6.1	Front Left leg joint angles comparison for the simulated and real robots.	59
6.2	Real and Simulated Front Left Shoulder joint angle results.	60
6.3	Shoulder joint percentage error.	60
6.4	Real and Simulated Front Left Elbow joint angle results	61
6.5	Elbow joint real percentage error.	61
6.6	Real and Simulated Front Left Wrist joint angle results	62
6.7	Wrist joint real percentage error.	62
6.8	Constant Speed Torque test for the Front Left Elbow servo.	64
6.9	Illustration of the dynamic test setup.	65
6.10	Illustration of the dynamic test setup.	65
6.11	ε Calculated values for the Front Left Elbow servo.	66
6.12	Angular velocity for formula example.	66
6.13	Uncorrected Feedback Torque.	67
6.14	Corrected Feedback Torque.	67

List of Tables

1	Contributions of PLA and Legged Robots to Sustainable Development Goals . . .	v
3.1	ST3215 and MG996R servo dimensions and mounting type.	20
3.2	Comparison between the SG90, MG966R, and ST3215 servos.	23
4.1	Max and min torque peaks for a non-optimized model.	34
4.2	Max and min torque peaks for a SHGO optimized model.	36
4.3	New and old leg link dimensions for the simulated model.	36
4.4	Max and min torque peaks for an optimized model with shortened legs.	37
4.5	Max and min torque peaks for distance and torque peak optimization.	38
4.6	Gait generation parameters	42
4.7	Crossover and Mutation parameters for the NSGA algorithms.	43
4.8	Population and Generation ratio testing for NSGA-II.	44
4.9	Best, worst and average hypervolume final results for the NSGA algorithms. . . .	46
4.10	Best, worst and average best solutions for the NSGA algorithms.	47
4.11	Best, worst, and average median evaluation for the null torque overdrive solutions.	48
5.1	Command message list.	53
6.1	Max, min and average absolute error results.	63
6.2	Max and min torque peaks for the real and simulated robots.	67
6.3	Distance traveled over 10 seconds.	68

Abreviaturas e Símbolos

DOF	Degrees of Freedom
PWM	Pulse-Width Modulation
UART	Universal Asynchronous Receiver-Transmitter
LSB	Least Significant Bit
RXD	Receive Data
TXD	Transmit Data
URDF	Unified Robot Description Format
CSV	Comma-Separated Values
FL	Front-Left
FR	Front-Right
RL	Rear-Left
RR	Rear-Right
SHGO	Simplicial Homology Global Optimization
NSGA	Non-Dominated Sorting Genetic Algorithm
TCP	Transmission Control Protocol
DO	Data Offset
ACK	Acknowledgement
URG	Urgent
SYN	Synchronize
FIN	Finish
GTI	Gait Type Indicator
LI	Leg Indicator
GUI	Graphical User Interface

Chapter 1

Introduction

The following chapter aims to provide an introduction of the work developed in this dissertation. The chapter is divided into three sections. First, Section 1.1 provides the context and motivation behind the development of this dissertation. Following that, Section 1.2 describes the objectives for this study. Finally, the structure of the document is presented in Section 1.3.

1.1 Context and Motivation

In the constantly evolving landscape of robotics and mechatronics, the pursuit of mobility in challenging terrains has become a centerpiece of research [13] and innovation. This emphasis on enhancing the mobility of robotic systems has been fueled by the limitations that many traditional robots encounter when navigating complex environments, which has spurred investigations into different forms of ground locomotion, particularly legged locomotion.

Generally speaking, there are three types of ground locomotion in robotics [13]: wheeled, legged, and tracked. Notably, wheeled and tracked locomotion often struggle with overcoming obstacles due to their lack of verticality and stability in uneven environments, this hinders their adaptability and even calls into question their suitability for exploration in extreme terrain.

In contrast, legged robots offer unique advantages in the form of stepping over obstacles, adjusting gaits for different terrains, and maintaining stability on uneven surfaces. These advantages are mainly due to their vertical mobility and ability to step, which when combined with the stability that these robots are capable of achieving when on uneven terrains, result in an extremely flexible type of locomotion.

Quadruped (4-legged) robots can also be divided into two distinct types: mammal types (robots that mimic the dynamics of a dog or a horse), and sprawling types (robots that mimic the dynamics of spiders and crawling insects). The main focus of this study will be the development and study of a mammal-type quadruped robot with three degrees of freedom.

Having legs with three degrees of freedom, meaning three joints on each leg, allows each of them to move in three independent directions. This is a crucial feature of the planned project, being that it provides so much spatial flexibility when in comparison to the same mammal-type

quadruped robot with only two degrees of freedom. It also allows for a much larger array of more efficient and sophisticated gaits which consequently results in an even more versatile robot.

The goal of this research is to improve legged locomotion using feedback servos and gait optimization. By refining gait patterns through simulation environments, the study aims to advance the capabilities of quadruped robots.

1.2 Objectives

Based on the discussions presented previously, the main focus of this dissertation is the development of a quadruped mammal-type robot. To achieve this, it will be crucial to develop an optimized gait algorithm that ensures efficient and safe movement. Additionally, the design of the robot's body must be carefully considered to ensure maximum stability.

Another key element in enhancing the robot's capabilities is the use of receiver-transmitter servos. These servos offer advanced control and communication features, allowing for real-time adjustments. Incorporating these servos into the design will ease dynamic and adaptive movements, complementing the robot's ability to traverse challenging terrains.

1.3 Document Structure

Apart from this chapter, this document is composed of six more chapters:

- Chapter 2 introduces the background and existing literature on legged robotics, with an emphasis on quadruped robots. It also addresses the importance of degrees of freedom in legged robotics, an in-depth analysis of the UART (Universal Asynchronous Receiver-Transmitter) and PWM (Pulse-Width Modulation) servo control methods, as well as a description of TCP (Transmission Control Protocol).
- Chapter 3 presents an overview of the procedure and decision process used to develop the quadruped robot. Additionally, the kinematic model is described.
- Chapter 4 explains how the robot simulation was conducted, including the process of optimizing the gait using and comparing single and multi-objective algorithms. This work led to a Conference paper accepted for publication in the 4th Optimization, Learning Algorithms and Applications Conference (OL2A 2024) with the title: "Enhancing Quadruped Robot Performance through Gait Optimization" [14].
- Chapter 5 depicts the integration of the control algorithms, software structure and communication protocols used in the robot prototype.
- Chapter 6 presents the tests and results used to evaluate the robot's performance, and optimized gait.

- Chapter 7 describes the main conclusions derived from the development of this dissertation, and details some suggestions for future work that may enhance it.

Chapter 2

Background and State-of-the-art

This chapter includes crucial background knowledge that is relevant to the understanding of the addressed topic but also provides an overview of the current advancements in legged robotics. First, in Section 2.1, a small introduction to legged robotics is presented, followed by an analysis of multiple types of legged robots, with special emphasis on quadruped robots. Afterward, in Section 2.2 the importance of degrees of freedom (DOF) is described, and how 3 DOF can be achieved on robot legs. Additionally, it delves into servo protocol control in Section 2.3. Finally, an overview of the Transmission Control Protocol is provided in Section 2.4.

2.1 Legged Robots

Legged locomotion is a common means of propulsion that allows for the existence of walking robots. They are highly advantageous for exploration of unknown areas due to their enhanced mobility in natural terrain, this comes as a result of their ability to exploit individual footholds as opposed to wheeled or tracked locomotion methods, which require a continuous support surface [15]. This means that by varying their leg configuration in response to the terrain conditions, the feet of these robots can establish contact with the ground in an exact manner. For these reasons, legs are considered inherently adequate systems for locomotion on irregular ground [16].

Despite these advantages, legged locomotion is still not traditionally used - a consequence of the full coordination that legged locomotion requires [17] and the difficulty of coping with the many degrees of freedom required for legged robot mobility [18]. This means that although this scope of robotics has been constantly researched, it is still fairly far from becoming a widely adopted mode of locomotion [17].

Even though this control problem is very challenging, it hasn't halted the development and research of various types of legged robots.

2.1.1 Humanoid/Bipedal Robots

Humanoid robots like the ASIMO [1] (displayed in Figure 2.1) unveiled in November 2000 by Honda not only utilize bipedal (two-legged) locomotion with three degrees of freedom in each leg

(DOF) but also display human-like appearance, movement, and capabilities. This means that it is fit to operate in human-centered environments, which makes them adaptable to a wide array of settings, from homes and offices to public spaces, where they can collaborate with people.

Not only that, but these robots can run, jump, step over, and possess the ability to rotate themselves around their vertical axis in the "hip joint" [19].



Figure 2.1: First generation ASIMO robot [1].

Humanoid robots like the ASIMO [1] (Figure 2.1) require a complex collaboration of mechanical, electrical, and software engineering, computer architecture, and real-time control [17]. This means that achieving stable bipedal locomotion is extremely complex since this type of locomotion is inherently unstable, and the task of maintaining balance proves itself to be very challenging.

2.1.2 Hexapod/Hexopedal Robots

Hexapod robots like the LAURON V [2] (Figure 2.2), developed by the FZI Research Center for Information Technology, are robots with six legs. These robots typically have a symmetrical shape and additional supporting limbs when in comparison to bipedal or quadruped robots [19].



Figure 2.2: LAURON V in the field [2].

It can also be noted that the center of gravity of hexapod robots when walking is maintained close to the inactive position [19]. This is an especially noticeable difference when in comparison to bipedal robots, whose constant swaying of the center of gravity requires specialized solutions,

like in the case of ASIMO (Figure 2.1), where the development of a proprietary walk stabilization control technology [1] was required to reach a more consistently stable gait.

One of the disadvantages of hexapod robots is the increase in energy usage in comparison to robots with fewer legs [19].

Note the following hexapod robot SILO-6 [3] developed by the Department of Automatic Control at the Industrial Automation Institute, Spanish National Research Council (CSIC) in Figure 2.3.

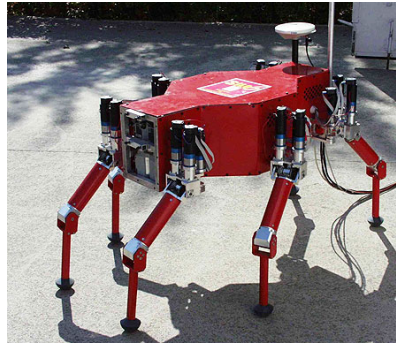


Figure 2.3: SILO-6 [3].

Intended for the detection and location of antipersonnel land mines and humanitarian missions [3], a feature of the robot displayed (in Figure 2.3) is that it has different widths of leg mount locations. Note that in Figure 2.3 [3] the robot utilizes four non-adjacent angled legs and two central support legs. As expected, the two central legs support a greater load than the corner legs and, to reduce this load, the central legs are placed further apart from each other when in comparison to the angled legs.

2.1.3 Octapod Robots

Octapod robots are similar to hexapod robots in the sense that they have almost the same advantages and disadvantages, only on a more extreme scale. Figure 2.4 shows a Robot Spider T8X developed by Robugtix [4].



Figure 2.4: T8X Robot Spider [4].

Note that these robots are very maneuverable and have the greatest stability when compared to the other types of legged robots. It is also worth noting that robots like the T8X Robot Spider

[4] (Figure 2.4) are extremely energy intensive due to the fact they employ 24 servos (when they have three degrees of freedom).

2.1.4 Quadruped robots

Quadruped robots, due to their 4-legged configuration, have a center of gravity that is often at the intersection of the mount of the limbs so it does not require special means to maintain balance while moving. Due to this characteristic, and many others explained further in sub-section 2.1.4.3, it has become the most popular and most researched type of legged robot.

2.1.4.1 Types of quadruped robots

In the realm of traditional quadruped robots, they can be classified into two main types based on their configuration: mammal-type (Figure 2.5) and sprawling-type (Figure 2.6) quadrupeds [5].

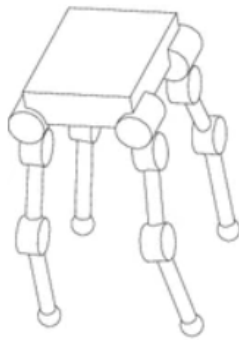


Figure 2.5: Mammal-type quadruped robot [5].

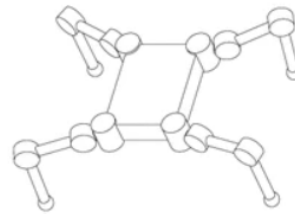


Figure 2.6: Sprawling-type quadruped robot [5].

Mammal-type quadruped robots (displayed in Figure 2.5) mimic the movement of horses or dogs. These types of quadruped robots have their legs set up beneath their body, which locates its foot vertically downward from the hip joint as a standard pose [5]. This configuration allows mammal-type robots to not only often exhibit more energy-efficient locomotion compared to sprawling-types [20], but also a faster walk by using two actuators in each leg [5].

Unfortunately, this type of quadruped robot is (usually) harder to build and program due to its dynamic gaits. In a dynamic gait, the robot's center of gravity shifts during movement, so to achieve stability, two or more legs move at the same time [5] this concept is further described in sub-section 2.1.4.2.

This has not stopped a growing interest in the research of these robots to develop. An example of this is the rise to fame of Boston Dynamics' Spot [6] (displayed in figure 2.7), a direct successor of Big Dog [21] [22]. This viral robot was developed to automate, sense, inspect, capture data, and "explore without boundaries" [6].



Figure 2.7: Boston Dynamics' Spot [6].



Figure 2.8: Boston Dynamics' Spot equipped with a manipulator arm [6].

Spot (Figure 2.7) is capable of achieving a top speed of 1.6 m/s and is stable at a max slope of $\pm 30^\circ$ [6]. It is also scalable, meaning it allows for the use of a fleet of coordinated robots. Furthermore, Spot is open to feature expansion like the addition of an integrated arm to enable manipulation (Figure 2.8).

Another case of successful use and research of this configuration is MIT's Mini Cheetah [7] (Figure 2.9), a lightweight, inexpensive, and high-performance quadruped robot capable of achieving 2.45 m/s while weighing in at only 9 kg.

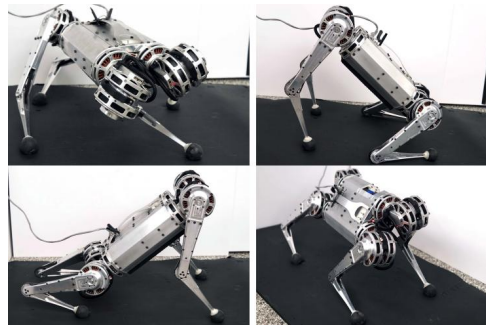


Figure 2.9: MIT's Mini Cheetah [7].

Taking advantage of its lightweight, the Mini Cheetah (Displayed in Figure 2.9) has proven to employ extreme mobility, which was further established in 2019 when it became the first robot to ever do a full 360° back-flip [23].

Sprawling-type robots (displayed in Figure 2.6) mimic the locomotion of animals with limbs that are spread out to the sides like spiders or other crawling insects, usually using a "creep gait" (displayed later in sub-chapter 2.1.4.2 in Figure 2.11) to move. These robots are configured so the first link of each leg (the thigh) is positioned horizontally and the second leg link (shank) is vertically oriented as a standard posture [5].

The positioning of the legs to the sides provides enhanced stability when compared to the mammal-type configuration, due to its wider base of support and ability to move its center of gravity to a low position [5]. Thanks to these capabilities, sprawling-type quadruped robots are extremely adaptable to irregular surfaces, making them especially well-suited for exploration in

challenging terrains. A big disadvantage of these robots in comparison to mammal-types is their walking velocity, which is generated by only one actuator [5].

However, this weakness didn't set back the research of sprawling-type robots throughout the years. One example of this is Tokyo Tech - Suzumori Endo Lab's TITAN-XIII [5] (Figure 2.10), a robot developed with the purpose of researching possible ways to improve the speed and energy efficiency of sprawling-type quadruped robots.



Figure 2.10: Tokyo Tech - Suzumori Endo Lab's TITAN-XIII [5].

To accomplish these objectives, the team behind TITAN-XIII robot [5](Figure 2.10) focused on increasing walking velocity, using a trot gait as a low-duty factor and safety gait, as well as increasing the gear ratio of the actuator supporting gravitational force [5].

2.1.4.2 Locomotion gaits

As it was previously displayed (in Sections 2.1.1, 2.1.2, 2.1.3), it is almost universal for legged robots to have an even number of legs, this is because it allows for efficient gaits and stability in performance [24]. Analyzing locomotion gaits for sprawling and mammal-type robots, some are more widely used:

For sprawling-type robots, the most common gait is the creep gait, this is especially due to its ease of implementation. It is characterized by its relatively slow movement, which is a consequence of a long setup time for each "forward propulsion". Notice the representation of a creep gait in Figure 2.11.

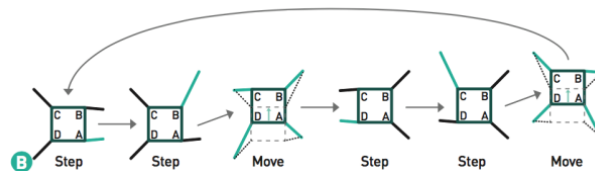


Figure 2.11: Creep gait step-by-step [8].

By using a creep gait (Figure 2.11), sprawling-type quadruped robots maintain their body close to the ground, enhancing stability and minimizing the risk of tipping. The slow setup time does

result in low overall speed, but in turn, because there are a minimum of three legs on the ground at all times preventing weight shifts, the robot is extremely stable.

On the other hand, trot and gallop gaits are the most common running gaits among all gait patterns that quadruped mammals exhibit [25]. Usually, gallop gaits are faster and trot gaits are preferred for more energy-efficient mobility [26].

When transitioning from walking to running in order to increase speed, animals typically employ trotting first, this serves as inspiration for mammal-type quadruped robots. The legs operate in diagonal pairs like it is described in Figure 2.12. This gait is regarded as a very stable dynamic gait because when each pair of legs is on the ground, a diagonal line of support passes beneath the body [9].



Figure 2.12: Horse in a trot gait [9].

Another intermediate speed gait is the pace gait, it works similarly to the trot gait (Figure 2.12), but instead of operating in diagonal pairs, it uses lateral pairs [9], as shown in Figure 2.13.



Figure 2.13: Horse in a pace gait [9].

Note that the pace gait (described in Figure 2.13) is not very stable, this is a consequence of a significant roll motion of the trunk since the legs provide support on alternating sides of the body [9].

The most well-known and common high-speed gait is the gallop gait. It is characterized by two front footfalls occurring sequentially, followed by two posterior footfalls. Generally, there is a pattern where pairs of feet do not make contact with the ground simultaneously as described in Figure 2.14. Moreover, the phase difference between the two front legs is usually greater than that observed between the two hind legs [9].

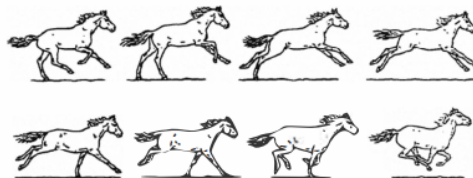


Figure 2.14: Horse in a gallop gait [9].

Note that in a gallop gait (Figure 2.14) when the front feet are lifted off the ground, the flight phase is termed "gathered". Alternatively, when the hind feet enter a flight phase, it is designated

as "extended" due to the legs extending outward from the body[9]. The gallop predominantly features the gathered flight phase, which tends to be lengthier compared to the extended phase, primarily observed at higher speeds [9].

2.1.4.3 Advantages

Quadruped robots take advantage of four legs instead of wheels or tracks as a method of locomotion, this allows them to traverse environments that otherwise would be impossible (stairs) or time and energy-intensive (navigating around an obstacle). This translates into better overall mobility in nature since each leg is its stand-alone support point contrasting with wheeled locomotion which requires continuous surface support [20]. By taking advantage of the isolated configuration of each leg, quadruped robots can stabilize themselves to adapt to irregular terrain. It is also noticeable that, because of the leg's servo configuration that results in multiple degrees of freedom, quadruped robots can alter their direction without slippage [20]. The use of legs also allows these robots to, when walking on non-rigid surfaces, deform the terrain considerably less when in comparison to tracked locomotion robots, meaning it not only is more energy efficient in this situation (because it requires less energy to exit the depressions made by previous trails) it is also often a more eco-friendly solution to tracked locomotion. It is for these reasons that legs are considered inherently suitable systems for locomotion on uneven terrain [20].

