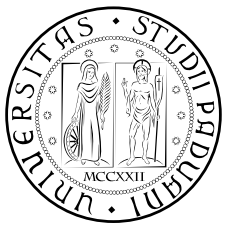


UNIVERSITY OF PADUA

DEPARTMENT OF ENGINEERING AND INDUSTRIAL SYSTEMS MANAGEMENT
BACHELOR'S DEGREE IN MECHATRONIC ENGINEERING

BACHELOR'S THESIS

Quadruped Robots: Design and Development of Locomotion Systems Inspired by Animal Biomechanics.

Supervisor: Reggiani Monica

Co-supervisor: Antonello Riccardo

Student: Matteo Piras
2033476-IMC

ACADEMIC YEAR: 2024-25

ABSTRACT

In recent years, quadruped robots have gained increasing importance due to their versatility. Their ability to adapt to diverse terrains, facilitated by machine learning algorithms and precise sensors, has made them highly agile and suitable for a wide range of tasks that humans cannot perform. These tasks include exploring toxic environments or areas with extreme temperatures, as well as transporting materials across rugged terrain. Understanding and studying these technologies has and will continue to have a significant impact on human life.

The study of robotics inspired by animal biomechanics has the potential to be a cornerstone in the development of technologies capable of replacing human labor with robots that can simulate the movements and functionalities currently unique to humans. For this reason, it is crucial to investigate technologies that can interact with the world using solutions inspired by nature's designs in quadruped animals.

This project aims to implement this concept by focusing on solutions for stability and movement.

May my work and effort somehow help to bring just a little more peace to this troubled world.

— Unknown

CONTENTS

1 INTRODUCTION 1
- 1.1 Previous Versions 1
- 1.2 State of the art and related trade-offs 2
  - 1.2.1 Actuation Modes 2
  - 1.2.2 Structural Parts 3
  - 1.2.3 Topological Structure of the Legs 5

2 STRUCTURAL DESIGN 7
- 2.1 Proportion 7
- 2.2 Motor configuration 7
- 2.3 Robot Body Shape 8
- 2.4 Joints 8
- 2.5 Feet 9
- 2.6 Materials used 9

3 ELECTRICAL OVERVIEW 11
- 3.1 Overview of Components 11
- 3.2 Component Arrangement 15
- 3.3 Circuit Diagram Description 16

4 INVERSE KINEMATICS 19
- 4.1 Definition 19
- 4.2 Implementation 19
  - 4.2.1 θ_{knee} 20
  - 4.2.2 θ_{hip} 21
  - 4.2.3 θ_{ankle} 22
- 4.3 Results 23

5 GAIT AND MOVEMENT 25
- 5.1 Stride 25
- 5.2 Gait 26
  - 5.2.1 Trot 26
- 5.3 Movements 26
  - 5.3.1 Standing-Sitting 27
  - 5.3.2 Forward-Backward Movement 27
  - 5.3.3 Right-Left Rotation 27

6 SERIAL COMMUNICATION 29
- 6.1 Configuration 29
- 6.2 Comparison 30

7 CONTROL METHOD 31
- 7.1 Introduction 31
- 7.2 Implementation 31
- 7.3 Autonomous Movement 33

8 EXPERIMENTAL RESULTS 35
- 8.1 Images of the implemented system 35
- 8.2 Results 36

8.3 Technical Specifications 36
9 CONCLUSIONS AND FUTURE WORK 37

BIBLIOGRAPHY 39

LIST OF FIGURES

Figure 1 Version 1. 1
Figure 2 Version 2. 1
Figure 3 Example of Hydraulic actuation. [5] 3
Figure 4 Example of Pneumatic Actuation. [5] 3
Figure 5 Example of Electric Motor Actuation. [4] 3
Figure 6 The types of body: a) rigid torso; b) torso with a revolute joint; c) torso with spring; and d) flexible material torso. [5] 4
Figure 7 a) Spring-loaded pantograph (SLP) for leg structures. b) The elastic load scissor mechanism. [5] 4
Figure 8 Degrees of Freedom for each leg. 4
Figure 9 Insect like. 5
Figure 10 Reptile like. 5
Figure 11 Mammal like.a)All-elbow structure b)All-Knee Structure c)Front-Knee and Rear-Elbow Structure d)Front-Elbow and Rear-Knee Structure 6
Figure 12 Horizontal Motor Mounting. 8
Figure 13 Frame. 8
Figure 14 Joint for carbon tubes. 9
Figure 15 90°joint. 9
Figure 16 Foot. 9
Figure 17 Carbon fiber tube. 10
Figure 18 Raspberry PI 4 Model B. 11
Figure 19 Arduino Mega. 11
Figure 20 Servomotor RDS3115MG. 12
Figure 21 LiPo Battery. 14
Figure 22 Buck Converter. 14
Figure 23 Voltage Regulator. 14
Figure 24 NRF24L01. 15
Figure 25 Servo driver. 15
Figure 26 Component Arrangement, level 2. 16
Figure 27 Circuit diagram. 16
Figure 28 Inverse Kinematics. 19
Figure 29 Projection on yz (leg). 19
Figure 30 Projection on xz (leg). 19
Figure 31 Projection on yz (θ_{knee}). 20
Figure 32 Projection on xz (θ_{knee}). 20
Figure 33 Projection on yz (θ_{hip}). 21
Figure 34 Projection on xz (θ_{ankle}). 22
Figure 35 Stride 1 25
Figure 36 Stride 2 25
Figure 37 Gaits [5] 26