It is important to notice that legged locomotion not only benefits the robot concerning its mobility, it also can be utilized to manipulate objects. By stabilizing the robot on two or three legs, it is possible to interact with objects using the remaining leg [20].

2.1.4.4 Disadvantages

It should be emphasized that legged locomotion is neither perfect nor better than wheeled or tracked locomotion. It is considerably slower than any of the traditional methods, while still being more expensive to mass produce, and complex to develop and research. The use of multiple servos to achieve the desired degrees of freedom and the constant processing of sensor data also results in very energy-intensive robots [20]. In addition, these types of robots require complex control algorithms [20] to accurately control movement and stabilize the robot dynamically.

2.2 Degrees of Freedom

Among the critical design considerations, the number of degrees of freedom (DOFs) in each limb stands out as a key factor influencing the robot's range of motion and agility. It is important to explore the effects of incorporating three degrees of freedom in each leg of the robot and how this design choice contributes to its overall performance. In order to analyze the inner workings of legs with three DOFs, subsequently are visual representations of a sprawling-type leg (Figure 2.15), a mammal-type leg (Figure 2.16), and a brief description of these configurations.

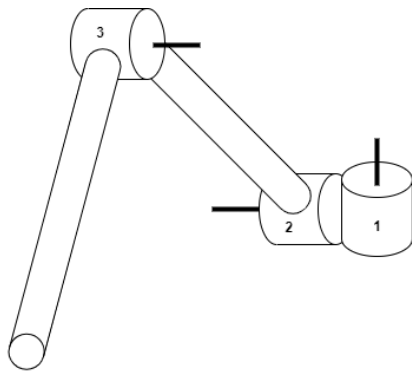


Figure 2.15: Design drawing of a sprawling-type quadruped leg with three degrees of freedom.

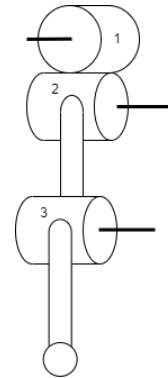


Figure 2.16: Design drawing of a mammal-type quadruped leg with three degrees of freedom.

Analyzing Figure 2.15, it can be concluded that to achieve three DOFs in a sprawling type leg, three revolute joints are implemented. Joint 1 (coxa joint), provides rotation in the yaw axis, enables the leg to move side to side, and is responsible for the initiation of movement. Joint 2 (femur joint), provides rotation on the pitch axis, allows the limb to flex and extend, and is instrumental in the adjustment of the limb's height during locomotion. Joint 3 (tibia joint) is crucial for adapting the foot to the terrain, providing stability.

On the other hand, in order to employ three DOFs for a mammal-type configuration 2.16, three revolute joints also need to be implemented, but with different axis rotations for each joint. Joint 1 (hip/shoulder joint) is a rolling joint, meaning it provides rotation around the roll axis which rotates the leg in the vertical axis. This allows the leg to swing up and down. Joint 2 (thigh/elbow joint), associated with the pitch axis, allows for flexion and extension of the limb, enabling the robot to maintain balance and stability while walking or running. Finally, Joint 3, (shank/wrist joint), similarly to the thigh joint, engages in pitch motion. This is crucial for adapting the foot to the terrain, meaning it serves the same purpose as the tibia joint on a sprawling robot leg 2.15.

Having more DOFs enhances the mobility of the robot by providing an enhanced level of locomotion flexibility [27]. When compared to robots with only two DOFs, this limb configuration can execute a more extensive range of movements, allowing for dynamic and adaptive responses to changes in the terrain. This is also a consequence of improved stability and balance due to the ability of the robot to adjust its height dynamically, optimizing its center of gravity and minimizing the risk of losing balance and tipping.

Most importantly, the inclusion of three DOFs allows for more precision in leg placement due to an increased range of motion [27]. It also enables the robot to reach surfaces that were not possible with fewer degrees of freedom.

2.3 Servo Motor Control Methods

Servo motors play a crucial role in the development of legged robots, they allow the movement of joints which in turn enables the robot to move. To control this movement with precision, servo

control protocols are employed, serving as a communication line between a microcontroller and the servo motor, allowing the controlling device to dictate the speed, and position of the servo motor.

Two prevalent servo control protocols, namely Universal Asynchronous Receiver-Transmitter (UART) and, Pulse-Width Modulation (PWM) stand out as the most popular methods for achieving accurate and responsive control over servo motors.

2.3.1 UART

UART (Universal Asynchronous Receiver Transmitter) is a serial communication protocol mostly used for short-distance, low-speed, low-cost data exchange [10]. UART allows for full-duplex communication, which implies simultaneous spectrum sensing and transmission on a given channel [28]. Basic UART communication uses two signal lines (RXD, TXD), the TXD signal line being the transmission (output) and the RXD signal line being the receiver (input) [10].

There are two possible states in the signal line, and to distinguish them, the protocol uses binary logic. This means that, when the transmitter is idle, the data line is in the high logic state. Otherwise, a bit called the "Start Bit" is prepended to each data word to synchronize the receiver with the sender. This Start Bit serves as a signal to the receiver that a new data word is about to be transmitted [10]. Note the following Figure 2.17 where the UART frame format is displayed.

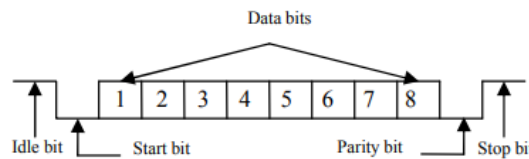


Figure 2.17: UART frame format [10].

As it can be observed in Figure 2.17, after the Start Bit, the individual data bits of the word are sent, with the Least Significant Bit (LSB) being first. Then, after the whole word has been sent, the transmitter line is capable of adding a Parity Bit to allow the receiver to perform an error check. Finally, a Stop Bit is sent to indicate the end of the transmission.

In the receiver end, if the Stop Bit is not received, the UART considers the entire word to be corrupted and consequentially reports a Framing Error to the host processor after the data word is read. Usually, a Framing Error is a consequence of the receiver and transmitter clocks not being synchronized or that the signal was interrupted [10]. Regardless if the data is correctly received or transmitted, the UART automatically discards the Start, Parity, and Stop bits.

2.3.2 PWM

Pulse-width modulation (PWM) is a modulation technique that generates variable-width pulses to represent the amplitude of an analog input signal [29], in the scope of robotics, it is a technique used to control analog circuits utilizing a microcontroller's digital outputs.

Note in Figure 2.18 how PWM works by outputting a repeatedly pulsing digital signal high and low at a fast rate which results in an effective average voltage.

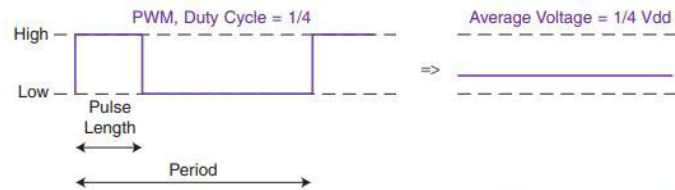


Figure 2.18: Pulse-width modulation and resulting average voltage [11].

The period of the signal is typically constant, and it is by altering the duty cycle that it is possible to control the average voltage [30] as it is displayed in Figure 2.18.

However, some devices, such as servos, do not rely on the power of the signal limited by PWM, but instead use the width of the high pulses to transmit information [11]. In most PWM-controlled servos, such as the SG90 [31], there are three pins: power, ground, and a control signal. The control signal is a PWM input where the width of the high pulse determines the position of the servo's angular position [11]. Internally, the servo motor compares the incoming PWM control signal against a signal it generates itself. The pulse widths of this internally generated signal are adjusted by the potentiometer, which determines the shaft angle. The servo motor then aligns these pulse widths by appropriately rotating the motor shaft[11].

2.3.3 Comparison

In order to understand the differences in the use of UART and PWM, analyzing and comparing them is extremely important.

The most relevant advantage UART has over PWM in the scope of servo control is its very flexible command structure, allowing complex commands with information about position, speed and other parameters to be transmitted. This means that it is possible to not only do position control, but also simultaneously, and relatively simply, speed control using UART.

UART also provides bidirectional communication between the microcontroller and the servos, allowing both devices to communicate with each other. This means that the microcontroller not only sends commands to the servo but also receives feedback and status updates from it. As a result, the microcontroller can receive real-time updates from the servo, including information about its current position, temperature, and voltage. This functionality allows the microcontroller to have precise knowledge of the robot's limb positions at any given moment giving the microcontroller not only the ability to implement dynamic control strategies but also to take corrective actions in the event of a malfunction, increasing its fault tolerance.

However, PWM also has some benefits over UART, namely its compatibility with most microcontrollers and servos. Unfortunately, some UART-protocolled servos, require specialized bus-servo boards (like the LX-16A [32] and the LX-225 [33] or the ST3215 [34]) in order to be

controlled. Also, PWM is more cost-effective when compared to UART, not only that but it is also noticeably more straightforward, which is why most hobby robotic projects usually use these types of servos.

2.4 Transmission Control Protocol

Transmission Control Protocol (TCP) is a highly reliable protocol primarily due to two key features: Firstly, TCP is connection-oriented, which means it establishes a connection between the two systems before data transfer begins, ensuring both parties are ready for communication. Secondly, it uses sequence numbers and acknowledgments, allowing for the re-transmission of lost or corrupted segments.

TCP generally consists of three phases: 1) Connection establishment, 2) Data transfer, and 3) Connection termination [35]. These stages are displayed in Figure 2.19.

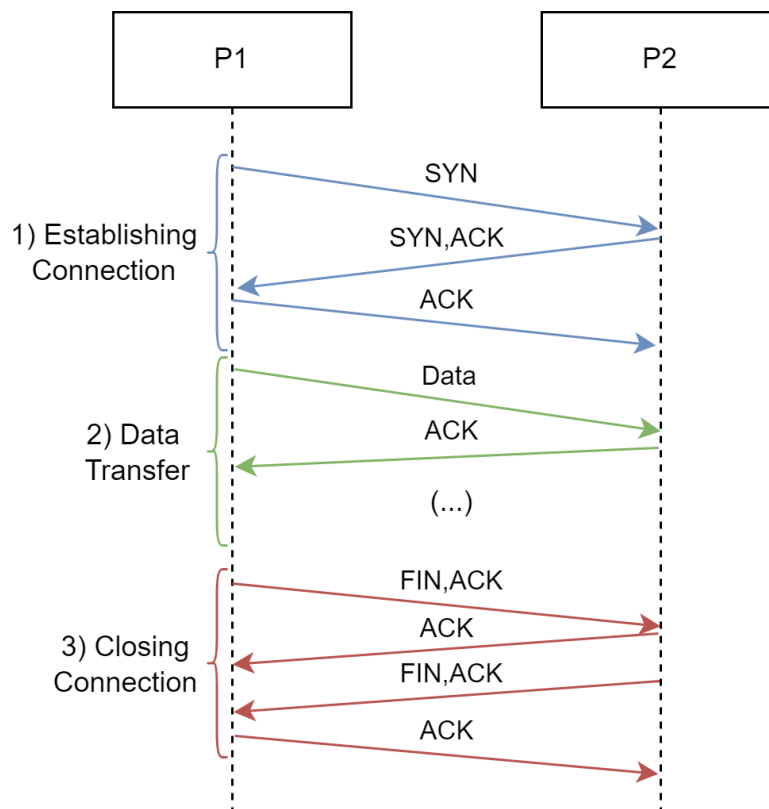


Figure 2.19: TCP Sequence diagram

Observing Figure 2.19, it is evident that when establishing the connection, TCP relies on a 3-way handshake. In this process, the client (P1) sends a SYN (Synchronize) packet to the server (P2) to initiate a connection, who in return responds with a SYN-ACK packet acknowledging the request and sending its own SYN to establish connection. The client finally confirms the receipt of the server's SYN-ACK via an ACK packet.

During the data transfer phase, all packets are acknowledged, which allows for truly reliable communication. Finally, when closing the connection, the client sends a FIN (Finish) segment to the server, indicating that it has finished sending data. the server acknowledges this, and then sends its own FIN packet, which the client acknowledges.

2.4.1 Header Format

Note in Figure 2.20 a visual representation of the TCP header format. The source and destination port, as the name suggests, indicate the sender’s and receiver’s ports. The sequence number is responsible for helping reassemble the data in the correct order even if the packets arrive out of sequence. It achieves this by identifying the position of the first byte of data in the segment within the overall byte stream being sent. The acknowledgment number confirms the reception of data, and indicates the value of the next sequence number the sender of the segment is expecting to receive [36]. The Data Offset (DO) indicates where the data begins and the reserved bits should be set to 0. The flags are the control bits such as SYN, ACK, FIN, etc. that are used to manage the connection state. The Window Size specifies the number of bytes that the sender of this segment is currently willing to receive. The Checksum is used for error checking in the header and data sections, while the Urgent Pointer points to urgent data if the URG flag is set. The Options section, which has variable size, provides additional functionalities such as setting maximum segment size, enabling timestamps, or utilizing selective acknowledgements [36].

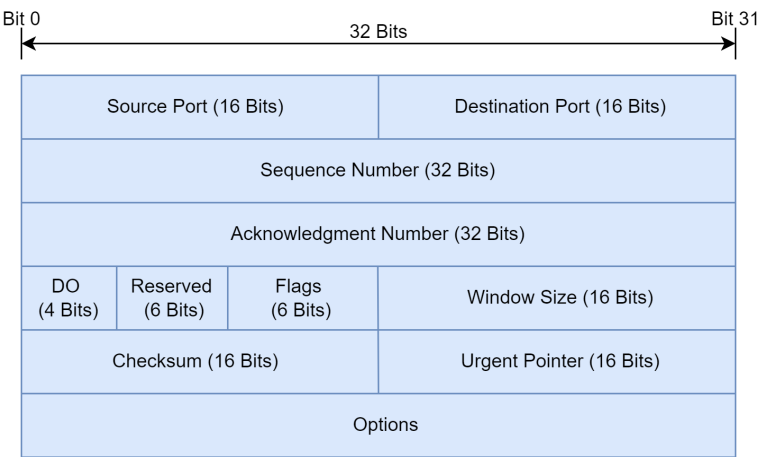


Figure 2.20: TCP Header Frame

2.5 Conclusion

In this chapter, multiple types of legged robots were reviewed, with special emphasis on quadruped robots. The importance of degrees of freedom in this area of robotics was also studied, as well as the functioning of the most common servo control protocols. Finally, an overview of the

Transmission Control Protocol (TCP) was provided, highlighting its fundamental principles and header format.

The literature review revealed the several advantages of mammal-type quadruped robots when compared to sprawling types. The differences in speed and energy efficiency, paired with the challenge of developing a gait capable of reaching "high speeds" became a crucial aspect for the decision on what type of robot would be developed.

Using three degrees of freedom has also proven to be essential to achieve the desired mobility and gaits.

Additionally, having a full understanding of the capabilities of UART in comparison to PWM proved to be extremely important. To achieve dynamic stability and make adjustments during movement, having the ability to retrieve servo feedback to ensure the correct position or speed is crucial.

After reviewing this information, it was decided that the developed robot would be a mammal-type quadruped robot with three degrees of freedom utilizing UART-controlled servos and TCP communication.

Chapter 3

Quadruped Robot

This chapter details the development process of the quadruped robot and is divided into four Sections. It starts with a description of the initial model for the robot in Section 3.1. Section 3.2 presents the modifications made to the used 3D model. Following that, an analysis of the servos used for the robot, along with a comparison with other popular servos is presented in Section 3.3. Finally, in Section 3.4 the kinematic analysis of the the legs is performed.

3.1 Initial Model

In the initial stages of the study, the plan was to use existing 3D models (from the online platform Thingiverse) in order to expedite the development process. This approach aimed to minimize the time and effort spent on the design phase by utilizing an established model and making only minimal modifications to suit the project's requirements. Two potential 3D models were initially considered for this purpose.

One of the considered models was DOGGO [12], which, although compatible with the initially intended servos, was ultimately not selected. The plan was to utilize LX-16A [32] servos, which offer a maximum torque of 16 kg.cm powered with 7.4 V and feature UART control via a dedicated servo serial communication board. Note the model in Figure 3.1.

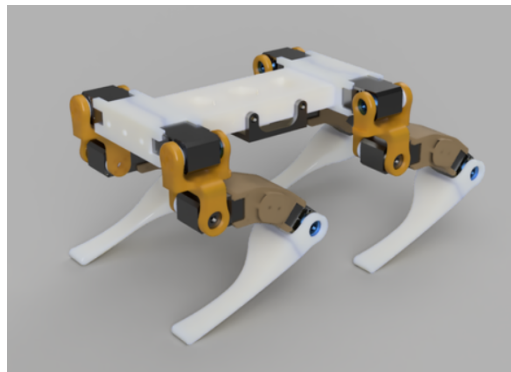


Figure 3.1: DOGGO 3D Model by Jakub Pływaczewski [12].

While this model matched the servo specifications, it appeared somewhat fragile and potentially unsuitable for the robust demands of a quadruped robot. Concerns about its structural integrity and durability during extended use led to the decision to seek alternative options.

Additionally, due to availability and shipping issues, the LX-16A servos were also no longer a viable option as the chosen servo. As a way to resolve this issue, a more powerful servo was chosen: the Waveshare ST3215 [34]. These servos provided a significant increase in torque, boasting a maximum of 30 kg.cm.

Similarly to the LX-16A, these servos used UART control, but instead of a separate communication board, the communication board is integrated with an ESP32. This change also required an adjustment in the power supply, as the ST3215 servos operate at 12 V.

The second 3D model (a SpotMicro [37]), which was eventually chosen, offered plenty of online resources and support. The detailed modifications made to this 3D model are discussed in the following section (Section 3.2), where the necessary structural modifications, adjustments for different servo configurations, and enhancements for overall functionality and performance are thoroughly explained.

3.2 3D Model

As described in the previous section (Section 3.1), for the development of this study an existing 3D model of a SpotMicro [37] (inspired by a Boston Dynamics' Spot [6]) was considered. The model provided a solid foundation, however, several modifications were necessary to align it with the requirements of the research.

3.2.1 Link and Joint Modifications

As will be discussed further into the document in Sub-Section 4.4.1 in Chapter 4, the initial leg links led to higher torque spikes and overdrive torque due to their length and weight. In order to mitigate this problem, the legs of the robot were shortened to decrease the lever arm effect, thus requiring less torque for movement.

As shown in Table 4.3 in Sub-Section 4.4.1, the elbow and wrist links were modified, while the shoulder link remained unchanged.

The joints were also redesigned to accommodate Waveshare ST3215 servos. The original joint design was incompatible with the dimensions and mounting requirements of these servos, as they were designed for MG996R servos. Note in Table 3.1 the differences between these servos.

Table 3.1: ST3215 and MG996R servo dimensions and mounting type.

Servo	Dimensions	Mounting Type
ST3215	45.2mm × 35.0mm × 24.7mm	Internal
MG996R	40.7mm × 19.7mm × 42.9mm	External

Observing the model differences in Table 3.1, it is clear that the joints had to be completely redesigned, in order to fit the ST3215 as well as feature an internal servo mounting system, where the screws are housed within the robot's structure.

To better visualize these modifications, note in Figure 3.2 the original (Figure 3.2a) and modified (Figure 3.2b) shoulder joints.

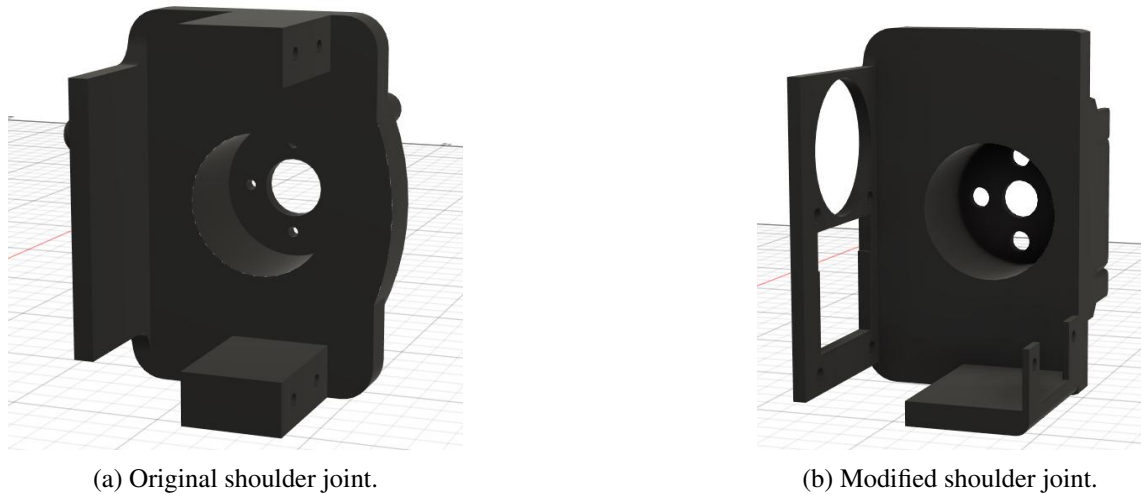


Figure 3.2: Shoulder joint modifications.

Notice in Figure 3.2, how the mounting points had to be changed in order to be compatible with ST3215 servos. It is also important to note that there are now two holes at the back of the joints. The round hole fits a free-spinning gear that is not included in MG996R, while the square-shaped hole allows easy access to the servo ports.

Now note in Figure 3.3, a side-by-side comparison between the original and modified legs.

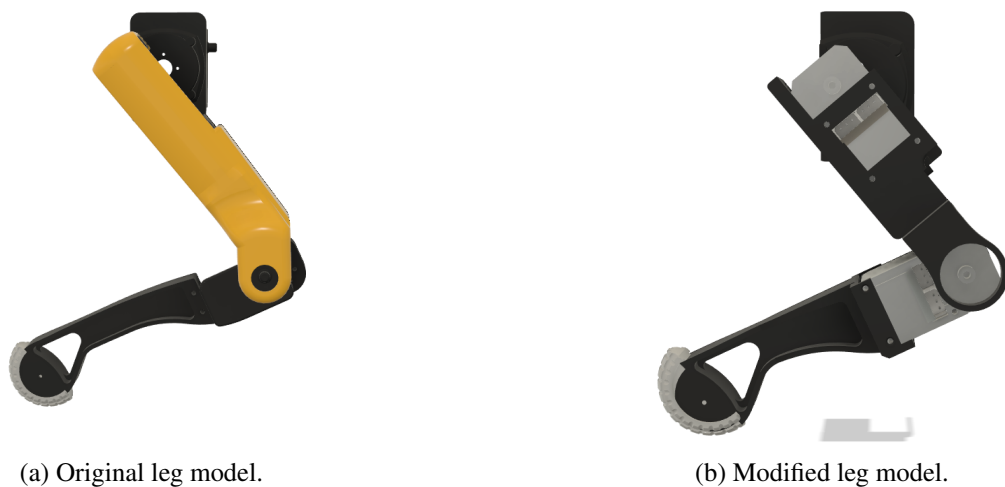


Figure 3.3: Leg modifications.

Analyzing Figure 3.3 it is clear that the wrist and elbow link were shortened, which also allowed for a substantial weight reduction. This was specially evident in the elbow link, where the new link model design (Figure 3.3b) is 2.045 times lighter than the original (Figure 3.3a).

3.2.2 Enhancements and Further Modifications for Improved Performance

To further reduce the model's weight, non-structural covers were removed to eliminate any unnecessary load. Although aesthetically pleasing, the original covers did not contribute to the robot's functionality. Note in Figure 3.4 the the original and final modified robot.



(a) Original full robot model with covers.



(b) Modified full robot model.

Figure 3.4: Full robot modifications

Removing the robot covers allowed for the elimination of 218g, which is a very significant cut off the total model weight.

It is also relevant to mention that, because the model was printed using PLA plastic, the robot would frequently slip during movement. To mitigate this problem and add traction to the robot, a layer of liquid rubber was applied to the bottom of each of the legs' foot.

Figure 3.5 displays photos of the full robot along with the side profile of the real robot leg.



(a) Real robot



(b) Modified full robot model.

Figure 3.5: Full robot modifications

3.3 Waveshare ST3215 Servo Comparisons

It is very common for quadruped robots to be built using low-cost servos like the SG90. In this dissertation, however, a higher-end type of servo will be used: the Waveshare ST3215 [34], which is a high-precision, high-torque actuator that allows for precise and fluid movement, and most importantly, UART (Universal Asynchronous Receiver-Transmitter) servo control, which provides real-time monitoring and speed control.

3.3.1 Comparison to other servos

To better understand the advantages of opting for the ST3215 servos over hobby servos, a comparison table that outlines crucial parameters for quadruped robots is presented below (Table 3.2). In order to include a servo of the same class as the ST3215 servo, an MG996R Tower-Pro will also be analyzed, a classic servo choice for legged robotics.

Table 3.2: Comparison between the SG90, MG966R, and ST3215 servos.

	Weight	Voltage	Max Torque	Control	Gear	Error	Operating Angle
SG90	9g	4.8-5V	1.8kg.cm	PWM	Plastic	$\pm 5^\circ$	360°
ST3215	69g	6-12.6V	30kg.cm	UART	Metal	$\pm 0.1^\circ$	360°
MG966R	55g	4.8-7.2V	11.4kg.cm	PWM	Metal	$\pm 3^\circ$	120°

Examining Table 3.2, it is evident that the ST3215 is considerably more powerful than the SG90 and the MG966R. Most noticeably, the maximum torque the ST3215 servo is capable of outputting is 16.6 times larger than the SG90. This translates into increased strength and stability for tasks requiring more power, like climbing obstacles or standing on three legs.

The difference in the servo positional error is also very apparent, the parameter describes positional accuracy, which refers to how closely a servo can reach and hold a commanded position. SG90's and MG966R's servo error of $\pm 5^\circ$ and $\pm 3^\circ$ implies a noticeable deviation. While this may be acceptable in some applications, it could impact the precision of movements. The ST3215 displays a significantly lower servo error, implying a higher degree of precision.

While at first glance the gear types might not seem like the most important parameter of comparison between the two servos, they are in fact one of the biggest differences between ST3215 and SG90 servos. Metal gears contribute to the durability and reliability of ST3215 and MG966R servos, while SG90's plastic gears can wear out over time, decreasing its positional accuracy (due to possibly "jumping" worn down gear teeth) or even resulting in servo breakdowns.

The control method is the most relevant difference, due to PWM's (Pulse Width Modulation) programmability limitations and the advantages UART (Universal Asynchronous Receiver-Transmitter) control has to offer. This topic was expanded upon further in Chapter 2 in Section 2.3.3.

So the main reasons to move away from the SG90 were the relatively low precision and resolution of this servo, the limited programmability and responsiveness of its control method, and the non-suitability for the loads a quadruped robot requires due to the very limited torque and plastic

gears. MG966R would be a possible candidate if it used a UART control method, this would consequently allow for better accuracy, which is another problem these servos carry.

3.4 Kinematics

A kinematic model is a representation of the motion of a system without considering the forces or torques that cause the motion. It describes the relationship between the different components of a system and how their movements impact each other. In the context of robotic systems, this typically involves the study of positions and movement of joints and links.

Kinematic models simplify the analysis of the system's motion and are particularly useful when dealing with systems where the dynamics (forces and torques) are not the primary focus.

To later ease the development of algorithms, a kinematic model of a mammal-type quadruped was developed. Note its simplified representation in Figure 3.6.

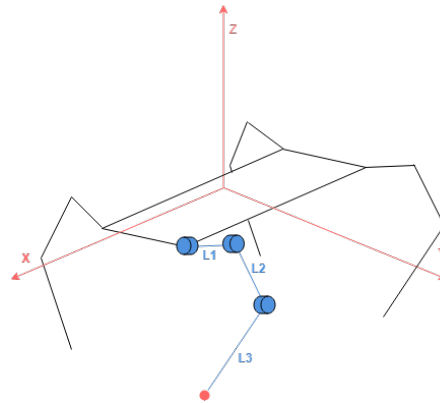


Figure 3.6: Representation of mammal-type quadruped robot.

3.4.1 Inverse Kinematics

Inverse kinematics is a method that deals with the inverse problem of determining the joint configurations required to achieve a desired position and orientation of the end-effector. This technique is particularly useful for tasks that involve accurate positioning or manipulation. Solving this problem is necessary to convert motion specifications assigned to the end-effector in the operational space into the corresponding joint space motions [27].

Figure 3.7 shows the front perspective of a mammal leg:

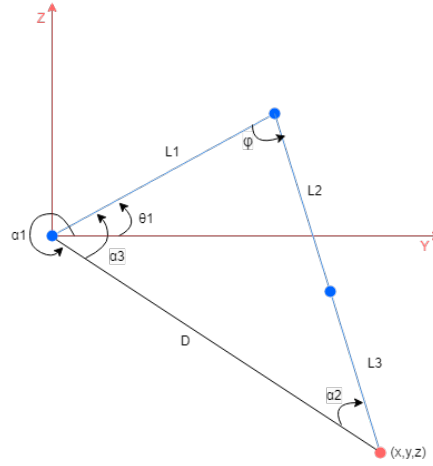


Figure 3.7: Front perspective of the mammal leg.

In this perspective (Figure 3.7), the goal is to find the θ_1 using the coordinates x, y, z . To do so, it was considered that ϕ is a constant angle that varies from model to model.

First, the length of D and α_1 is calculated:

$$D = \sqrt{z^2 + y^2} = \text{norm}(y, z) \quad (3.1)$$

$$\alpha_1 = \text{atan2}(z, y) \quad (3.2)$$

With length D and knowing that:

$$\frac{\sin(\alpha_2)}{L_1} = \frac{\sin(\phi)}{D} \quad (3.3)$$

It is deduced that:

$$\alpha_2 = \sin^{-1} \left(\frac{L_1}{D} \sin(\phi) \right) \quad (3.4)$$

In order to find α_3 , using the triangle sum theorem:

$$\alpha_3 = \pi - \alpha_2 - \phi \quad (3.5)$$

With all this information, θ_1 can be computed:

$$\theta_1(\text{right}) = \alpha_1 - \alpha_3 \quad (3.6)$$

$$\theta_1(\text{left}) = \alpha_1 + \alpha_3 \quad (3.7)$$

To ease the analysis of the side perspective, the referential is first rotated with an angle R in Figure 3.8

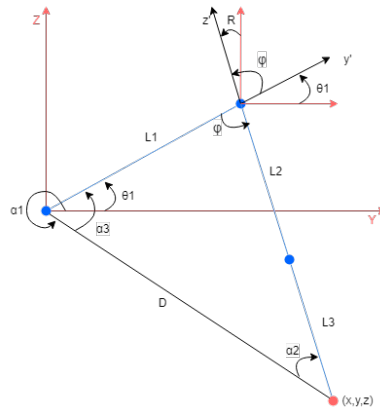


Figure 3.8: Rotation of referential.

To calculate R:

$$R = \theta_1 + \phi - \frac{\pi}{2} \quad (3.8)$$

Using this, the point x, y, z was rotated by $-R$ using a transformation matrix:

$$(x_2, y_2(=0), z_2) = T \cdot xyz \quad (3.9)$$

Analyzing the side perspective in Figure 3.9:

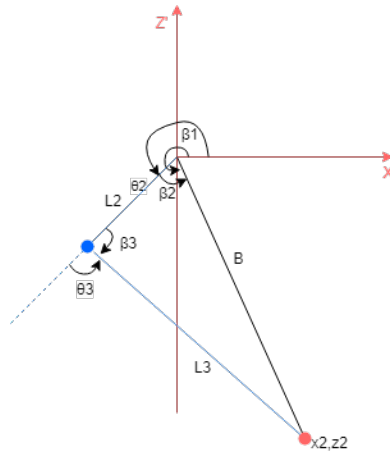


Figure 3.9: Side perspective of mammal leg.

First, calculate the length of B and β_1 :

$$B = \sqrt{x_2^2 + z_2^2} \quad (3.10)$$

$$\beta_1 = \text{atan2}(z_2, x_2), 0 \leq \beta_1 \leq 2\pi \quad (3.11)$$

Then β_2 and β_3 is deduced

$$\beta_2 = \arccos\left(\frac{L_2^2 + B^2 - L_3^2}{2 * L_2 * B}\right) \quad (3.12)$$

$$\beta_3 = \arccos\left(\frac{L_2^2 - B^2 + L_3^2}{2 * L_2 * L_3}\right) \quad (3.13)$$

Therefore,

$$\theta_2 = \beta_1 - \beta_2 \quad (3.14)$$

$$\theta_3 = \pi - \beta_3 \quad (3.15)$$

3.4.2 Forward Kinematics

Forward kinematics is a method used to predict the position and orientation of a robot's end-effector based on its joint configurations. This technique is particularly useful for robotic manipulators, as it can answer the question of where the end-effector will be in space, given the current joint angles or positions. This mathematical representation allows the mapping of the joint space to Cartesian space, facilitating the observation and development of the robot's motion. Let's again analyze the front perspective in Figure 3.10:

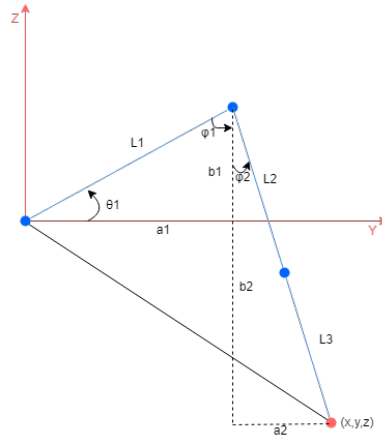


Figure 3.10: Front perspective of the mammal leg

In order to calculate the end effector, the components of triangle 1 must be deduced as such:

$$b_1 = \sin(\theta_1) \cdot L_1 \quad (3.16)$$

$$a_1 = \cos(\theta_1) \cdot L_1 \quad (3.17)$$

$$\phi_1 = \pi - \theta_1 - \frac{\pi}{2} \quad (3.18)$$

Now, calculating the front perspective distance of $L_2 + L_3$, which will be referred to as D :

$$D = L_2 \cdot \cos(\theta_2) + L_3 \cdot \cos(\theta_2 + \theta_3) \quad (3.19)$$

It is known that D is the hypotenuse of triangle 2:

$$\phi_2 = \phi - \phi_1 \quad (3.20)$$

$$a_2 = \sin(\phi_2) \cdot D \quad (3.21)$$

$$b_2 = \cos(\phi_2) \cdot D \quad (3.22)$$

Now using the cathetus calculated for the two triangles:

$$y = a_1 + a_2 \quad (3.23)$$

and

$$z = b_2 - b_1 \quad (3.24)$$

The transformed side perspective can now be used as shown in Figure 3.11.

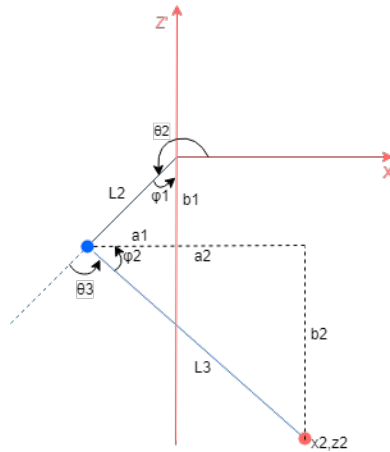


Figure 3.11: Side perspective for direct kinematics.

Calculating the components of the triangle 1:

$$\phi_1 = \frac{3\pi}{2} - \theta_2 \quad (3.25)$$

$$a_1 = \sin(\phi_1) \cdot L_2 \quad (3.26)$$

$$b_1 = \cos(\phi_1) \cdot L_2 \quad (3.27)$$

And triangle 2:

$$\phi_2 = \pi - \theta_3 - \left(\pi - \frac{\pi}{2} - \phi_1 \right) \quad (3.28)$$

$$a_2 = \cos(\phi_2) \cdot L_3 \quad (3.29)$$

$$b_2 = \sin(\phi_2) \cdot L_3 \quad (3.30)$$

It can be deduced that:

$$z_2 = b_1 + b_2 \quad (3.31)$$

$$x_2 = a_2 - a_1 \quad (3.32)$$

In order to undo the referential rotation, knowing that:

$$x_2 y_2 z_2 = T \cdot xyz \quad (3.33)$$

The inverse matrix of T is used:

$$xyz = T^{-1} \cdot x_2 y_2 z_2 \quad (3.34)$$

Chapter 4

Simulation

This chapter outlines the procedure for simulating the robot and using the simulator to optimize the robot's gait and structure. This chapter is divided into seven Sections. First, Section 4.1 describes an overview of the simulator. Subsequently, Section 4.2 details the steps taken during the simulation loop along with an explanation of the key classes of the simulator architecture. Section 4.3 presents the initial analysis of the torque evolution during gaits that lead to torque peak gait optimization presented in Section 4.4. Next, Section 4.5 focuses on further optimization, taking into account distance traveled as well as torque peak, using the SHGO algorithm. It is in Section 4.6 that multi-objective optimization is used and studied. Finally, in Section 4.7, the final optimization result for the robot is defined.

4.1 Overview

Simulation offers a risk-free platform for experimentation. It allows the testing of algorithms, models, and designs before deploying them in the physical world. This not only saves time and resources but also mitigates the potential for damage to expensive hardware.

Pursuing the validation of the kinematic model previously calculated, a Pybullet simulator developed by the Spot Micro community [38] was adapted and used. Originally developed to be used as a Gym environment for Reinforcement Learning tasks, this simulator was found to be suitable for the purpose of testing, even if not utilizing its full potential.

The simulator provided a Unified Robot Description Format (URDF) model of the Spot Micro (the original 3D model that was modified for this dissertation - Section 3.2 in Chapter 3), which is an Extensible Markup Language (XML) file type that includes the physical description of the robot (such as its links, joints, structure, and collision space). It also already contained an integrated Open-Loop Gait using 12-point Bezier Curves [39] that allowed for a further understanding of the weight distribution of the robot during motion. Figure 4.1 illustrates the simulated robot in Pybullet.

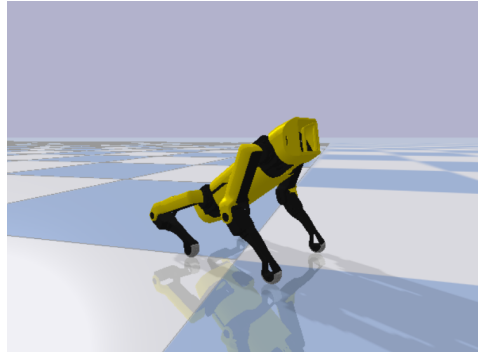


Figure 4.1: Simulated quadruped robot sitting.

Since the simulator also provided leg and full body kinematics (that were later modified to match the kinematics calculated in section 3.4) it allowed for testing of the controlled movement of the robot's body in every direction.

```
Shoulder Angle Calculated: 0.36 Elbow Angle Calculated: 1.03 Wrist Angle Calculated: -1.87
Shoulder Angle Simulator: 0.36 Elbow Angle Simulator: 1.03 Wrist Angle Simulator: -1.87
```

Figure 4.2: Resulting joint angles comparison for the same end-effector position.

It also allowed for comparison in the resulting joint angles of the provided IKs (Inverse Kinematics) and the calculated IKs when faced with the same movement. The resulting angles matched perfectly as expected (Figure 4.2).

4.2 Simulation Loop

The simulation loop is a fundamental element of the quadruped robot's virtual environment, responsible for maintaining the robot's state, processing gait parameters, and controlling movements. By analyzing the loop, it is possible to understand how various subsystems function together to achieve realistic motion.

To do so, a sequence diagram describing the simulation loop was drawn. The diagram visually represents the interactions between the following key classes:

- **spotBezierEnv:** Responsible for simulating the locomotion of the quadruped robot. The state space includes the angles, velocities and torques for each servo, while the action space describes the desired angles.
- **spot:** Encapsulates the physical parameters of the robot, such as the length of its limbs and range of motion of each joint. It is also responsible for describing the kinematic model of the robot.
- **BezierGait:** Responsible for generating the gait for the robot using Bezier Curves [39]. This involves calculating the phase of the gait, which can be either "Stance" or "Swing", based on the duration of the stance and swing phase, as well as the total stride period. Once the phase is determined, it is used to calculate the desired foot positions.

- **BezierStepper:** Initializes parameters related to the robot's motion, such as *Step Length*, *Step Velocity*, *Lateral Fraction*, *Yaw Rate*, *Clearance Height* and *Penetration Depth*, as well as the position and orientation of the robot.
- **LegIK:** Performs the inverse kinematics calculations for a leg, meaning that it determines the joint angles required to achieve a desired pose.