Figure 38 Sitting. 27
Figure 39 Standing. 27
Figure 40 Movements. 28
Figure 41 Control Flow. 32
Figure 42 Movement with Two Coordinates. 33
Figure 43 Movement with Intermediate Steps. 33
Figure 44 Side View. 35
Figure 45 Front View. 35
Figure 46 Top View. 36

LIST OF TABLES

Table 1 Specifications of the RDS3115 servo motor. 12
Table 2 Comparison between execution time. 30
Table 3 Specifications 36

1 INTRODUCTION

The goal of this thesis is to develop a quadruped robot. The development process begins with an analysis of two quadruped robots, identifying problems and inefficiencies that need to be addressed. Subsequently, a state-of-the-art analysis is conducted to understand the technological choices and trade-offs necessary to integrate current technologies into this project.
The study then progresses to an investigation of structural design, electronics, control systems, and the results obtained through the design and implementation of this project.

1.1 PREVIOUS VERSIONS

Prior to the current project, two versions of quadruped robots were developed, but both exhibited numerous issues:

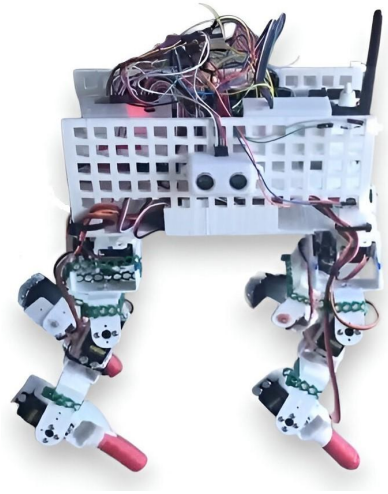


Figure 1: Version 1.

Figure 2: Version 2.

1. **First Version**
   The first version had very long legs with little lateral spacing between them, resulting in a high center of gravity and, consequently, significant instability.
   Additionally, the structural components of the legs were entirely 3D-printed with low infill density, which inevitably led to high fragility and frequent part failures.
   Furthermore, the control system was highly inefficient, as it relied on simple angular control rather than a position-based control system.

2. **Second Version**
   The second version addressed the issues of instability and leg fragility by incorporating aluminum components and 3D-printed parts with higher infill density compared to the previous version.
   However, this version still exhibited several issues, the most significant being leg length. The robot's legs were undersized relative to the torso dimensions, resulting in limited mobility and requiring a higher step frequency to achieve acceptable speeds.

To achieve a satisfactory result, it is essential to address these issues in this third version.
The main objectives to be achieved are:

- Stability and balance
- Structural durability
- Accurate and precise control

However, the design and construction of this project are constrained by time and budget limitations. The budget for this project is 500 euros, which significantly restricts the selection of certain high-cost components, such as motors.
Time is also a major constraint, as only approximately two months were available for development. As a result, several necessary compromises were made during the project's development.

1.2 STATE OF THE ART AND RELATED TRADE-OFFS

1.2.1 *Actuation Modes*

In the most advanced quadruped robots, three primary actuation modes are implemented:

- **Hydraulic actuation**
  Hydraulic actuation utilizes a fluid, typically hydraulic oil, to transmit energy by controlling the flow and pressure within hydraulic cylinders and valves, thereby enabling joint movement.
- **Pneumatic Actuation**
  Pneumatically actuated quadruped robots employ compressed air or gas to power actuators, such as pneumatic cylinders or artificial muscles, generating motion by regulating gas pressure. The control of pneumatically actuated robots involves adjusting gas pressure while monitoring actuator position and force to ensure precise operation.

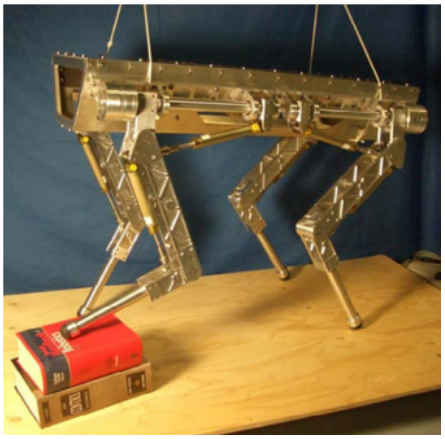


Figure 3: Example of Hydraulic actuation. [5]

Figure 4: Example of Pneumatic Actuation. [5]

- **Electric Motor Actuation**
  Motor-driven motion is achieved by controlling the motor's speed and torque to facilitate joint movement. Typically, operation requires a battery or an alternative power source. The control of motor-driven robots entails regulating the motor's speed, position, and current to ensure precise and efficient performance.

Figure 5: Example of Electric Motor Actuation. [4]

The first two methods are rarely used in practice due to their complexity and cost, making the third actuation method, which utilizes electric motors, the preferred choice for this project [5].

1.2.2 *Structural Parts*

The robot is composed of three distinct parts:

- **Frame or central body**
  In the main solutions used today, the frame can be either rigid or flexible. Although flexible frame designs theoretically offer better performance in terms of vibration reduction, shock ab-

sorption, and agility, they are challenging to implement in practice, as they introduce additional Degrees of Freedom (DoF). In the version to be implemented, a rigid frame will be used, as it is simpler to construct and implement [5].