Before the loop, all key classes are initialized, as well as the GUI. In the spot initialization, robot's physical parameters are defined, allowing for tweaking of the kinematic model. During the initialization of the BezierGait and BezierStepper, the timestep is also defined (considering the simulation timestep). Note in Figure 4.3, the sequence diagram that describes the interactions between classes during the loop, for legibility purposes, the parameters returned by the Bezier-Stepper are denoted by an asterisk (*), the parameters include the ones defined in the previously, as well as the robots position and orientation (*pos* and *orn*).

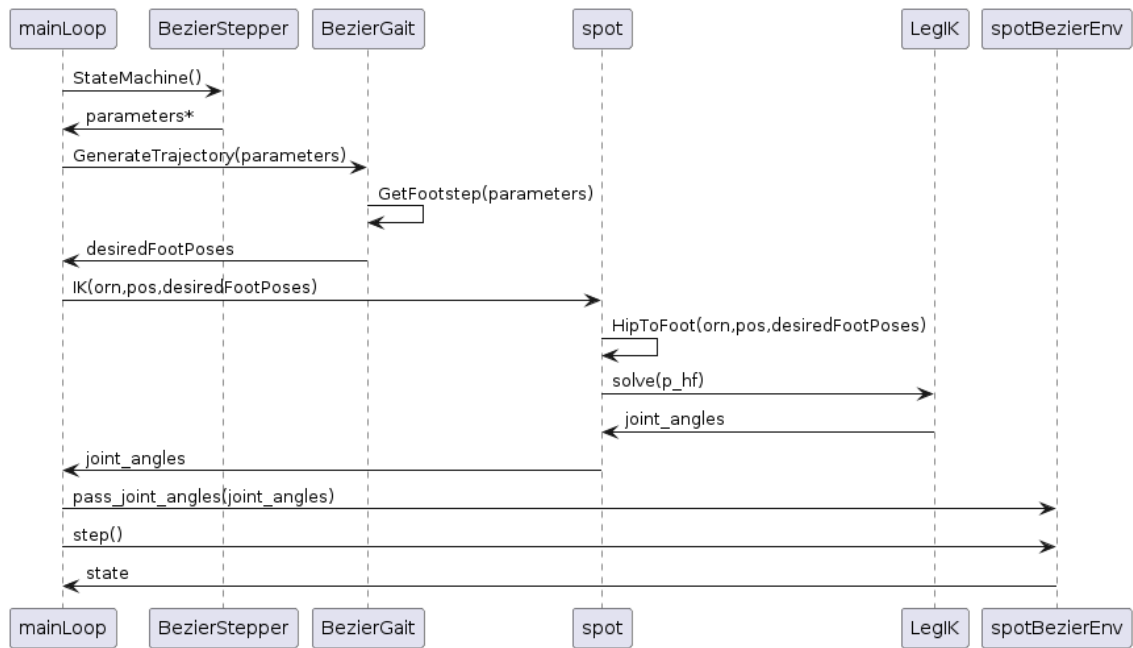


Figure 4.3: Simulation loop UML sequence diagram.

Analyzing Figure 4.3, firstly, the simulation loop requests the state of the robot (position and orientation), as well as the gait parameters it requires in order to generate the leg trajectories and foot poses. Subsequently, these foot poses are ran through the inverse kinematics model to calculate the required joint angles for each leg. Finally, the simulator environment receives these joint angles, executes them, and steps forward in the simulation, returning the state of the robot.

A more in-depth analysis of the gait generation is described further into the document in Chapter 5, Section 5.1, where the process is used for the real robot.

4.3 Torque Study

To further understand how the torque each servo exercised evolved during movement (in this case, using the 12 Point Bezier Curve Gait), the torque values recorded in the simulation were exported into a .CSV (Comma-Separated Values) file. These values were then used to create line graphs that illustrate the evolution of torque throughout the movement. Note the graph that describes the Rear Right Leg in Figure 4.4.

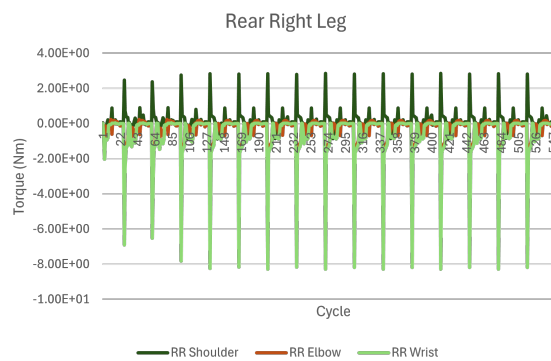


Figure 4.4: Rear Right leg torque evolution of a non-optimized model.

Upon analyzing Figure 4.4, there appears to be a noticeable pattern where each servo experiences a significant peak at a point in the gait cycle. This peak occurs cyclically when the leg comes in contact with the ground while the opposite leg rises. Note Table 4.1 which describes the maximum and minimum torque registered for each servo.

Table 4.1: Max and min torque peaks for a non-optimized model.

Servo	Max (N.m)	Min (N.m)
FL Shoulder	0.76	-1.39
FL Elbow	3.59	-0.61
FL Wrist	0.04	-2.27
FR Shoulder	1.69	-0.92
FR Elbow	4.00	-0.70
FR Wrist	0.04	-2.35
RL Shoulder	0.84	-3.76
RL Elbow	0.43	-6.06
RL Wrist	0.04	-11.62
RR Shoulder	2.84	-1.07
RR Elbow	0.24	-3.43
RR Wrist	0.04	-8.29

Analyzing Table 4.1 it is clear that, without optimizing the gait and model, not only every simulated rear servo exceeds the maximum torque an ST3215 servo is capable of exerting (2.95 N.m at 12 V - Table 3.2), but also both front elbows exceed this limit. During simulations, these

joints consistently experienced the most stress, this is a consequence of the trajectory defined by the gait which inadvertently placed greater mechanical stress on the rear legs.

4.4 Torque Peak Optimization

For the Bezier gait to be achievable with the ST3215 servos, it had to be optimized. To do so, a global optimization algorithm was developed using the following four decision variables:

- **Step Length:** describes the distance the model moves in a single step.
- **Step Velocity:** describes the speed at which the leg moves during the step swing.
- **Clearance Height:** describes the height above the ground that the foot of the robot needs to clear during a step.
- **Penetration Depth:** describes the maximum allowed depth the foot can go into the ground.

To diminish the complexity of the problem and tighten the search area, each of these variables was bound by an adequate maximum and minimum value.

The objective function executed the simulation for a predetermined amount of cycles and outputted the sum of all torque peaks that were greater than a threshold, meaning it outputted the sum of all torques that peaked at a greater value than the maximum torque of an ST3215 servo (overdrive torque).

To find the global minimum of this function, Simplicial Homology Global Optimization (SHGO) [40] was used. This algorithm divides the search space into a number of smaller regions (simplexes) and searches for local minima within those regions. The regions are then subdivided based on the results of the local searches, focusing on the ones where the function values are the lowest. It then repeats the previous steps until the change in function output values is below a specified tolerance (convergence).

Note in Figure 4.5 where the resulting torque evolution of this optimization for the Rear Right leg is displayed in a line graph.

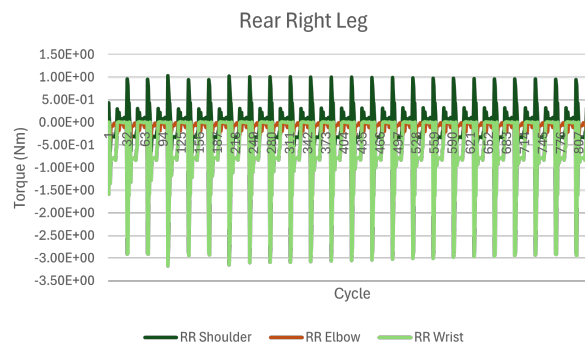


Figure 4.5: Rear Right leg torque evolution of an optimized model.

Examining the graph, it's evident that the optimized model (Figure 4.5) has significantly decreased peak torques in comparison to the non-optimized model (Figure 4.4).

This is further displayed in Table 4.2 where the largest peaks for each servo are displayed.

Table 4.2: Max and min torque peaks for a SHGO optimized model.

Servo	Max (N.m)	Min (N.m)
FL Shoulder	0.77	-1.45
FL Elbow	4.37	-0.35
FL Wrist	0.02	-1.47
FR Shoulder	1.23	-0.58
FR Elbow	-3.92	-0.30
FR Wrist	0.02	-1.48
RL Shoulder	0.60	-1.32
RL Elbow	0.01	-1.69
RL Wrist	0.02	-4.00
RR Shoulder	1.03	-0.66
RR Elbow	0.00	-1.71
RR Wrist	0.02	-3.17

Upon analyzing Table 4.2, it becomes apparent that there is a significant improvement in comparison to the non-optimized results (Table 4.1). The torque distribution along the legs is now more balanced, with significantly reduced peak values.

4.4.1 Leg Model Configuration

Although the optimization results were extremely positive, the maximum torque capacity of ST3215 was still being exceeded by the front elbows and rear wrists. A solution to this problem was the modification of the 3D model. Consider the torque definition:

$$T = d \times F \quad (4.1)$$

By shortening the leg links (lever length, d), it is possible to reduce the torque (T) required to exercise the same force (F).

Table 4.3 describes the changes made to the leg links.

Table 4.3: New and old leg link dimensions for the simulated model.

Model	Shoulder Link	Elbow Link	Wrist Link
Old Model	0.04 m	0.11 m	0.125 m
New Model	0.04 m	0.07 m	0.11 m

The changes to the dimensions of the elbow and wrist links (displayed in Table 4.3) proved extremely effective. Note in Figure 4.6 Rear Right leg torque evolution.

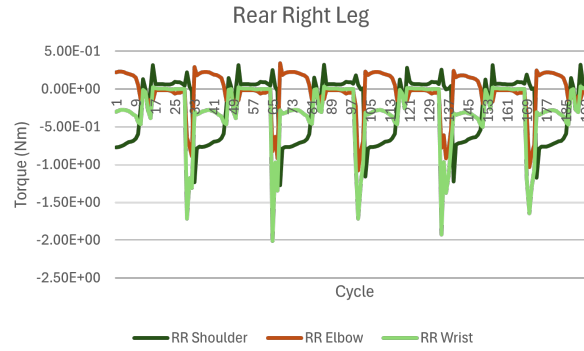


Figure 4.6: Rear Right leg torque evolution of a SHGO optimized model with shortened links.

After analyzing the Rear Right leg torque graph (Figure 4.6), it is evident that the overall torque on the front elbows and rear wrists can be significantly reduced by shortening the leg links. Table 4.4 provides a clear representation of this observation.

Table 4.4: Max and min torque peaks for an optimized model with shortened legs.

Servo	Max (N.m)	Min (N.m)
FL Shoulder	1.12	-0.26
FL Elbow	2.32	-0.43
FL Wrist	0.03	-1.03
FR Shoulder	0.24	-0.99
FR Elbow	2.44	-0.35
FR Wrist	0.03	-0.98
RL Shoulder	1.35	-0.34
RL Elbow	0.34	-1.23
RL Wrist	0.04	-2.32
RR Shoulder	0.34	-1.40
RR Elbow	0.47	-1.13
RR Wrist	0.04	-2.32

Analyzing Table 4.4, it is noticeable there is no absolute torque peak greater than the maximum torque the ST3215 servos are capable of exerting. This means that using this model, the optimized gait is feasible.

4.5 Further SHGO Optimization

Since no points exceed the maximum threshold established, the minimization function converged at 0 (the minimum possible value). This resulted in the algorithm choosing very conservative values for the decision variables. To counter this phenomenon and further optimize the gait, the objective function was modified to not only include the sum of all peak absolute torque values but also add a penalty when the robot failed to travel a predefined distance.

Another change to reach further optimization was the use of only two decision variables - Step Length and Step Velocity - rather than four. Using optimized values for Clearance Height and Penetration Depth (that didn't fluctuate a lot in previous tests), allowed the decreasing of number of dimensions of the problem. This was important because the SHGO algorithm has been proven to work better with lower-dimension problems [40].

Note the graph in Figure 4.7 describing the comparison in travel distance in the same amount of time between the old and new optimization algorithm resulting gaits.

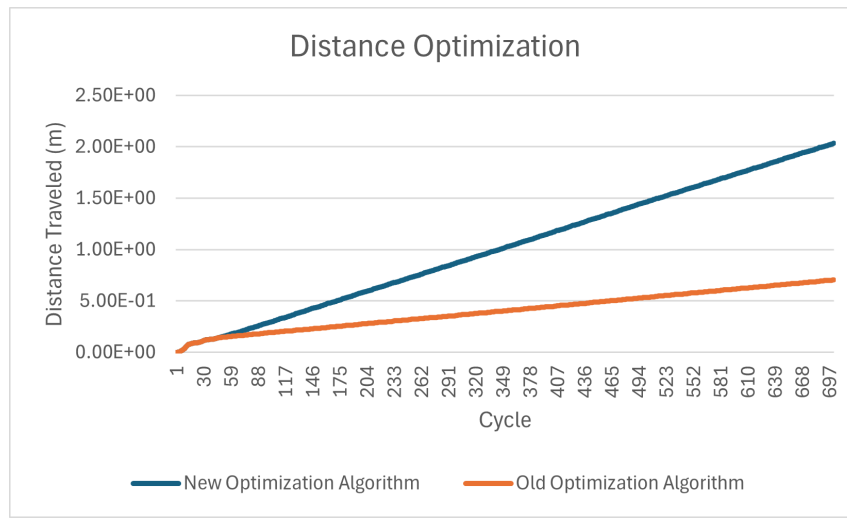


Figure 4.7: Distance traveled comparison between optimization algorithms.

Analyzing the traveled distance graph in Figure 4.7, the difference in velocity between the new and old optimization results is clear. The new optimization algorithm resulting gait is 2.88 times faster than the old optimization algorithm.

Note in Table 4.5 the torque peaks of each servo.

Table 4.5: Max and min torque peaks for distance and torque peak optimization.

Servo	Max (N.m)	Min (N.m)
FL Shoulder	2.07	-0.19
FL Elbow	2.19	-0.56
FL Wrist	0.04	-1.09
FR Shoulder	0.16	-2.10
FR Elbow	2.08	-0.59
FR Wrist	0.04	-1.02
RL Shoulder	2.52	-0.67
RL Elbow	0.60	-0.80
RL Wrist	0.05	-1.11
RR Shoulder	1.06	-2.24
RR Elbow	0.69	-0.86
RR Wrist	0.05	-1.16

After analyzing the data presented in Table 4.5, it can be observed that some of the torque has been distributed across the shoulder joints. This is especially noticeable in the rear shoulder peak values. The more uniform distribution of torque along the joints enables the robot to maintain a faster gait without exceeding the peak torque threshold established by the servo limits.

Note in Figure 4.8 the torque evolution of all legs in the SHGO distance optimized gait configuration for the simulated robot.

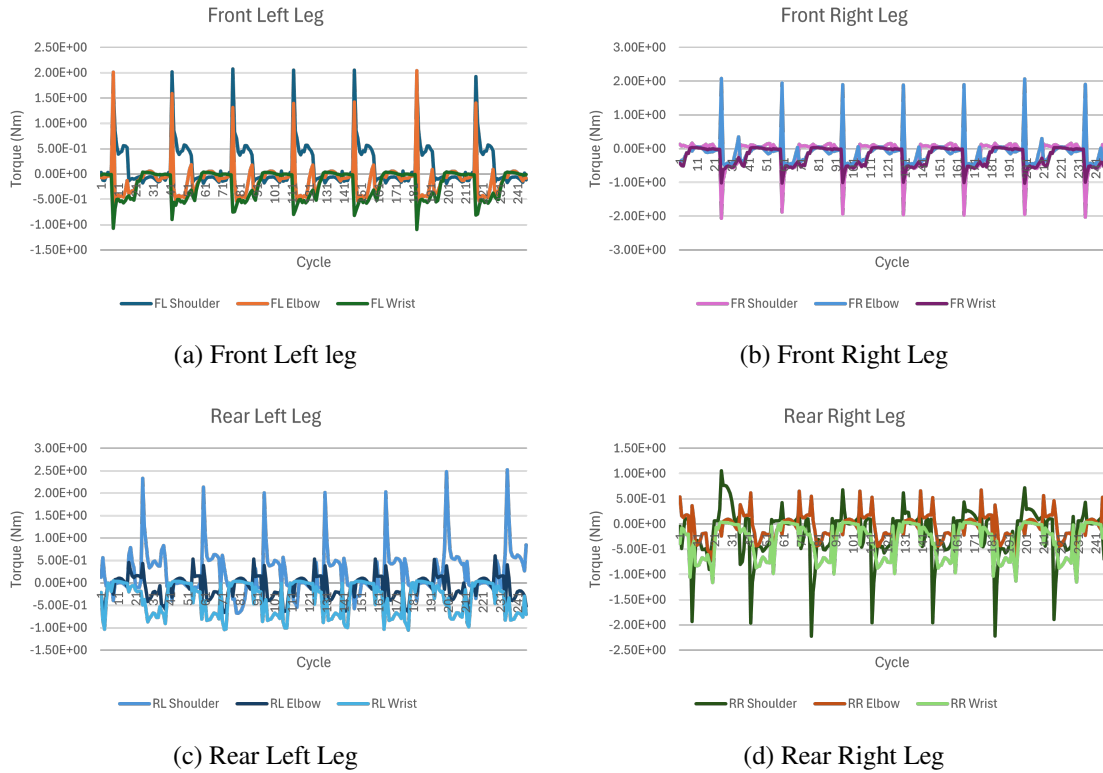


Figure 4.8: Torque evolution of the SHGO distance optimized gait

After analyzing the graphs in Figure 4.8, there is a noticeable difference in comparison to the other optimization processes: There is a clear increase in shoulder torque and a slight decrease in rear wrist and front elbow torque peaks. This means that in order to compensate for the increase in overall torque, the torque was evenly distributed and minimized within the capabilities of the ST3215.

4.6 Multi-Objective Optimization

Although SHGO produced satisfactory results, it did not thoroughly explore the search space. It identified and prioritized solutions that covered greater distances than the original SHGO, yet it still faced challenges in discovering truly optimal solutions, resulting in still very conservative gait parameters. To address this, some Multi-Objective Optimization algorithms were studied and compared.

This section was accepted as a full conference paper in the fourth edition of the Optimization, Learning Algorithms and Applications international conference (OL2A 2024) with the title: "Enhancing Quadruped Robot Performance through Gait Optimization"[14].

Multi-Objective Optimization can help manage trade-offs between multiple objectives in a unified framework, allowing for better exploration of the search space, taking into account competing objectives to achieve more optimal and diverse outcomes.

This section explores the use of Non-Dominated Sorting Genetic Algorithms, namely NSGA-II, NSGA-III, and U-NSGA-III, in conjunction with multi-objective optimization techniques, to optimize the simulated gait. Similar to before, the goal is to minimize overdrive torque while maximizing the distance traveled within a set timeframe, and in the process compare the performance of these NSGA algorithms.

4.6.1 NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm-Version 2) [41] is a widely-used multi-objective genetic optimization algorithm based on Pareto dominance. It features fast non-dominated sorting, efficient crowded distance estimation, and a straightforward crowded comparison operator [41] [42]. It has been reported to outperform PAES and SPEA in terms of solution diversity [43]. NSGA-II has also been successfully applied to optimizing quadruped robot gaits for both single and multi-objective problems [44], though it often struggles with Many-Objective Optimization Problems (MaOP) [45].

The algorithm begins by initializing and evaluating a population, where each solution is evaluated based on its performance with respect to the objectives. Fitness is determined using Pareto dominance, where a solution is "dominated" if another solution is better in at least one objective and not worse in any other.

NSGA-II sorts solutions into different fronts based on Pareto dominance: The first front contains non-dominated solutions, the second front consists of solutions dominated by those in the first front, and so on. In order to enhance diversity within each front, the algorithm uses the concept of Crowding Distance. During selection, a combination of crowding distance and Pareto dominance values is used to select the solutions for reproduction. Crossover and mutation are then applied to these selected solutions to generate new offspring which is then combined with the current population. The next generation is then selected from this pool.

4.6.2 NSGA-III

Motivated by the limitations NSGA-II faced in solving many-objective problems [45] [46], Deb and Jain proposed an update to the algorithm in 2014, known as NSGA-III [47]. While retaining the basic structure of NSGA-II, Version III introduces significant changes in the mating and survival selection processes [48]. Mating selection remains random, but survival selection uses reference directions to represent trade-offs between solutions based on objective values. These

reference directions can be set by the user or generated uniformly using the Das-Dennis method [49].