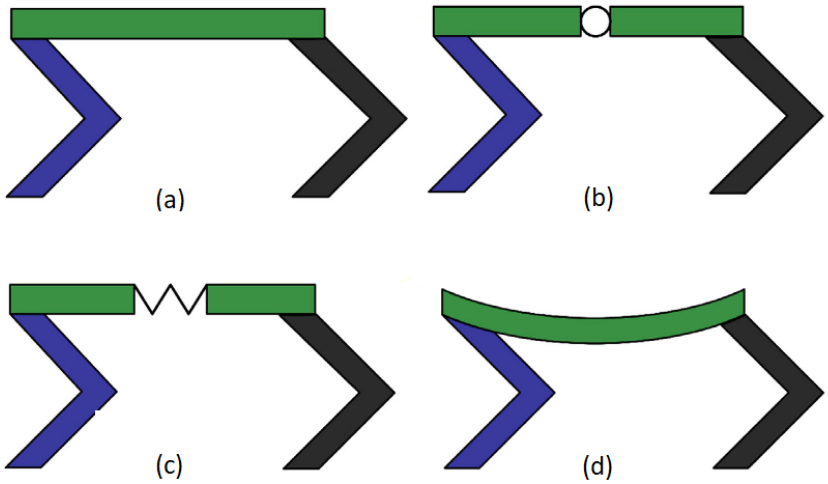


Figure 6: The types of body: a) rigid torso; b) torso with a revolute joint; c) torso with spring; and d) flexible material torso. [5]

- **Legs**
  In the main solutions used today, two types of legs are primarily implemented: scaled and linkage-based. Scaled legs involve excessive mechanical complexity, although they are more adaptable to various types of terrain. Linkage-based legs, particularly serial-linkage designs, will be used in this project due to their simplicity, wide range of motion, and direct control. Each leg has three Degrees of Freedom (DoF), enabling both forward and lateral movement. Overall, the robot, with its four legs, possesses 12 DoF, allowing for movement in two dimensions [5].

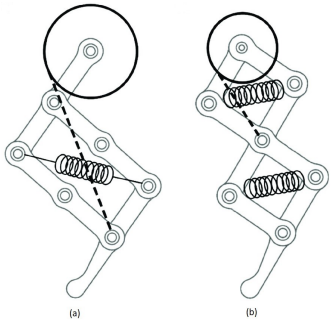


Figure 7: a) Spring-loaded pantograph (SLP) for leg structures. b) The elastic load scissor mechanism. [5]

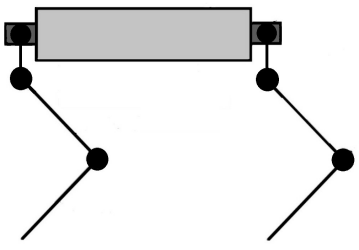


Figure 8: Degrees of Freedom for each leg.

- **Feet**
  In practice, two types of feet are commonly used: spherical/hemispherical and cylindrical/semicylindrical. In most application scenarios, the spherical design proves to be the most effective. While the cylindrical type offers a larger contact surface and thus greater static stability, the spherical type is more adaptable to various terrain types and multidirectional stresses compared to the cylindrical design. For this project, the spherical design has been selected [5].

1.2.3 *Topological Structure of the Legs*

In quadruped robots, the topological structure of the legs can be classified into three types:

- **Insect-like Structure**
  The insect-like quadruped robot features an angular leg arrangement that enhances static stability while minimizing leg interference. However, this configuration requires higher joint torques to maintain body balance and primarily supports static gaits.

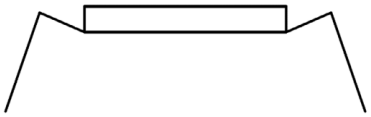


Figure 9: Insect like.

- **Reptile-like Structure**
  The reptile-like quadruped robot has a more horizontal leg orientation, offering a large single-leg workspace with reduced inter-leg contact. Despite improved static stability, its locomotion is often constrained by static gaits and the need for greater joint torques to sustain balance.

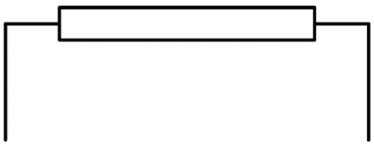


Figure 10: Reptile like.

- **Mammal-like Structure**
  Biomimetic quadruped robots modeled after mammals feature an extensive range of motion and superior obstacle avoidance capabilities. They excel in speed and dynamic performance, making them well-suited for agile and adaptive locomotion.

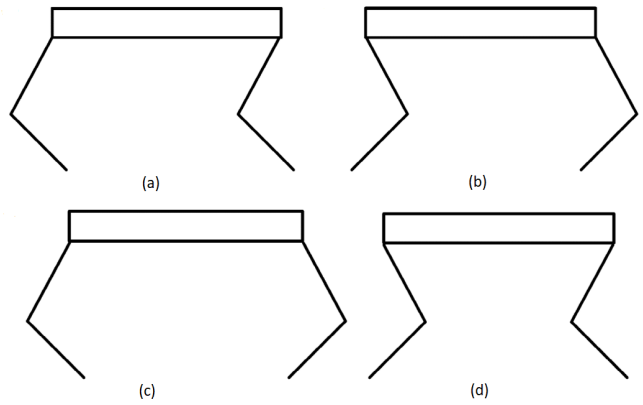


Figure 11: Mammal like.a)All-elbow structure b)All-Knee Structure c)Front-Knee and Rear-Elbow Structure d)Front-Elbow and Rear-Knee Structure

Although the first two topological structures offer greater static stability, the third topological structure mammal-like proves to be the best choice for this project in terms of agility and energy efficiency [3].
The mammal-like structure is further divided into four types:

- **All-elbow structure**
- **All-Knee Structure**
- **Front-Knee and Rear-Elbow Structure**
- **Front-Elbow and Rear-Knee Structure**

The configuration with all joints oriented as knees is rarely used in practice, as it is poorly suited for navigating rough terrain and climbing stairs due to the forward-facing knees, which may interfere with objects or obstacles. The other three configurations are commonly employed, particularly the all-elbow configuration and the front-knee and rear-elbow configuration.
For this project, the all-elbow configuration will be used due to its simplicity in design and implementation .

2 STRUCTURAL DESIGN

The structural design of a quadruped robot is of paramount importance for ensuring proper functionality and the complete implementation of the robot's movements.

2.1 PROPORTION

After extensive analysis and research in various scientific papers, it was concluded that there are no precise formulas or proportions to size the legs of a quadruped robot in relation to the length or overall dimensions of the torso [2] [3] [5]. In summary, while no universal standard formulas exist, advanced projects like Spot or MIT Cheetah rely on specific optimizations tailored to particular applications (stability, agility), with leg lengths determined through simulations and prototype testing.
However, by analyzing existing designs, I found that for a sufficient balance between stability and agility, leg lengths are typically about 60-70% of the torso length. Another important observation is that the lengths of the two straight elements of the leg are almost always the same.During the construction of the robot, it was necessary to resize the leg components, as their initial dimensions were overestimated.

The main issue with legs that are too long is stability and the torque required by the motor-leg system. Longer legs raise the center of gravity, significantly reducing stability. Additionally, the longer the legs, the greater the torque demand on the motors to achieve the same moment, leading to higher power consumption and an increased risk of motor stall.

Conversely, legs that are too short can result in limited mobility, restricting the range of motion.

To determine the optimal leg length, a series of tests were conducted, ultimately leading to the final design. In the final solution, each straight segment of the leg is approximately 50% of the total robot length.

2.2 MOTOR CONFIGURATION

Each leg consists of three motors that enable its full range of motion. The first motor, located near the robot's body, is horizontally oriented. This configuration was chosen to avoid limitations in lateral movements, which would otherwise be restricted due to the motor's

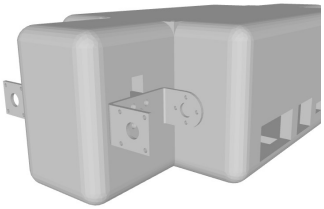


Figure 12: Horizontal Motor Mounting.

proximity to the robot's body. This arrangement ensures greater freedom of lateral motion for the leg, particularly towards the inner side.

2.3 ROBOT BODY SHAPE

The shape of the robot's body is a critical factor in its functionality and freedom of movement. The body is designed as a flattened, wide parallelepiped that houses all the components, with two narrower extensions added one at the front and one at the rear. These two extensions serve as mounting points for the four legs and are narrower to reduce the lateral distance between the legs.
Due to the robot's considerable lateral width and the horizontal orientation of the motors, the lateral distance between the legs would otherwise be too large, hindering the robot's ability to move freely and agilely. The motors are therefore mounted on these narrower blocks to address this issue.

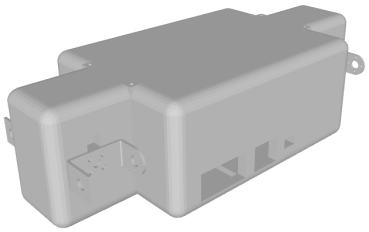


Figure 13: Frame.

2.4 JOINTS

Two types of joints were designed to connect the components of the legs: the first joint allows two brackets to be connected at a 90-degree angle, while the second joint connects the carbon fiber tubes to the metal brackets. These joints are among the elements in the legs that endure the highest structural stress and therefore must be extremely

robust. The joints are designed to accommodate threaded inserts for bolts, ensuring optimal attachment. They also feature a hole that allows for securing the carbon fiber tubes.

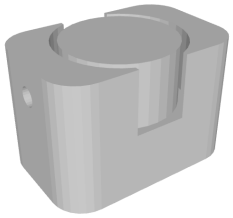


Figure 14: Joint for carbon tubes.

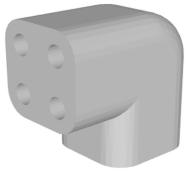


Figure 15: 90°joint.

2.5 FEET

The foot is designed with a housing to accommodate the carbon fiber tube, which is then secured with a bolt and a threaded insert.

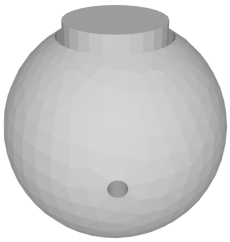


Figure 16: Foot.

2.6 MATERIALS USED

In most applications, the materials most commonly used are carbon fiber or various aluminum alloys [5]. Due to time and cost constraints, aluminum will not be used in this project, except for components connected to the motors.