For an algorithm with M Objectives and a population size N , the survival selection starts by organizing the whole population P_t into non-domination levels. Then, using recombination and mutation, an offspring population Q_t is created from P_t . Combining Q_t and P_t , R_t is formed, as shown in Equation 4.2.

$$R_t = P_t \cup Q_t \quad (4.2)$$

Similarly to NSGA-II, the selection process for the next population involves selecting points one at a time, starting from the first non-dominated front until a complete front cannot be included in the next population. This front is called "Potential Splitting Front", F_L [48]. Then, in order to choose members from the F_L front for P_{t+1} , a niching method is used. This method chooses solutions that best represent the least commonly represented reference directions. This allows for more diverse fronts. A population member that is closest to and unrepresented reference point is immediately preferred [50].

NSGA-II and NSGA-III have been compared regarding many-objective optimization, with NSGA-III generally yielding better results. However, there are instances of mixed outcomes. For example, while NSGA-III outperforms NSGA-II on most DTLZ1-4 test problems, NSGA-II has shown more success on many-objective knapsack problems [51]. Additionally, for bi-objective problems, NSGA-II has been found to perform better than NSGA-III [52].

4.6.3 U-NSGA-III

U-NSGA-III is a unified approach to NSGA-III developed in 2014 [50]. The aim was to improve the performance of NSGA-III in mono and bi-objective optimization problems while maintaining the ability to optimize for 3 or more objectives.

In U-NSGA-III, when the population size, N , exceeds the number of reference points, H , a selection operator is implemented. The variable H represents the anticipated number of optimal solutions after the problem is solved. Given that multiple population members are expected to align with each reference direction, a niching-based tournament selection operator is introduced to manage this association [50].

When selecting two solutions from different reference directions, one is chosen at random. If the solutions originate from the same reference direction, the one with a better non-dominated rank is selected. If both solutions belong to the same niche and non-dominated front, the solution closer to its reference direction is chosen.

In Seada and Deb's proposal, U-NSGA-III displayed a performance comparable to NSGA-II for bi-objective problems and matched the performance of NSGA-III for problems with three or more objectives [50].

4.6.4 Problem Formulation and Evaluation System

In order to minimize torque overdrive and maximize distance traveled, two objective functions were used. The distance traveled evaluation function, $eval_{dist}$, reflected the total distance the robot traveled, $DistTraveled$, during evaluation time. It was defined as the sum of the Euclidean distances traveled during each simulation cycle, i , as described in Equation 4.3.

$$DistTraveled_i = DistTraveled_{i-1} + (Pos_i - Pos_{i-1}) \quad (4.3)$$

To accelerate the search of optimal solutions, if a solution's distance traveled, $DistTraveled$, was lesser than a certain threshold, the evaluation function, $eval_{dist}$, added a punishment to the returned evaluation. It is also important to note that, in order to later compare the NSGA algorithm's regarding hypervolume, it is necessary for both objective functions to be minimization problems, as such, and in order to motivate the algorithm to find solutions that maximized the distance traveled, the evaluation was amplified and inverted. Note in Equation 4.4, the distance evaluation function.

$$\begin{cases} eval_{dist} = (-10) \times DistTraveled_{total} \\ eval_{dist} = 40 \quad \text{if } DistTraveled_{total} < 0.60 \end{cases} \quad (4.4)$$

In the torque overdrive objective function, similarly to the function used in SHGO, the maximum torque, $maxTorque$, recorded out of all the joints during each cycle, i , was checked. If it exceeded the maximum torque capacity of the servo (≈ 3 N.m for the Waveshare ST3215 servo), it was added to a cumulative sum of all other maximum torques that also exceeded the maximum torque, $eval_{Torque}$. After a predetermined number of simulation cycles, the sum was returned as the overdrive torque evaluation for that solution. If the robot detected a failure (such as flipping on its back or falling), the evaluation function would return the worst possible evaluations for both objective functions.

Equation 4.5 presents the torque overdrive evaluation for a simulation cycle i .

$$\begin{cases} eval_{Torque,i} = eval_{Torque,i-1} + maxTorque_i \quad \text{if } maxTorque_i > 3 \\ eval_{Torque,i} = eval_{Torque,i-1} \end{cases} \quad (4.5)$$

4.6.5 NSGA Setup

Similarly to the SHGO setup, the four gait generation parameters were initially used as decision variables using the ranges described in Table 4.6.

Table 4.6: Gait generation parameters

Name	Step Length	Step Velocity	Clearance Height	Penetration Depth
Limit	[-0.100,0.100]	[0.001,3.000]	[0.000,0.100]	[0.000,0.050]

However, as it was previously found (Section 4.5), the Clearance Height and Penetration Depth parameters were found to be redundant and always converged to the same results, while

simultaneously amplifying significantly the complexity of the problem. Taking this into consideration, only Step Length and Step Velocity were used as decision variables.

In order to also make the search space smaller and avoid the exploration of irrelevant (like walking backwards) or doomed solutions, Step Length and Step Velocity were limited to [0.01, 0.1] and [0.1, 0.8], respectively.

The Pymoo [53] library was used to compare the NSGA algorithms due to its customizability and ease for providing graphical visualization.

Crossover and mutation are crucial genetic operators that are responsible for the creation of new solutions from existing ones. These parameters are responsible for ensuring diversity in the population and aiding in exploring the solution space efficiently.

Crossover combines genetic information of two parent solutions to generate new offspring, mimicking the process of biological reproduction, while mutation functions to alter the genes of offspring, diversifying the population and allowing the exploration of promising areas in the search space [54] avoiding local optima.

Table 4.7 describes the mutation and crossover parameters used for each algorithm tested.

Table 4.7: Crossover and Mutation parameters for the NSGA algorithms.

Algorithm	SBX Crossover		Polynomial Mutation	
	Distribution Index	Prob.	Distribution Index	Prob.
NSGA-II	15	0.9	20	1
NSGA-III	30	1	20	1
UNSGA-III	30	1	20	1

Note in Table 4.7, that all algorithms used Simulated Binary Crossover (SBX) [55] which simulates the behavior of single-point crossover (where a single crossover point is selected, and the segments of the parents' chromosomes are swapped) in binary coded genetic algorithms, but for continuous variables, as well as Polynomial Mutation, where small changes are made to the values based on a polynomial distribution.

The crossover distribution index describes the spread of the offspring around the parent solutions, where a higher distribution index results in offsprings that are closer to the parents, encouraging exploitation of the search space, while lower indexes result in its exploration. The mutation distribution index, similarly to crossover, controls the mutations extent. The probability parameters describe the likelihood that crossover will occur between pairs of individuals, or that mutation will occur on a member of the population.

For the NSGA-III and UNSGA-III algorithms, the reference directions were created using the pymoo library [53] and the Das-Dennis method [49], which generates uniformly distributed reference directions and divides the number of divisions along each direction, for this study, the number of divisions was 12.

Due to lengthy evaluation times, a relatively small population size N that was a multiple of 4 was chosen to comply with U-NSGA-III requirements [50] and NSGA-III recommendations [47]. A separate study using NSGA-II was conducted to determine an appropriate population-to-generation ratio. It's important to note that NSGA-II itself does not require a population size to be a multiple of 4, hence some populations in this test may have different sizes.

Table 4.8: Population and Generation ratio testing for NSGA-II.

Population	Generations	Average Distance Traveled of Best Solution(m)
50	30	3.2185
100	15	3.1108
200	7	3.1525

Analyzing Table 4.8, it is clear that larger generation numbers generated a more thorough exploration of the search space. This led to quicker convergence and better final solutions. Considering these results, a population size of 40 (so N is a multiple of 4) was used with a number of total generations set to 30. Each algorithm was then ran 8 times.

4.6.6 Hypervolume Setup

The hypervolume performance indicator is a metric commonly used to evaluate the quality of a set of solutions resultant of multi-objective optimization. This indicator measures how well a set of solutions covers the Pareto front by calculating the volume it occupies in objective space relative to a reference point. It allows us to compare algorithms quantitatively based on their generated Pareto fronts.

The Pymoo library [53] was used for the hypervolume calculations. Since the goal involved minimizing torque overdrive and maximizing distance traveled (both minimization problems can be easier to handle), the distance traveled values were negated (as described in Equation 4.4). This ensured both objectives were minimization problems. The reference point was then chosen to be worse than all solutions captured in NSGA-II, NSGA-III and UNSGA-III with coordinates (280, 1). The hypervolume results were then normalized to fit between 0 and 1.

4.6.7 NSGA Optimization Results

In this sub-section, the results for the NSGA algorithms regarding minimization of overdrive torque and maximization of distance traveled (inverted to become minimization) are presented, and analyzed regarding the quality of their pareto fronts, generations and best solutions.

4.6.7.1 Hypervolume and Pareto Fronts

Figures 4.9, 4.10, and 4.11 illustrate the aggregated Pareto fronts produced by each NSGA algorithm across 8 separate runs. The purpose of combining these Pareto fronts in a single graph is to evaluate the consistency and reliability of the final Pareto fronts generated by each algorithm.

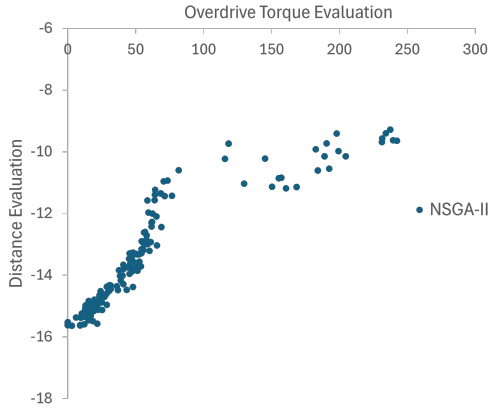


Figure 4.9: Combined NSGA-II Pareto fronts.

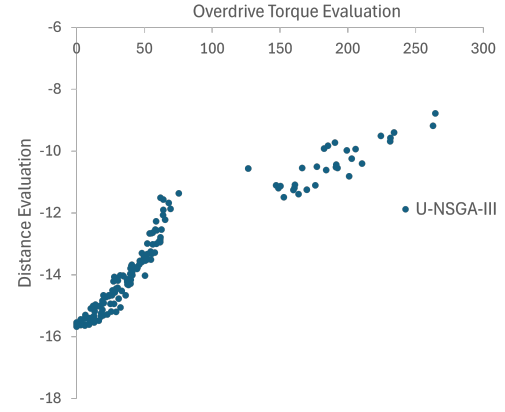


Figure 4.10: Combined U-NSGA-III Pareto fronts.

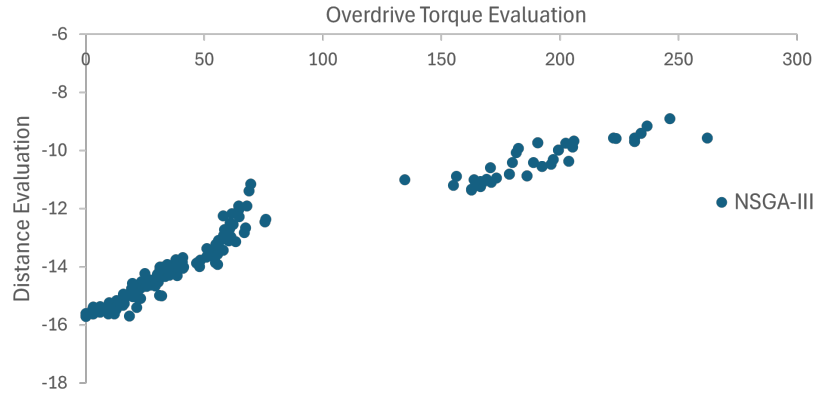


Figure 4.11: Combined NSGA-III Pareto fronts.

The graphs reveal that while all three algorithms achieve similar Pareto fronts, NSGA-III stands out by producing more compact and tightly clustered results, particularly for higher overdrive torque solutions. In contrast, both NSGA-II and U-NSGA-III display more variability in their Pareto fronts. This tighter clustering seen with NSGA-III suggests a superior convergence towards the true Pareto front. Moreover, the convergence of the Pareto fronts towards optimal solutions implies that there are regions where improvements in one objective can be realized without substantial sacrifices to the other objectives.

Building upon these observations, the study proceeds to analyze the hypervolume values associated with the Pareto fronts generated by each algorithm. Table 4.9 presents the best, worst, and average hypervolume values for each algorithm, with the best results in each category highlighted in bold.

Table 4.9: Best, worst and average hypervolume final results for the NSGA algorithms.

	Worst	Average	Best
NSGA-II	0.9590	0.9620	0.9631
NSGA-III	0.9602	0.9628	0.9656
U-NSGA-III	0.9598	0.9618	0.9645

The results in Table 4.9 demonstrate that, regarding hypervolume evaluations, all three algorithms generate solutions of similar quality. Nevertheless, NSGA-III consistently achieves the best results in every category, indicating that its solutions are more optimally distributed across the objective space. Figures 4.12, 4.13, and 4.14 depict the average hypervolume values for each algorithm over successive generations.

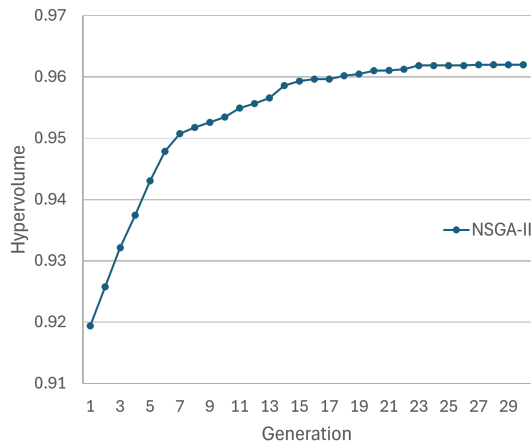


Figure 4.12: Average hypervolume evolution for NSGA-II.

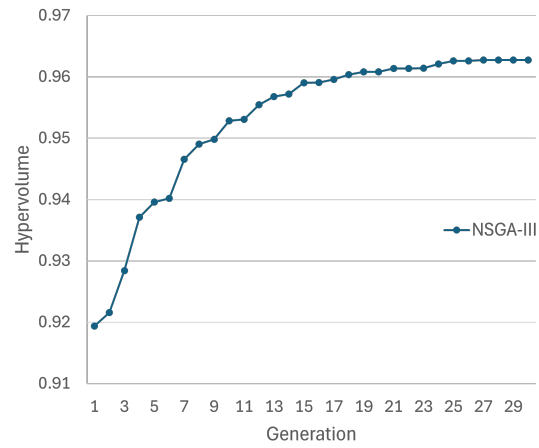


Figure 4.13: Average hypervolume evolution for NSGA-III.

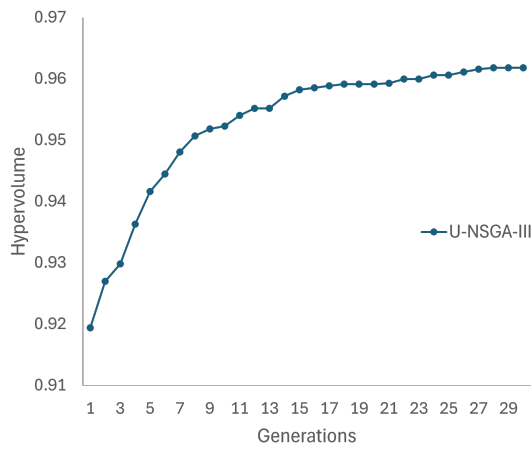


Figure 4.14: Average hypervolume evolution for U-NSGA-III

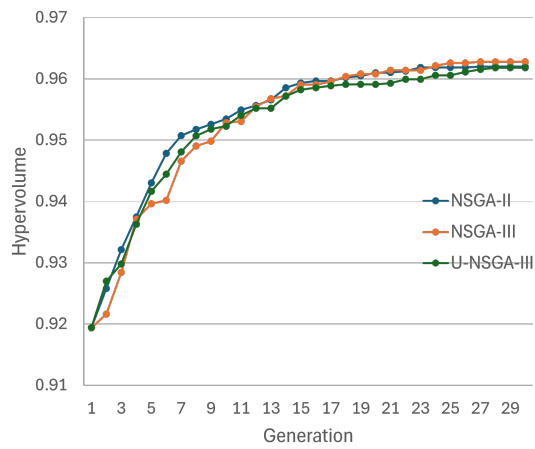


Figure 4.15: Average hypervolume evolution for the three algorithms.

The graphs illustrate that the hypervolumes follow a relatively similar trajectory, with NSGA-II and U-NSGA-III initially outperforming NSGA-III until the 7th generation, after which NSGA-III quickly matches and exceeds their performance. These graphs confirm the findings presented in Table 4.9 and Figure 4.11, showing that while NSGA-III has the superior solution set, the advantage is not substantial.

4.6.7.2 Best Solution

Table 4.10 presents a comparative analysis of the best evaluation results obtained over 8 runs. Like Table 4.9, it highlights the best, worst, and average best evaluations achieved by each algorithm.

Table 4.10: Best, worst and average best solutions for the NSGA algorithms.

	Worst	Average	Best
NSGA-II	-15.5239	-15.5931	-15.6234
NSGA-III	-15.5286	-15.6071	-15.7000
U-NSGA-III	-15.4614	-15.5705	-15.6713

Table 4.10 shows that NSGA-III consistently outperformed both NSGA-II and U-NSGA-III, delivering superior average solutions, the best overall results, and the least undesirable outcomes. While NSGA-II had the second-best average, U-NSGA-III achieved a better overall result.

These results indicate that, although all three algorithms are appropriate for the problem, NSGA-III stands out by producing a better set of solutions as well as superior individual solutions.

4.6.7.3 Null-Overdrive Torque Results

If overdrive torque represents the torque peaks exceeding the maximum capability of the servos in the leg joints, then truly optimal solutions must exhibit zero overdrive torque. To explore this, by considering only the null-torque results from the function evaluations, one can isolate and visualize the progression of these zero-torque evaluations across generations.

To achieve this, all solutions with non-zero torque overdrive were removed from the data set. Subsequently, the median of the function values for each generation was calculated and averaged over 8 runs. Figure 4.16 presents the results of this analysis.

Analyzing the graph reveals that the development of null overdrive torque solutions follows a similar path. While NSGA-III initially progresses at a slower pace compared to NSGA-II and U-NSGA-III, it eventually surpasses them in later generations, similarly to the hypervolume evolution depicted in Figure 4.15.

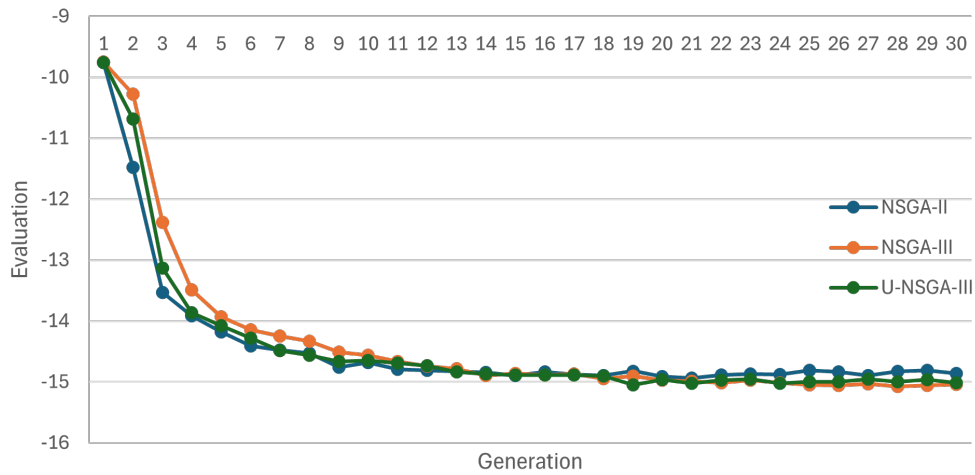


Figure 4.16: Median function values for the null overdrive torque solutions for each algorithm - Averaged over 8 runs.

Table 4.11 highlights the most significant findings from these median values. It does so by singling out the best-performing median fitness value for each run of each algorithm, and then averaging these best median fitness values for each algorithm.

Table 4.11: Best, worst, and average median evaluation for the null torque overdrive solutions.