- **Frame or central body**
  To construct the central frame of the robot, PLA (Polylactic Acid) was chosen, a material commonly used in 3D printing due to its lightweight and durable properties. The frame is the largest structural component and, therefore, must be as light as possible. Additionally, it is not required to withstand significant mechanical loads. As a result, it was printed with a 15% infill, ensuring extreme lightness (554g, including the top cover

and housing for the radio antenna) while maintaining moderate structural integrity.

- **Legs**
  The legs, defined as the connecting elements between one motor and the next, were constructed using hollow carbon fiber tubes. Given that the leg structure is the heaviest part of the robot (approximately 1kg due to the motors, each weighing 60g), it is crucial to use a material like carbon fiber, which is both extremely lightweight and highly durable.

Figure 17: Carbon fiber tube.

- **Feet**
  In most applications, the feet are made from rubber-like materials that allow for both shock and vibration absorption as well as excellent ground adherence. For this project, TPU (Thermoplastic Polyurethane) was selected, a material known for its high flexibility and rubber-like properties. TPU is particularly advantageous for its ability to absorb vibrations and provide strong traction with the ground. The characteristics of a TPU-printed object vary depending on the infill percentage used during printing. With a high infill (>60%), the object becomes relatively rigid, similar to an eraser, whereas a lower infill (<20%) results in a more flexible and compressible object. For the robot's feet, a 15% infill was chosen, striking a trade-off between the functionalities mentioned above without making the object overly flexible.

- **Joints**
  They were printed in PLA with 100% infill to provide maximum structural strength. Additionally, due to their relatively small size, they do not significantly contribute to the overall weight, even though they are highly dense.

3

ELECTRICAL OVERVIEW

3.1 OVERVIEW OF COMPONENTS

The components were selected by balancing functionality and cost:

1. **Raspberry PI 4 Model B**
   The Raspberry Pi plays a key role within the robot, serving as the platform for performing inverse kinematics calculations and communicating via serial connection with the Arduino. It runs a Raspbian operating system.

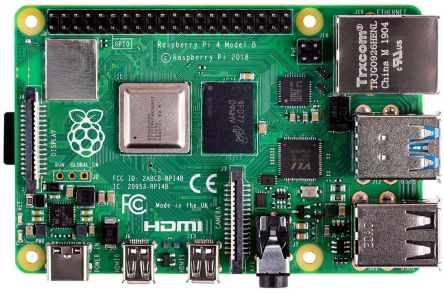


Figure 18: Raspberry PI 4 Model B.

2. **Arduino Mega 2560 Rev 3**
   The Arduino serves as the central element of the project, controlling the motors and other system components, including sensors and the radio module. The Mega model was chosen over others for its higher computational capacity and greater number of digital pins.

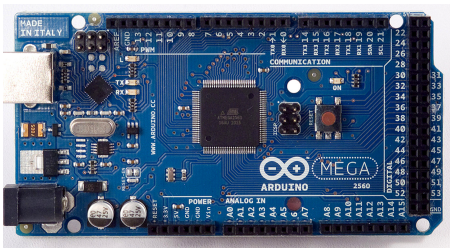


Figure 19: Arduino Mega.

3. **Servomotor RDS3115MG**
   The selection of actuators is of paramount importance, as the choice of one motor type over another significantly impacts the

final characteristics of the robot. In robots like Boston Dynamics' SPOT, Brushless DC Motors are generally used due to their high torque and precision. However, they are very expensive and more complex to implement, requiring the use of gear reducers and belts.
For this project, servomotors were chosen, specifically the RDS3115MG model, due to their availability and low cost. While they are less suited for this type of robot compared to brushless motors, they are still a good choice thanks to their high angular resolution.

Figure 20: Servomotor RDS3115MG.

Parameter	Value
Dimensions	40×20×40 mm
Stall Torque	17 kg-cm
Weight	60 g
Operating Voltage	4.8-6.8 V
Rotation Angle	180° / 270°
Stall Current(at 6.8V)	2.2 A
Operating Frequency	50-330 Hz

Table 1: Specifications of the RDS3115 servo motor.

4. **3S 11,1V LiPo battery 70C 7100mAh**
To properly size a battery for our system, we must calculate the power consumption of all its components. The primary consumers are the 12 servomotors, though for estimation purposes, we assume a simultaneous usage of 8 to 9 motors at any given time. According to the technical specifications, each motor has a stall current of 2.2A at 6.8V, so we approximate an operating current of around 1.2A per motor. After the motors, the Raspberry Pi is the next most power-demanding component, requir-

ing approximately 2.5–3A for stable operation. The remaining electronic components consume less than 200mA in total.

Below, the calculations for estimating the average battery runtime are presented.