	Worst	Average	Best
NSGA-II	-14.9489	-15.0438	-15.1345
NSGA-III	-14.9489	-15.1428	-15.1982
U-NSGA-III	-14.8785	-15.0744	-15.1954

Upon examination of Table 4.11, the findings reaffirm what other performance indicators have indicated: while all three algorithms demonstrate competency for this problem, NSGA-III consistently outperforms the others, leading to superior results.

4.6.7.4 Conclusion

During the analysis of these results, it became evident that all three algorithms yielded impressive outcomes and demonstrated suitability for this problem. Nevertheless, upon thorough evaluation and comparison of their performance, NSGA-III emerged as the superior option, consistently delivering better results across all metrics.

NSGA-III displayed superior performance across all metrics. It generated Pareto fronts with reduced variation across multiple runs, particularly in higher overdrive torque regions. Additionally, it produced better hypervolume results, indicating a more diverse and optimal array of solutions. NSGA-III consistently yielded the best evaluated solutions and also showed better results in the null overdrive torque section of the search space. These outcomes display NSGA-III's capability to outshine both NSGA-II and U-NSGA-III in this bi-objective optimization scenario.

4.7 Final Optimization using NSGA-III

During the final stages of the simulator analysis, it became apparent that comparing the optimized results with the performance of a real robot could offer valuable insights. However, after extensive testing, it was found that when the Step Length and Step Velocity parameters exceed certain thresholds, the simulator's behavior increasingly diverged from that of the real robot. This discrepancy could be a result of real-world variables which are not fully captured by the simulator.

To address this issue and preserve the integrity of the simulator study findings, an additional run of NSGA-III, incorporating constraints that align with the robot's safe operational thresholds. Given that NSGA-III demonstrated superior performance in the previous section, it was the natural choice for this adjusted optimization run. The new run used the Step Length and Step Velocity limits $[0.01, 0.04]$ and $[0.1, 0.5]$, respectively, and resulted in a Step Length of 0.039468 and a Step Velocity of 0.30264.

Note in Figures 4.17, 4.18, 4.19 and 4.20 the torque evolution of all legs in the final optimized gait configuration for the simulated robot.

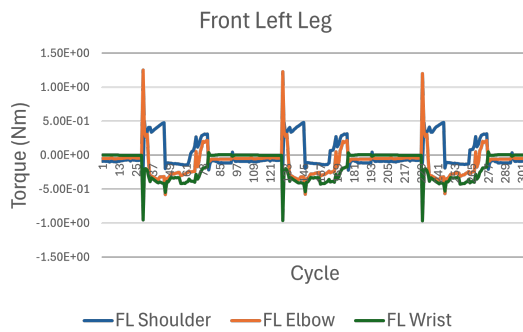


Figure 4.17: Front Left Leg.

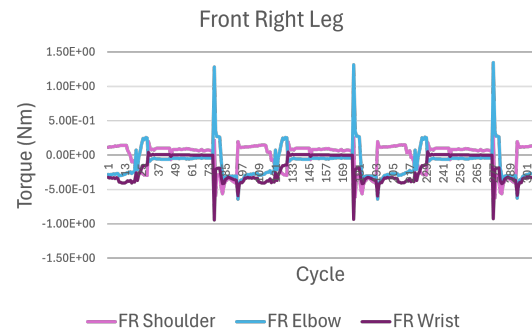


Figure 4.18: Front Right Leg.

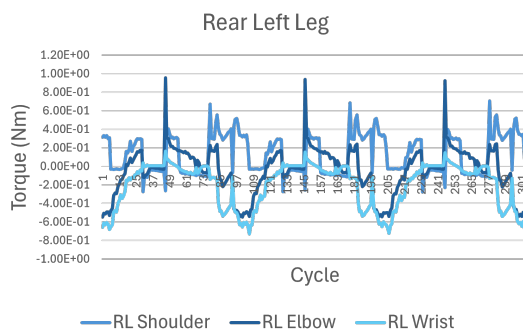


Figure 4.19: Rear Left Leg

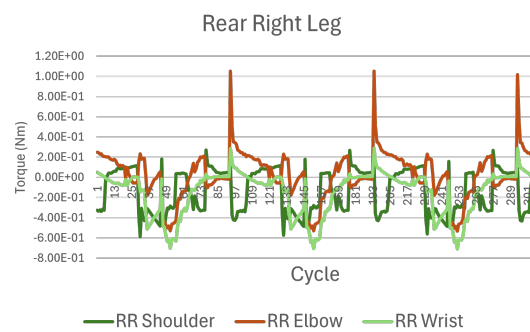


Figure 4.20: Rear Right Leg.

Figure 4.21: Leg torque evolution for the NSGA-III optimal gait solution.

Chapter 5

Robot Prototype

This chapter describes how the robot operates, detailing its software structure, communication and control algorithms. This chapter is divided into five Sections. First, the process of gait generation is outlined in Section 5.1, followed by Section 5.2 where the communication between the PC and microcontroller is described. In Section 5.3, a class diagram is used to explain the microcontroller's software structure. Subsequently, in Section 5.4 the gait execution loop is presented. Finally, the graphical user interface (GUI) used to command the robot is explained in Section 5.5.

5.1 Gait Generation

The gait is generated following a logic similar to that of the simulation (as described in Chapter 4, Section 4.2), except that the logic of the environment is omitted. Since the robot is not simulated, rendering and updating the environment would be a waste of resources and computational time.

The gait generation algorithm can be explained through the following steps, which are illustrated in an activity diagram in Figure 5.1.

1. The time-step in the gait is updated to accurately determine the current (trajectory) phase and in what stage (Stance or Swing) of the gait the leg is in.
2. If the leg is in the swinging stage of the gait, a point in the Bezier curve that matches the gait phase is calculated. This point is then converted into foot coordinates using the base center as the reference point. If the leg is in the stance period of the gait (meaning that it is in contact with the ground), the desired foot trajectory is calculated using an adjustable sinusoidal wave that takes into account Step Length and Lateral Fraction.
3. The foot coordinates are then converted to be relative to the hip reference frame of the respective leg.
4. The desired foot coordinates are then inputted into the inverse kinematic model (Described in Chapter 3, Section 3.4) in order to calculate the required joint angles to achieve the desired end-effector position.

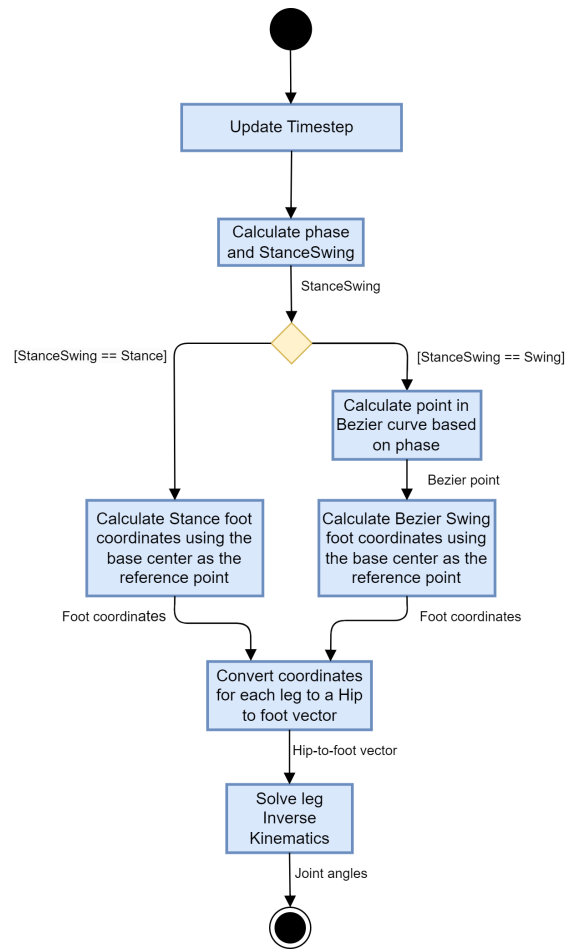


Figure 5.1: Gait Generation activity diagram.

5.2 Communication

The system was split into two sub-systems: the PC and the ESP32. The PC handled calculating and loading gaits and issued commands to the robot, while the ESP32 stored and executed the gaits, controlled the servos, and received feedback.

In any networked application, the way systems communicate is crucial for ensuring data is transferred accurately and reliably. For this study, TCP (Transmission Control Protocol) was used to handle the communication between the PC and the ESP32.

5.2.1 Data Format

The data is transferred in strings, which are encoded in UTF-8 (Unicode Transformation Format - 8-bit), a variable-width character encoder that is compatible with ASCII, which is then turned into a character array in the client side.

In order to effectively explain the format of the messages sent from the server to the client, they were categorized into two types: Commands and Loaders.

5.2.1.1 Command Messages

Command Messages function mostly as flags: the server sends a single character to the ESP32 that is acknowledged and interpreted. These messages serve the purpose of instructing the robot to perform a specific action or execute a predefined command. Table 5.1 displays the possible command messages:

Table 5.1: Command message list.

Command Name	Command Message	Command Description
Forward	'1'	Execute loaded Forward gait
Backward	'2'	Execute loaded Backward gait
Right	'3'	Execute loaded Right gait
Left	'4'	Execute loaded Left gait
Rotate Right	'5'	Execute loaded Right Rotation gait
Rotate Left	'6'	Execute loaded Left Rotation gait
Custom Gait	'7'	Execute Custom gait loaded from the GUI
STOP	'8'	STOP all movement, return to Prone Stance and Shutdown
Prone Stance	'P'	Set robot in Prone Stance
Calibration Stance	'C'	Set robot in Calibration Stance

5.2.1.2 Load Messages

Load messages are responsible for sending the gaits calculated in the PC into the ESP32, to do so, the gait is divided into 4 messages, where each message sends the positions in the gait for the joints in one leg.

The message is structured using a comma-separated format, where each piece of information is delineated by comma (','), this allows for easy parsing and understanding of the message content. The messages are divided into 3 sections.

1. **Gait Type Indicator (GTI):** The segment before the first comma indicates what type of gait is being loaded (Forward, Backward, Left, Right, Rotate Left, Rotate Right, or Custom).
2. **Leg Indicator (LI):** The next segment indicates the the leg that is being loaded (Front Left, Front Right, Rear Left, or Rear Right).
3. **Joint Positions:** The remaining segments are grouped into three sets, each corresponding to the positions of each joint. Each group contains a series of numbers separated by commas, detailing the positions for each joint.

Due to the limited buffer size, it's crucial to optimize resource usage. To achieve this, a method was implemented where an "s" is sent in a message whenever a joint angle remains unchanged

Analyzing Figure 5.3, it is clear that, for each leg message, there are three stages. 1) The data is sent from the PC to the ESP32, 2) The ESP32 receives this message, processes it, and then acknowledges it and informs the server of the leg that was processed. 3) The server confirms that it was the correct leg. This process is repeated until the leg gait messages have been processed, and there is no more information to be received. The client then sends a message with the flag corresponding to the GTI, and the server now knows that the gait is completely saved in the ESP32.

If a message does not get an acknowledgment on the server side, like the situation described in Figure 5.3, a timeout is detected and the unacknowledged message is sent again.

5.3 Microcontroller Software Structure

The microcontroller's software is structured around two main classes: SpotServo and Spot, which interact with the enumerations **LegType** and **JointType**. Figure 5.4, describes the Class Diagram of the code running in the ESP32 followed by a brief explanation of the key attributes and methods for the Spot and SpotServo classes.

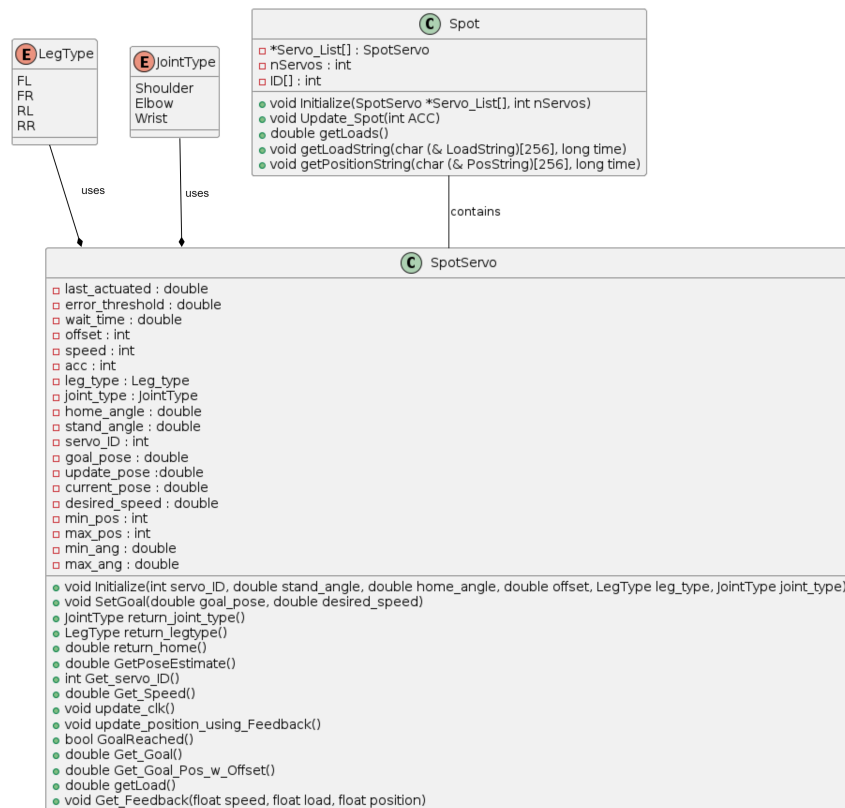


Figure 5.4: ESP32 Class Diagram

5.3.1 SpotServo

The SpotServo class is responsible for managing individual servo motors, providing methods for initialization, goal setting, feedback and general control. Key attributes include **servo_ID**, a unique identifier necessary for executing any commands. Attributes like **leg_type** and **joint_type** specify the exact location within the robot's structure where the servo operates. During Initialization, this allows the establishing of the servos operational boundaries (**min_pos**, **max_pos**, **min_ang**, and **max_ang**) taking into account what role the servo is serving.

To manage servo movement, the **SetGoal()** method allows the setting up of the **goal_pose** and **desired_speed** which dictate where and how quickly the servo should move when commanded to do so. The **GoalReached()** method returns a boolean value indicating whether the servo's **current_pose** aligns within an acceptable range defined by the **error_threshold** around the **goal_pose**.

In order to track the servo's state, **current_pose** is updated using **update_position_using_feedback()**, reflecting the current angle of the servo. **Get_Feedback()** also allows to get servo feed back for load, speed and position values.

5.3.2 Spot

The spot class serves as a representation of the robot, structured around an array of SpotServos. At its core, it includes the **Servo_List[]** which houses the SpotServos of the robot, allowing for managing and interacting with the individual servos. The initialization of the class ensures that the servos within **Servo_list[]** are configured and ready for operation.

The most important method of this class is by far the **Update_Spot()** method which commands all servos within the **Servo_list[]** to move to their respective **goal_pose** at the pre-defined **desired_speed**. This allows for synchronized movements which is crucial for achieving coordinated actions.

The other methods (**getLoads()**, **getLoadString()**, and **getPositionString()**) are used for feedback control of the robot's servos. This data allows for real-time monitoring of the operational parameters as well as debugging and saving results.

5.4 Gait Execution

When a gait execution command (described in Table 5.1) is received, the robot interprets the requested gait and proceeds with executing it if the gait is loaded. The gait then follows the loop defined in Figure 5.5, where each cycle is one of the positions in the loaded gait.

The loop begins by retrieving the joint angles corresponding to the position intended for execution in cycle i . Subsequently, the desired speeds for each servo are computed to ensure simultaneous completion of movement for all servos within the same leg. T_{max_i} , which represents the maximum time required by the servos, is also stored for future reference.

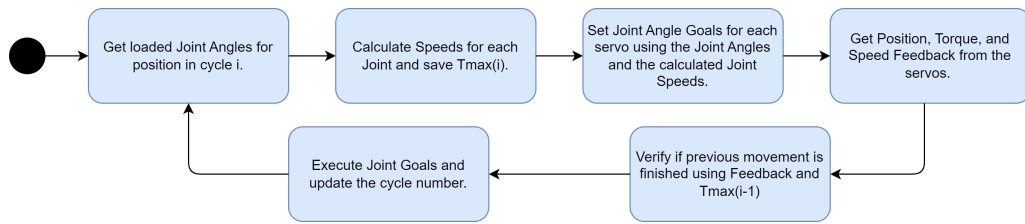


Figure 5.5: Gait Execution loop diagram.

The goals for each servo are then established using the method **SetGoal()** (previously mentioned in Section 5.3, Sub-Section 5.3.1), along with the computed speeds. This ensures that everything is prepared for the execution of the movement when the time comes.

It is at this point in the loop that Position, Torque and Speed feedback is pulled from the servo, which is used to verify if the previous movement has been completed (along with $Tmax_{i-1}$) and result retrieval.

Following the verification, the servos execute the previously set goals and update the cycle number. It is very important to note that if at any point during the cycle a new message is received from the server, the loop is broken and the newly received command is executed.

5.5 Graphic User Interface (GUI)

A Graphical User Interface (GUI) offers an intuitive means for users to interact with the robot, providing easy configuration and execution of commands. Leveraging the Tkinter Python library [56], an interface was developed that allows users to command the robot from a PC. Additionally, users can send custom gaits for the robot to load and execute.

Note in Figure 5.6 the GUI that was used to control the robot.

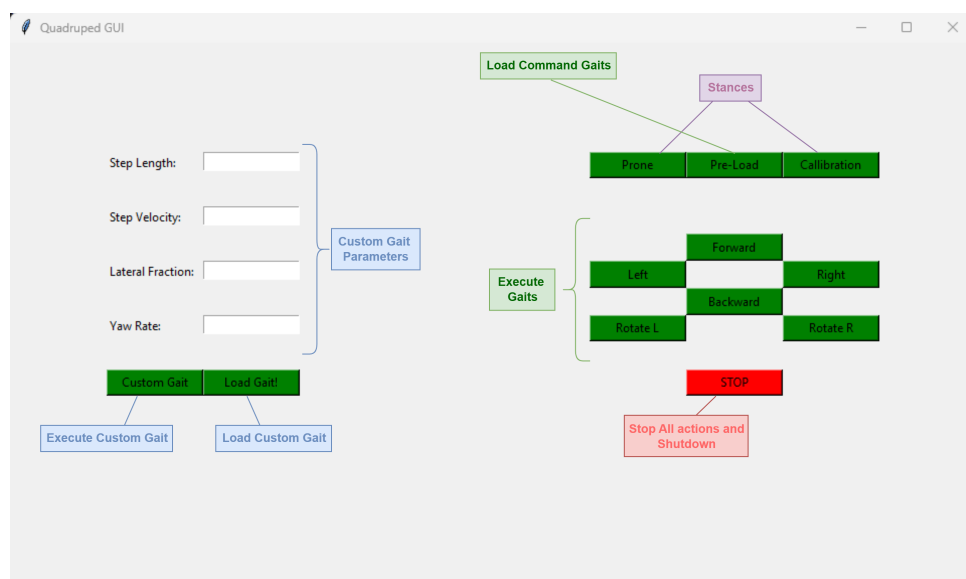


Figure 5.6: Graphic User Interface (GUI).

As described in Figure 5.6, the GUI has several key features:

- Users can input values for Step Length, Step Velocity, Lateral Fraction and Yaw Rate to define a new custom gait, which the user can load and execute to verify its performance.
- The "Prone" and "Calibration" buttons allow users to position the robot in these specific stances.
- Users can also control the robot using pre-loaded gaits in all directions, including rotations.
- The emergency STOP button can be used at all times, which stops all actions and shuts down the robot to ensure safety.

Chapter 6

Tests and Results

In this chapter, the tests used to validate the robot are explained along with their respective results. This chapter is divided into three Sections. First, Section 6.1, describes the gait accuracy test, where the simulated and real positions are compared after the same gait is executed on both systems. Subsequently, in Section 6.2 a filter to mitigate the error from the servo's feedback load is developed through a dynamic and a static test, which is then used to compare real and simulated torque peak results. Finally, Section 6.3 presents the tests used to validate the traveled distances after executing the optimized gait.

6.1 Gait Accuracy

To evaluate the system's gait accuracy, a gait was performed and the joint angles were recorded. This same gait was then replicated in the simulator, and the results were compared. The gait used was the final NSGA-III optimized gait from Sub-Section 4.7, specifically a forward gait with a Step Length of 0.039468 and a Step Velocity of 0.30264. Figure 6.1 illustrates the results for the Front Left leg.