Battery Usage Calculation

$$I_{servo} = 9 \times 1.2\,A = 10.8\,A \tag{1}$$

$$P_{servo} = 6.8\,V \times I_{servo} = 6.8\,V \times 10.8\,A = 73.44\,W \tag{2}$$

$$P_{in,servo} = \frac{P_{servo}}{\eta} = \frac{73.44\,W}{0.95} \approx 77.31\,W \tag{3}$$

$$I_{batt,servo} = \frac{P_{in,servo}}{V_{batt}} = \frac{77.31\,W}{11.1\,V} \approx 6.96\,A \tag{4}$$

$$P_{elett} = 5\,V \times I_{elett} = 5\,V \times 3.2\,A = 16\,W \tag{5}$$

$$P_{in,elett} = \frac{P_{elett}}{\eta} = \frac{16\,W}{0.95} \approx 16.84\,W \tag{6}$$

$$I_{batt,elett} = \frac{P_{in,elett}}{V_{batt}} = \frac{16.84\,W}{11.1\,V} \approx 1.51\,A \tag{7}$$

$$I_{batt,tot} = I_{batt,servo} + I_{batt,elett} \approx 6.96\,A + 1.51\,A \approx 8.47\,A \tag{8}$$

$$t_{batt} = \frac{Batterycapacity}{I_{batt,tot}} = \frac{7100mAh}{8.47A} \approx 50min \tag{9}$$

To power the entire system, an 11.1V 7100mAh LiPo battery was chosen.

Figure 21: LiPo Battery.

5. **DC-DC Buck Converter 7-24V to 5V 4A**
   The buck converter is used to step down the 11.1V supply voltage to 5V, required to power the Raspberry Pi.

Figure 22: Buck Converter.

6. **Voltage Regulator**
   The voltage regulator is used to step down the 11.1V supply voltage to approximately 6.8V to power the motors.

Figure 23: Voltage Regulator.

7. **Radio module**
   The NRF24L01 module, compatible with Arduino, was selected for radio communication.

8. **PWM/Servo Driver**
   To interface with and control the motors, an I2C module with 16 channels for PWM control is used.

9. **MPU6050 Gyroscope Accelerometer Module**

Figure 24: NRF24L01.

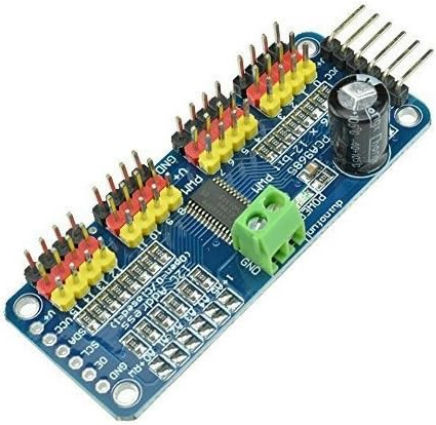


Figure 25: Servo driver.

3.2 COMPONENT ARRANGEMENT

To achieve proper overall balance for the robot, it is essential to arrange the internal components in a way that avoids overloading certain legs relative to others. A weight imbalance caused by improper component distribution can result in the robot falling during walking or making certain movements more difficult.
Internally the robot has two levels. The lower level houses the battery, which is the largest and heaviest component (almost 0.5 kg), in order to lower the robot's center of gravity as much as possible. In addition to the battery, a switch and the battery charge indicator are mounted on the robot's frame [13].
The second level contains the remaining components, namely the Raspberry Pi 4 (78g), the Arduino Mega (36g), and the voltage regulator for the motors (40g). Due to limited space, it is not possible to achieve precise balance for the components on the second level, but they can be arranged to mitigate significant imbalances. The combined weight of the Arduino Mega and the voltage regulator is equivalent to the weight of the Raspberry Pi, allowing them to be arranged as shown in the diagram to achieve sufficient balance.
To balance the switch and the battery charge indicator, the radio receiver will be positioned on the opposite side of the second level, just outside the frame. The remaining components, such as sensors, weigh less than 10g and are therefore not considered significant in terms of balancing. The remaining internal components, such as cables and the buck converter, do not have sufficient weight to unbalance the robot and are therefore also not considered in terms of balancing. A critical factor in the robot's equilibrium is the motors, which, due to the leg configuration, tend to shift the center of gravity backward relative to the geometric center of the robot's body. However, this issue is compensated for through the control method [7].

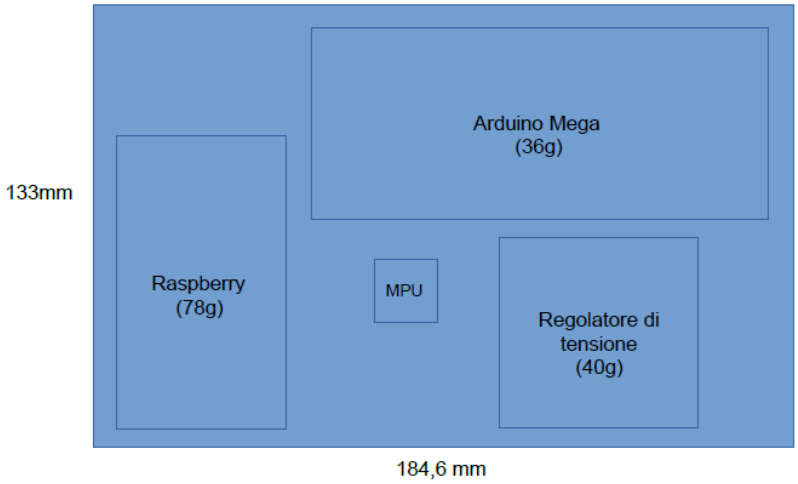


Figure 26: Component Arrangement, level 2.