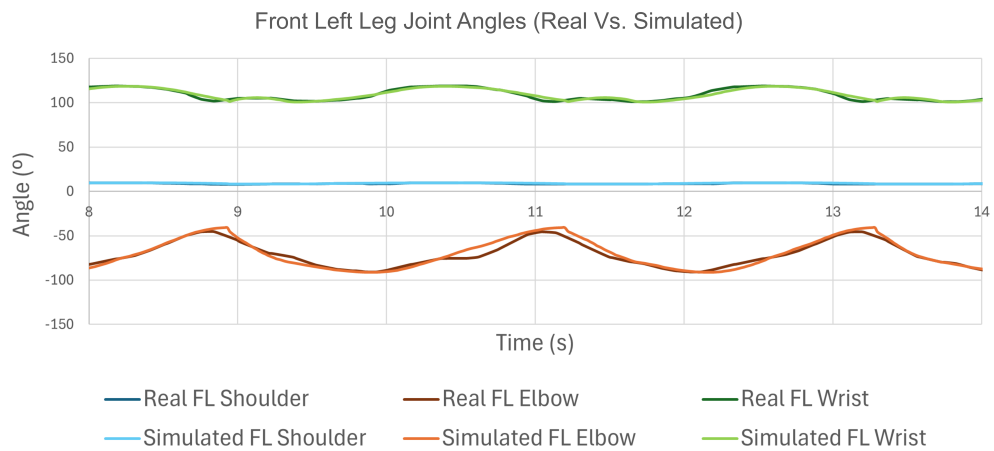


Figure 6.1: Front Left leg joint angles comparison for the simulated and real robots.

Analyzing Figure 6.1, it is clear that the real results follow the same curve as the simulated angles. This shows that the robot is following the commands correctly and achieving the expected positions. Notably, the simulated curve is more defined and there are some discrepancies due to the more frequent sampling of positions compared to the real robot.

To provide a clearer understanding of the differences between the simulated and real results, each joint will be individually analyzed, starting by the Shoulder joint in Figure 6.2.

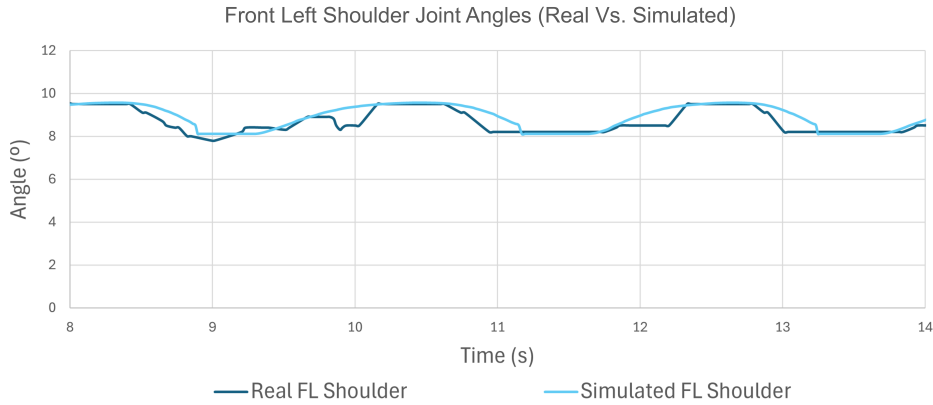


Figure 6.2: Real and Simulated Front Left Shoulder joint angle results.

Observing Figure 6.2, there is a noticeable difference between the curves as consequence of the shoulder commands being issued in small increments (decimals) varying between 8 and 9.5. Due to the buffer size, the loaded gaits are received as integer numbers (as shown in Figure 5.2 in Chapter 5), this means that the servo only processes the commands at a larger resolution. Adding to this, the joints also consider that they have reached their goal when they are within 0.1° of their target. Figure 6.3 describes the error in percentage between the Simulated and Real Shoulder joint angles.

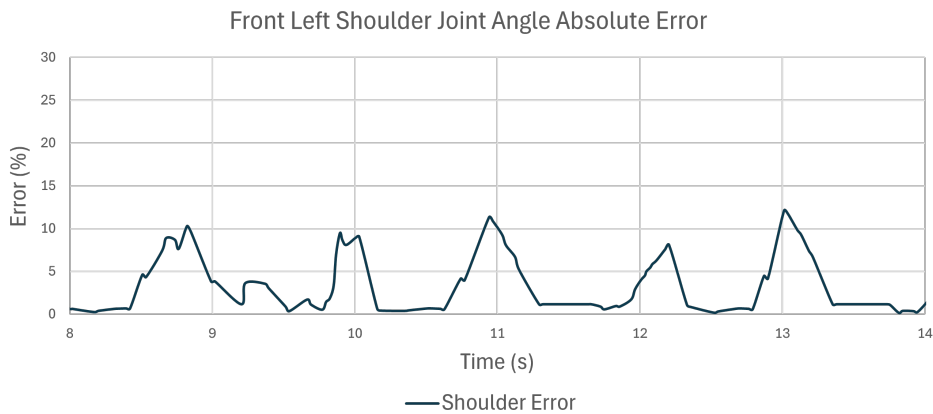


Figure 6.3: Shoulder joint percentage error.

The percentage error plot (Figure 6.3) confirms the analysis of Figure 6.2, where the error fluctuates between 0 and 12%, caused by the previously mentioned limitations. Despite these differences, the overall trends in the curves are aligned, indicating that the robot is following the intended movements. Figure 6.4 illustrates the Elbow joint angle results.

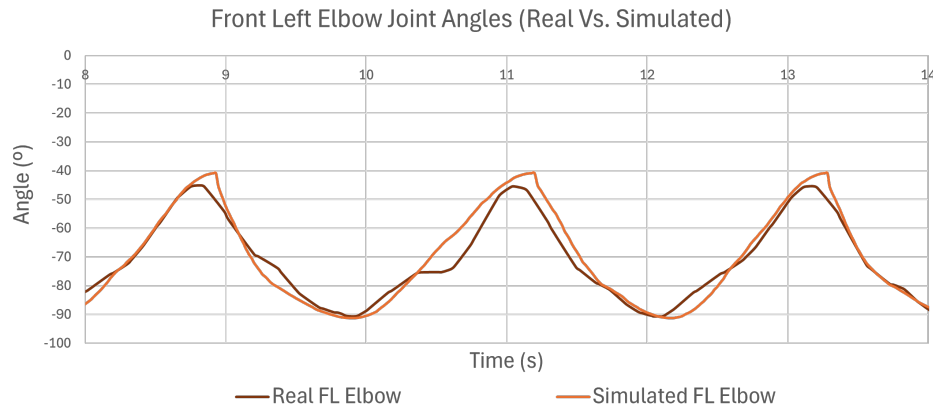


Figure 6.4: Real and Simulated Front Left Elbow joint angle results

Analyzing Figure 6.4, it is possible to observe some differences between the real and simulated angles. These discrepancies result from differences in sampling rates and minor misalignment of the curves. However, these differences do not significantly affect the overall alignment. Additionally, the simulated curve exhibits sharper transitions compared to the real curve, due to the higher resolution of the simulated data, which captures more detailed changes in joint angles than the real-world data. Figure 6.5 illustrates the percentage error between the data.

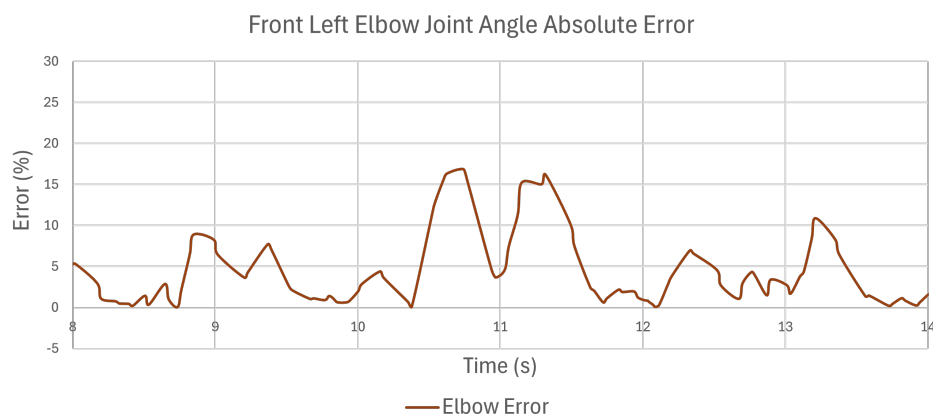


Figure 6.5: Elbow joint real percentage error.

Observing Figure 6.5, it is clear that, there are two noticeable peaks around 10.5s and 11.5s. These peaks indicate specific instances of higher discrepancy, but they are not cyclical, suggesting

that they are due to occasional anomalies rather than recurring issues in the simulation or measurement. For the majority of the time, the error fluctuates around 0-10%, indicating that the overall discrepancy is generally low (Table 6.1). Note in Figure 6.6 the Wrist joint results.

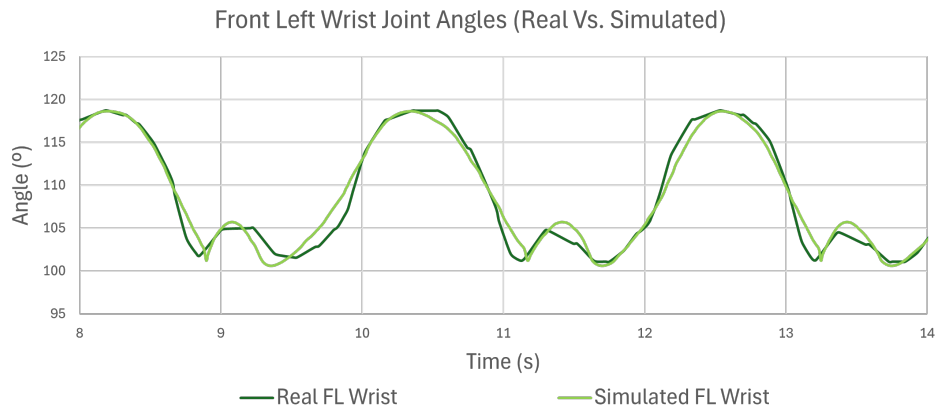


Figure 6.6: Real and Simulated Front Left Wrist joint angle results

The wrist joint angle results shown in Figure 6.6 again demonstrate expected minor deviations. Figure 6.7 further emphasizes this by plotting the absolute percentage error.

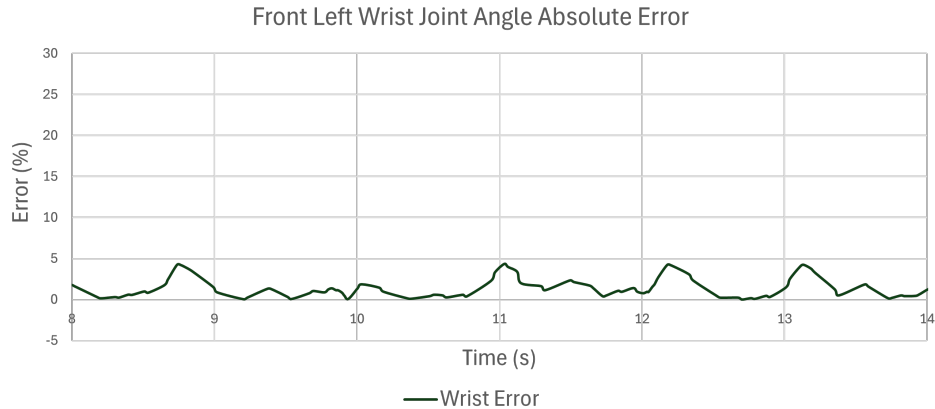


Figure 6.7: Wrist joint real percentage error.

Figure 6.7 illustrates that the Wrist joint exhibits notably lower percentage error compared to both the Elbow and Shoulder joints, indicating excellent performance in replicating intended movements. Table 6.1 describes the minimum, average, and maximum absolute error results for each joint.

Overall, even though the shoulder and elbow joints show higher maximum and average percentage errors, these differences are not very significant due to their relatively minimal average errors. The average percentage error for the Wrist joints being around 1.39% is quite good, indicating accurate replication of intended movements.

Table 6.1: Max, min and average absolute error results.

Joint Name	Min (%)	Average (%)	Max (%)
FL Shoulder	0.12	3.64	12.08
FL Elbow	0.04	3.95	16.85
FL Wrist	0.001	1.39	4.35

It's important to note (in Table 6.1 that the percentage errors for the shoulder and elbow joints appear larger because their actual angle values are smaller. Therefore, the same degree of error results in a higher percentage error for these joints.

6.2 Leg Torque

When examining the feedback related to the load on the servo, it became evident that there was a notable error associated with these values. To resolve this issue, a filter was created to remove the relevant error. The process of setting up this formula was carried out for the Front Left Elbow in the following steps.

6.2.1 Servo Load Feedback Calibration

Initially, it was observed that the Load output by the servos in the feedback, denoted as L , was equal to the total motor torque, T_m , multiplied by a scaling factor, ϵ , as described in Equation 6.1.

$$L = \epsilon T_m \quad (6.1)$$

To account for and remove the torque associated with factors such as friction, B , which opposes the motion of the servo, as well as any disturbances or uncertainties, T_q , the following Equation 6.3 was used to calculate the servo torque, T_S , where ω is the angular speed:

$$T_S = T_m - (B\omega + T_q) \quad (6.2)$$

$$\Leftrightarrow T_S = \frac{L}{\epsilon} - (B\omega + T_q) \quad (6.3)$$

To find the necessary components that describe this equation, two test had to be conducted.

6.2.2 Null Acceleration Test

In order to calculate B and T_q , some tests with null acceleration were conducted. It is important to first understand why null angular acceleration important. Note Equation 6.4, which describes the relationship between the Inertial Momentum, I , angular acceleration, α , and servo torque, T_S .

$$I\alpha = T_S \Leftrightarrow I\alpha = T_m - (B\omega + T_q) \quad (6.4)$$

This means that if the servo is at a constant velocity (null acceleration), it is possible to achieve Equation 6.6

$$0 = T_m - (B\omega + T_q) \Leftrightarrow T_m = B\omega + T_q \quad (6.5)$$

$$\Leftrightarrow L = \varepsilon(B\omega + T_q) \Leftrightarrow L_{null} = \varepsilon(B\omega + T_q) \quad (6.6)$$

To determine these parameters, the servos were spun at various constant speeds over an extended duration to eliminate acceleration, and the corresponding Load values (in kg·cm) were recorded. Note in Figure 6.8 the graph that illustrates these results.

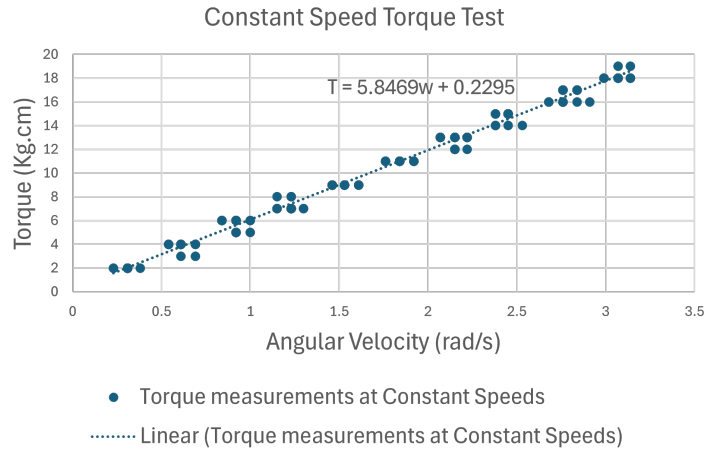


Figure 6.8: Constant Speed Torque test for the Front Left Elbow servo.

Analyzing Figure 6.8, it is clear that the trend line defines Equation 6.6, where $\varepsilon B = 5.8469$ and $\varepsilon T_q = 0.2295$.

6.2.3 Acceleration Test

From Equation 6.4, it is possible to calculate ε , if the angular acceleration, α , and moment of inertia, I , is known. Note Equation 6.8, which describes how this was achieved.

$$I\alpha = T_m - (B\omega + T_q) \Leftrightarrow \varepsilon I\alpha = L - \varepsilon(B\omega + T_q) \quad (6.7)$$

$$\Leftrightarrow \varepsilon = \frac{L - L_{null}}{I\alpha} \quad (6.8)$$

However, to use Equation 6.8, the moment of inertia is needed, along with the angular acceleration. To obtain these, a uniformly weighted disk was spun using the servo as illustrated in Figure 6.9.

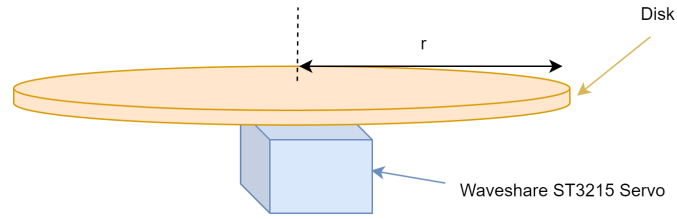


Figure 6.9: Illustration of the dynamic test setup.

The disk described in Figure 6.9 had mass, $m_{disk} = 0.268kg$, and radius, $r = 0.15m$, Using the moment of inertia of a disk formula [57] (described in Equation 6.9), it was possible to define the moment of inertia, I :

$$I = \frac{m_{disk} \times r^2}{2} \Rightarrow I = 3.015 \times 10^{-3} (kg.m^2) \quad (6.9)$$

To calculate the angular acceleration, α , the disk was spun until it reached a target angular velocity. the velocity was sampled at intervals during the acceleration process, and subsequently plotted in Figure 6.10.

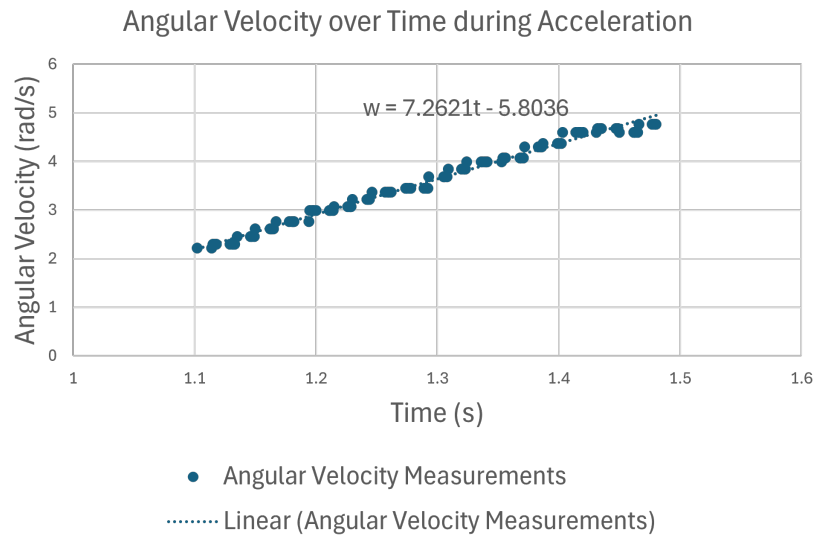


Figure 6.10: Illustration of the dynamic test setup.

The trend line in Figure 6.10 illustrates the evolution of angular velocity over time, indicating that $\alpha = 7.2621 \text{ rad/s}^2$. Developing Equation 6.8, it is possible to find ϵ , as shown in Equation 6.10.

$$\epsilon = \frac{L - (5.8469\omega + 0.2295)}{7.2621 \times 3.015 \times 10^{-3}} \quad (6.10)$$

Plotting ϵ results in the following Figure 6.11

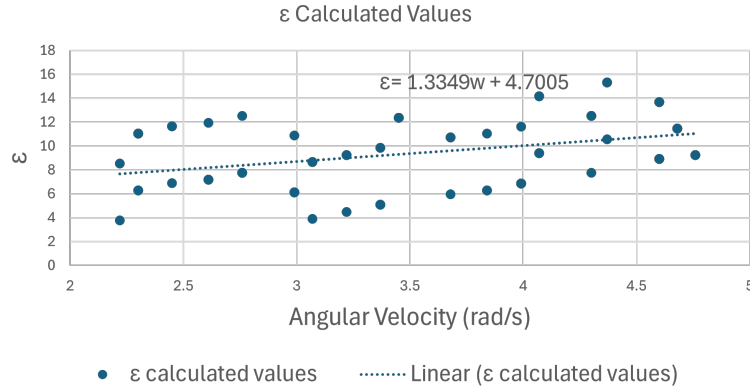


Figure 6.11: ε Calculated values for the Front Left Elbow servo.

Using the average of the calculated ε values, the ε value used was $\varepsilon = 9.54$. The servo torque error mitigation formula is illustrated in Equation 6.11.

$$T_S = \frac{L - \varepsilon(B\omega + T_q)}{\varepsilon} \Leftrightarrow T_S = \frac{L - (5.847\omega + 0.230)}{9.54} \quad (6.11)$$

6.2.4 Resulting Correction

To understand the impact of the torque evaluator function, consider an example of the speed evolution (in Figure 6.12) from one of the tests conducted with the setup shown in Figure 6.9.

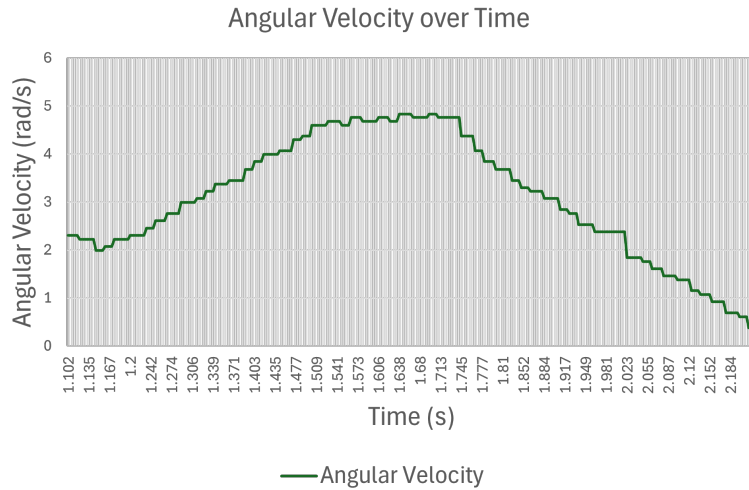


Figure 6.12: Angular velocity for formula example.

Analyzing Figure 6.12 it is clear that initially, the speed evolves with constant acceleration subsequently decelerating. This indicates that the torque evolution should exhibit two constant torque values at these respective phases. Using Equation 6.4 it was calculated that the first section should have $\approx 0.22kg.cm$ and $\approx -0.23kg.cm$ for the decelerating section. Figure 6.13 and 6.14.

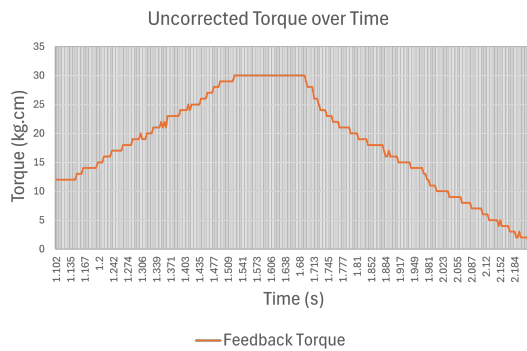


Figure 6.13: Uncorrected Feedback Torque.

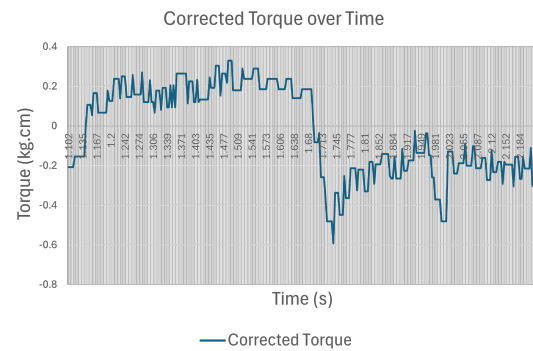


Figure 6.14: Corrected Feedback Torque.

Analyzing Figure 6.13, it is clearly following the same progression as the angular velocity evolution (described in Figure 6.12), which does not correspond to the expected Torque values. The resulting corrected torque becomes very noisy after filtering due to the original torque feedback being filled with a significant amount of error. However, the corrected Torque (Figure 6.14) not only accurately represents the expected torque curve (albeit with some noise), but also correctly displays the expected constant torques during sections of constant acceleration. This provides confirmation that the calculated formula effectively filters out most of the error.

6.2.5 Leg Torque Gait Results

The same process of testing was then applied for the Front Left Shoulder and Wrist servos in order to create their respective formulas. Subsequently the gait performed in Section 6.1, was repeated and the Load feedback from each joint was recorded and passed through the calculated filters. Table 6.2 describes the Max and Min torque values for each joint in the simulated and real joints.

Table 6.2: Max and min torque peaks for the real and simulated robots.

Servo Joint	Max (N.m)	Min (N.m)
Real FL Shoulder	0.200	-0.221
Real FL Elbow	0.596	-0.322
Real FL Wrist	0.196	-0.930
Simulated FL Shoulder	0.537	-0.225
Simulated FL Elbow	0.593	-0.418
Simulated FL Wrist	0.043	-0.967

Analyzing Table 6.2, the Shoulder joint exhibits the most evident differences between the real and simulated load results. These large differences are expected as a consequence of the previously described communication and resolution limitations which prevent the servo from following the exact same path as it's simulated counterpart (Figure 6.2). Since the servo does not have as small of a resolution as the simulation servos, it often does not move as gradually as the simulation, leading to less torque-intensive movements.

The Elbow and Wrist joints show smaller differences between real and simulated values compared to the Shoulder joints. The discrepancies present are a consequence of filter noise as well as torque compensation from the Shoulder joint. The smaller differences for the Elbow and Wrist indicate a more accurate replication of intended movements, despite the presence of noise and compensation effects.

6.3 Distance Traveled

To assess the system's velocity, a series of tests were conducted to measure the distance traveled by the robot over a duration of 10 seconds using the same gait used in the previous tests. The average traveled distance was then calculated, and used as the reference point to calculate the average deviation.

In order to validate these results, the simulated traveled distance was also used as a reference point to calculate the deviation and error of each test iteration. Table 6.3 illustrates the results for these tests.

Table 6.3: Distance traveled over 10 seconds.

i	Simulated Traveled Distance (m)	Real Traveled Distance (m)	Average Deviation (m)	Sim Deviation (m)	Error (%)
1	0.9516	0.983	0.024	0.031	3.30
2		0.977	0.017	0.025	2.67
3		1.001	0.042	0.049	5.19
4		0.921	0.038	0.031	3.21
5		0.965	0.005	0.013	1.41
6		0.957	0.002	0.005	0.57
7		0.963	0.004	0.011	1.20
8		0.941	0.018	0.011	1.11
9		0.922	0.037	0.030	3.11
10		0.961	0.002	0.009	0.98
Average Traveled Distance		0.9591	Average Error		2.27

Analyzing Table 6.3 it is clear that the real traveled distance results closely match the simulated traveled distance, with an average error of 2.27% and a maximum error of 5.19%. This result validates the optimization done in Chapter 4, where distance traveled is maximized using NSGA algorithms. It is also important to note that the values are very repeatable, with a maximum deviation from the average traveled distance being only 0.042 m.

Chapter 7

Conclusion and Future Work

The main goal of this thesis was the development of a quadruped robot with modular legs. To achieve this, several steps were taken. First, a literature review was conducted on legged robotics, with special emphasis on quadruped robots, with the purpose of understanding the various types and respective advantages of these robots. The review revealed the benefits of mammal-type quadruped robots over sprawling-types, particularly in terms of speed and energy efficiency.

After examining the importance of degrees of freedom for mobility and comparing servo control protocols, a decision was made to develop a mammal-type quadruped robot with three degrees of freedom using UART-controlled servos. Additionally, TCP was chosen for its reliability as the communication protocol between systems.

Subsequently, a study on the available mammal-type quadruped 3D model frames was directed. Initially, the objective was to use one of these models with minimal modifications. However, due to compatibility issues and the potential for optimization, extensive changes were made to the 3D model. These modifications were necessary to accommodate Waveshare ST3215 Servos, as well as to shorten the leg links and reduce the overall weight significantly.

A simulation of the robot was then conducted using an open-source Pybullet simulator for quadruped robots. Modifying this simulator to more closely represent the real robot's weight and dimensions, allowed for the validation of the IK model and study of the torque evolution on each joint during a forward gait. After analyzing the results of the torque study, it was determined that gait optimization was necessary to ensure that the servos did not exceed their maximum torque and break. Initially, Simplicial Homology Global Optimization (SHGO) was used to find the gait parameters that minimized overdrive torque. This method provided optimal results regarding the overdrive torque, but returned solutions with overly conservative gait parameters that created gaits that were slower than necessary. To solve this, it was decided that a Multi-Objective optimization algorithm that minimized torque overdrive while maximizing the distance traveled in a fixed amount of time would be used. This enabled a study where three multi-objective genetic algorithms (NSGA-II, NSGA-III, and U-NSGA-III) were tested and compared, which concluded that the NSGA-III produced the best results for this situation. This multi-objective optimization study was also accepted as a full conference paper for the fourth edition of the Optimization, Learning

Algorithms and Applications international conference (OL2A 2024) [14].

After obtaining positive results from the simulation, the focus of the study shifted to the development of the real robot. The system was divided into two sub-systems: The PC which was responsible for calculating and loading gaits, as well as commanding the robot, and the ESP32 which saved and executed the gaits, commanded the servos, and received feedback.

It was through a Transmission Control Protocol that these two sub-systems communicated, using two types of messages, commands and load messages. The load messages carried the gaits that were calculated in the PC, to the ESP32, which read the message and saved it as the correct type of gait. The command messages, as the name implies, allowed the PC sub-system to send commands to the ESP32.

To create a better user experience and easier control, a graphical user interface was developed that allowed for complete control of the robot through command messages, along with the ability of creating, loading and executing completely custom gaits on the spot.

For the test phase, the NSGA optimized gait was ran on both the simulated and real robots, the joint positions were the recorded during the gaits and then compared. The results showed that the average error for movements under 1 degree (like in the shoulder joint) was 3.64% and for over 1 degree (like the elbow and wrist joints) it had an average error of $\approx 2.7\%$.

For the second test, the gaits were ran again and, instead of the positions of each servo, the torques were recorded. However, after analyzing the torque values the servos outputted through feedback, there was a large amount of error that had to be mitigated. To create a filter to mitigate this error, dynamic and constant velocity tests were conducted to determine the necessary parameters. After the filters were created, they were then applied to the original recorded torques, which when compared to the simulated torques, showed some discrepancies caused by residual noise from the faulty torque feedback.

The third and final test compared the distances traveled on the real and simulated robots over the same amount of time which resulted in an average error of 2.27%, thereby validating the accuracy of the system and simulator.

In conclusion, the developed prototype, along with the gait optimization and control algorithms, demonstrated successful operation in both real-world and simulation environments, consistently delivering accurate results.

Based on the conclusions drawn, there are several potential areas where the system could be improved in the future:

- **Raspberry Pi 4 and sensor integration:** Since the Waveshare ESP32 was limited to only having servo ports, adding a Raspberry Pi 4 would allow for the sensorization of the robot. Specifically the addition of an IMU (Inertial Measurement Unit) would allow for dynamic gaits as well as automatic yaw rate adjustments which would greatly improve stability during movement. Adding a camera or a LiDAR sensor could also allow for trajectory planning and collision avoidance for added safety.

- **Real-time gait loading:** Achieving real-time gait loading through simultaneous calculation of positions and execution, rather than pre-loading gaits, would allow for better stability, and smoother transitions between gaits. This requires using a faster method of communication like I2C or Serial communication with the Raspberry 4.0 previously mentioned.
- **Energy Optimization:** Optimizing for a more energy-efficient gait rather than solely minimizing overdrive torque can help optimize the angular speed while still taking torque into account.

References

- [1] Satoshi Shigemi, Ambarish Goswami, and Prahlad Vadakkepat. Asimo and humanoid robot research at honda. *Humanoid robotics: A reference*, pages 55–90, 2018.
- [2] Arne Roennau, Georg Heppner, Michal Nowicki, and Rüdiger Dillmann. Lauron v: A versatile six-legged walking robot with advanced maneuverability. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 82–87. IEEE, 2014.
- [3] Héctor Montes, Lisbeth Mena, Roemi E Fernández Saavedra, Javier F Sarria Paz, Pablo González-de Santos, and Manuel Armada. Hexapod robot for humanitarian demining. 2015.
- [4] Robugtix. T8X, 2015. URL: <https://www.robugtix.com/t8x>.
- [5] Satoshi Kitano, Shigeo Hirose, Atsushi Horigome, and Gen Endo. Titan-xiii: sprawling-type quadruped robot with ability of fast and energy-efficient walking. *Robomech Journal*, 3:1–16, 2016.
- [6] Boston Dynamics. Spot, 2015. URL: <https://bostondynamics.com/products/spot/>.
- [7] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019. doi:10.1109/ICRA.2019.8793865.
- [8] Josh Elijah. How to Program a Quadruped Robot with Arduino, 2016. URL: <https://makezine.com/article/technology/robotics/robot-quadruped-arduino-program/>.
- [9] Lianqing Yu, Yujin Wang, and Weijun Tao. Gait analysis and implementation of a simple quadruped robot. In *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, volume 2, pages 431–434. IEEE, 2010.
- [10] Yi-yuan Fang and Xue-jun Chen. Design and simulation of uart serial communication module based on vhdl. In *2011 3rd International Workshop on Intelligent Systems and Applications*, pages 1–4. IEEE, 2011.
- [11] Nathaniel Pinckney. Pulse-width modulation for microcontroller servo control. *IEEE potentials*, 25(1):27–29, 2006.
- [12] Jakub Plywaczewski. DOGGO - simple robot dog, 2021. URL: <https://www.thingiverse.com/thing:4916566>.
- [13] Jorge Filipe Magalhães Ferreira. Mapping, localization and trajectory planning for quadruped robots in highly uneven terrains. Master thesis, FEUP, 2022.

- [14] Gustavo Cohen, José Lima, and Paulo Costa. Enhancing quadruped robot performance through gait optimization. In *Optimization, Learning Algorithms and Applications (OL2A)*, 2024.
- [15] Rui Pedro Sousa Ferreira. Hybrid robot with leg & wheel locomotion. Master thesis, FEUP, 2023.
- [16] JA Tenreiro Machado and Manuel Silva. An overview of legged robots. In *International Symposium on Mathematical Methods in Engineering*, pages 1–40, 2006.
- [17] João Manuel Pinto Marques de Oliveira et al. Microsoft robotics soccer challenge: movement optimization of a quadruped robot. 2008.
- [18] Uwe Dueffert and Jan Hoffmann. Reliable and precise gait modeling for a quadruped robot. In *RoboCup 2005: Robot Soccer World Cup IX 9*, pages 49–58. Springer, 2006.
- [19] Vyacheslav Lyashenko, M Ayaz Ahmad, Nataliya Belova, and Svitlana Sotnik. Modern walking robots: a brief overview. 2021.
- [20] Manuel Fernando dos Santos Silva et al. Sistemas robóticos de locomoção multipernas. 2005.
- [21] C Dario Bellicoso, Marko Bjelonic, Lorenz Wellhausen, Kai Holtmann, Fabian Günther, Marco Tranzatto, Peter Fankhauser, and Marco Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018.
- [22] Boston Dynamics. Legacy Robots, 2004. URL: <https://bostondynamics.com/legacy/>.
- [23] Jennifer Chu. Mini cheetah is the first four-legged robot to do a backflip, 2019. URL: <https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304>.
- [24] David J Todd. *Walking machines: an introduction to legged robots*. Springer Science & Business Media, 2013.
- [25] Dong Jin Hyun, Jongwoo Lee, SangIn Park, and Sangbae Kim. Implementation of trot-to-gallop transition and subsequent gallop on the mit cheetah i. *The International Journal of Robotics Research*, 35(13):1627–1650, 2016.
- [26] Thomas A McMahon. The role of compliance in mammalian running gaits. *Journal of experimental biology*, 115(1):263–282, 1985.
- [27] Maria Silva Lopes. Quadruped manipulator for potential agricultural applications. Master thesis, FEUP, 2023.
- [28] Muhammad Amjad, Fayaz Akhtar, Mubashir Husain Rehmani, Martin Reisslein, and Tariq Umer. Full-duplex communication in cognitive radio networks: A survey. *IEEE Communications Surveys Tutorials*, 19(4):2158–2191, 2017. doi:10.1109/COMST.2017.2718618.
- [29] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles*. Butterworth-Heinemann, 2013.

- [30] Shikhar. Design and implementation of multiple pwm channels using universal asynchronous receiver transmitter. pages 485–492, 2020. doi:10.1109/ICECA49313.2020.9297566.
- [31] Luxorparts. SG90 Micro Servo, 2017. URL: https://www.kjell.com/globalassets/mediaassets/701916_87897_datasheet_en.pdf.
- [32] Hiwonder. LX-16A, 2021. URL: <https://www.hiwonder.com/products/lx-16a>.
- [33] Hiwonder. LX-225, 2021. URL: https://www.hiwonder.com/products/lx-225?_pos=1&_sid=b42d58496&_ss=r.
- [34] Waveshare Wiki. Waveshare ST3215 Servo, 2021. URL: https://www.waveshare.com/wiki/ST3215_Servo.
- [35] Raghunath Deshpande. *TCP Proxies Chaining: Performance Implications*. PhD thesis, 11 2015. doi:10.13140/RG.2.1.3486.9205.
- [36] W Eddy. Rfc 9293: Transmission control protocol (tcp), 2022.
- [37] Kim Deok-yeon. Spotmicro - robot dog, 2019. URL: <https://www.thingiverse.com/thing:3445283>.
- [38] Maurice Rahme, Ian Abraham, Matthew Elwin, and Todd Murphey. Spotminimini: Pybullet gym environment for gait modulation with bezier curves, 2020. URL: https://github.com/moribots/spot_mini_mini.
- [39] Dong Jin Hyun, Sangok Seok, Jongwoo Lee, and Sangbae Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *The International Journal of Robotics Research*, 33(11):1417–1445, 2014.
- [40] Stefan Endres, Carl Sandrock, and Walter Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72, 10 2018. doi:10.1007/s10898-018-0645-y.
- [41] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
- [42] Yusliza Yusoff, Mohd Salihin Ngadiman, and Azlan Mohd Zain. Overview of nsga-ii for optimizing machining process parameters. *Procedia Engineering*, 15:3978–3983, 2011.
- [43] Jianling Chen. Multi-objective optimization of cutting parameters with improved nsga-ii. In *2009 International Conference on Management and Service Science*, pages 1–4, 2009. doi:10.1109/ICMSS.2009.5302835.
- [44] Tønnes F Nygaard, Jim Torresen, and Kyrre Glette. Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.
- [45] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426, 2008. doi:10.1109/CEC.2008.4631121.

- [46] Christian Von Lücken, Benjamín Barán, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational optimization and applications*, 58:707–756, 2014.
- [47] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. doi:[10.1109/TEVC.2013.2281535](https://doi.org/10.1109/TEVC.2013.2281535).
- [48] Julian Blank, Kalyanmoy Deb, and Proteek Chandan Roy. Investigating the normalization procedure of nsga-iii. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 229–240. Springer, 2019.
- [49] Indraneel Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998. URL: <https://doi.org/10.1137/S1052623496307510>, doi:[10.1137/S1052623496307510](https://doi.org/10.1137/S1052623496307510).
- [50] Haitham Seada and Kalyanmoy Deb. U-nsga-iii: A unified evolutionary algorithm for single, multiple, and many-objective optimization. *COIN report*, 2014022, 2014.
- [51] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016.
- [52] Martin Andersson, Sunith Bandaru, Amos Ng, and Anna Syberfeldt. Parameter tuning of moeas using a bilevel optimization approach. In *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29–April 1, 2015. Proceedings, Part I* 8, pages 233–247. Springer, 2015.
- [53] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. doi:[10.1109/ACCESS.2020.2990567](https://doi.org/10.1109/ACCESS.2020.2990567).
- [54] Siew Mooi Lim, Abu Bakar Md Sultan, Md Nasir Sulaiman, Aida Mustapha, and Kuan Yew Leong. Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing*, 7(1):9–12, 2017.
- [55] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.
- [56] Fredrik Lundh. An introduction to tkinter. URL: www.pythonware.com/library/tkinter/introduction/index.htm, 1999.
- [57] Eric Jobiliong. The calculation of moment of inertia without integral. *Advances in Engineering and Technology*, page 51.