3.3 CIRCUIT DIAGRAM DESCRIPTION

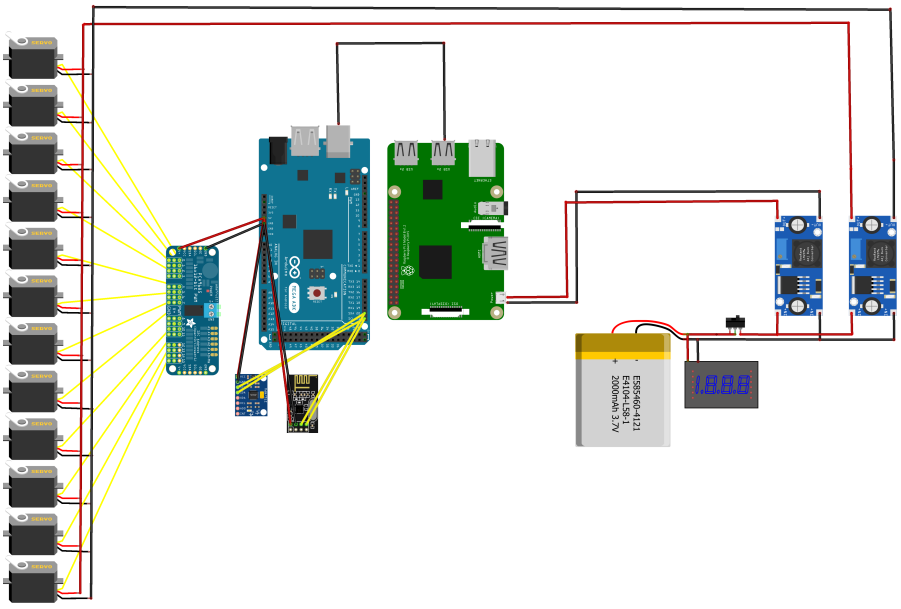


Figure 27: Circuit diagram.

The system is powered by an 11.1V LiPo battery, which supplies power depending on the state of the main switch. Additionally, a digital voltmeter is included to monitor the battery voltage in real-time. The power supply is connected in parallel to a voltage regulator and a buck converter, which step down the voltage to 6.8V for the motors and 5V for the Raspberry Pi, respectively. The Raspberry Pi is connected to the Arduino via a serial interface and also provides power to it.

The motors are powered in parallel and controlled by the Arduino through a PWM control module, which is connected to the Arduino via an I2C interface. The Arduino also powers and communicates with the gyroscope and the radio module, which enables communication with the remote controller.

4

INVERSE KINEMATICS

4.1 DEFINITION

Inverse kinematics is a fundamental component for achieving efficient control of the robot's movements. As previously mentioned, the system consists of 12 motors, each with angular control, allowing movement within a 270° range. Even if the exact movement of one of the four legs is predetermined, it is practically impossible to control the robot precisely by directly commanding the motors without a mathematical framework.

Inverse kinematics[7] enables the transition from direct angular control of individual motors to position-based control of the entire leg. This approach allows movement to be defined in terms of the desired position of the leg's terminal point in three-dimensional space, rather than manually adjusting each motor's angle.

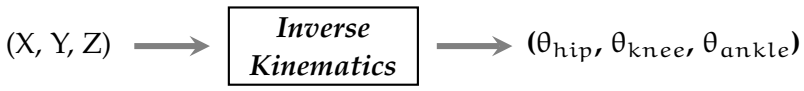


Figure 28: Inverse Kinematics.

4.2 IMPLEMENTATION

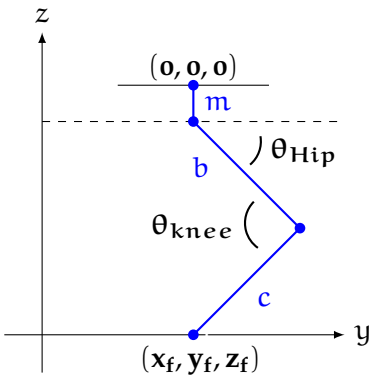


Figure 29: Projection on yz (leg).

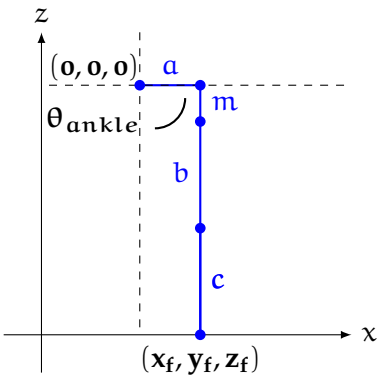


Figure 30: Projection on xz (leg).

Each leg, when simplified, can be represented as a series of triangles from both the front and lateral perspectives. These triangles have the straight segments of the legs as their sides, while their angles correspond to the joints between each segment and the motors. This

approach provides a simplified representation of the legs; for clarity, we consider only one leg, as all four are identical.
By analyzing these triangles, we can derive the equations for calculating the three key angles of interest: $\theta_{hip}, \theta_{knee}, \theta_{ankle}$. Since the lengths of the straight segments a, b, c and m are known, the following trigonometric calculations can be used to determine these angles:

4.2.1 θ_{knee}

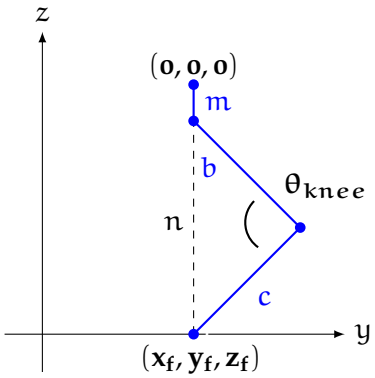


Figure 31: Projection on yz (θ_{knee}).

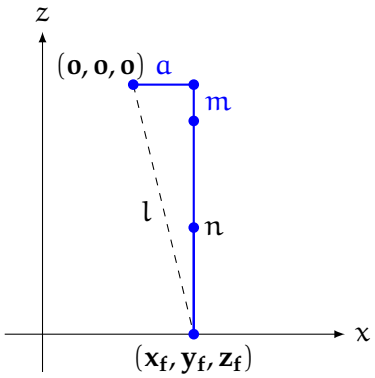


Figure 32: Projection on xz (θ_{knee}).

Calculation of θ_{knee}

$$l = \sqrt{x_f^2 + y_f^2 + z_f^2} \tag{10}$$

$$l^2 = (n + m)^2 + a^2 \Longrightarrow n = \sqrt{l^2 - a^2} - m \tag{11}$$

$$n^2 = b^2 + c^2 - 2 \cdot b \cdot c \cdot \cos(\theta_{knee}) \tag{12}$$

$$\theta_{knee} = \arccos\left(\frac{b^2 + c^2 - n^2}{2 \cdot b \cdot c}\right) \tag{13}$$

4.2.2 θ_{hip}

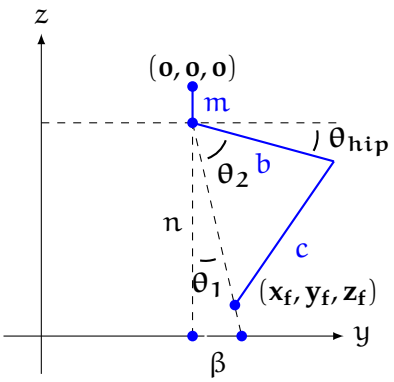


Figure 33: Projection on yz (θ_{hip}).

Calculation of θ_{hip}

$$\theta_{hip} = 90 - \theta_1 - \theta_2 \tag{14}$$

$$\frac{\sin\beta}{n} = \frac{\sin\theta_1}{-y_f} \Longrightarrow \theta_1 = \arcsin\left(\frac{-y_f}{n}\right) \tag{15}$$

$$c^2 = b^2 + n^2 - 2 \cdot b \cdot n \cdot \cos(\theta_2) \Longrightarrow \theta_2 = \arccos\left(\frac{b^2 + n^2 - c^2}{2 \cdot b \cdot n}\right) \tag{16}$$

$$\theta_{hip} = 90 - \arcsin\left(\frac{-y_f}{n}\right) - \arccos\left(\frac{b^2 + n^2 - c^2}{2 \cdot b \cdot n}\right) \tag{17}$$

4.2.3 θ_{ankle}

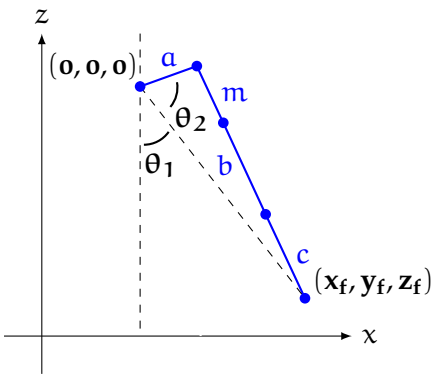


Figure 34: Projection on xz (θ_{ankle}).

Calculation of θ_{ankle}

$$\theta_{ankle} = \theta_1 + \theta_2 \tag{18}$$

$$\theta_1 = \arctan(\frac{x_f}{z_f}) \tag{19}$$

$$\theta_2 = \arccos(\frac{a}{\sqrt{x_f^2 + y_f^2}}) \tag{20}$$

$$\theta_{ankle} = \arctan(\frac{x_f}{z_f}) + \arccos(\frac{a}{\sqrt{x_f^2 + y_f^2}}) \tag{21}$$

4.3 RESULTS

Thanks to inverse kinematics, we have derived three equations that allow us to determine the motor configuration from any given set of Cartesian coordinates.

Results

$$\theta_{knee} = \arccos\left(\frac{b^2 + c^2 - n^2}{2 \cdot b \cdot c}\right) \tag{22}$$

$$\theta_{hip} = 90 - \arcsin(\frac{-y_f}{n}) - \arccos(\frac{b^2 + n^2 - c^2}{2 \cdot b \cdot n}) \tag{23}$$

$$\theta_{ankle} = \arctan(\frac{x_f}{z_f}) + \arccos(\frac{a}{\sqrt{x_f^2 + y_f^2}}) \tag{24}$$

5 GAIT AND MOVEMENT

5.1 STRIDE

In this project, the stride is defined as the movement performed by a single leg to enable locomotion. In the case of forward movement, from which all other movement types such as lateral and backward motion are derived, there are two types of strides.
In the first type, the movement consists of two phases:

- A first phase in which the terminal part of the leg follows a forward semicircular trajectory.
- A second phase in which the leg, in contact with the ground, moves backward for a distance equal to the diameter of the semicircle, generating thrust and thus enabling movement.

The second type of stride, in contrast, consists of four movements. One of these movements is shared with the first type, while the other three (all linear) replace the semicircular movement, which has proven to be less effective.
In this second approach, the movement follows the perimeter of a rectangle rather than the semicircle used in the first type.
Although the first type of stride is simpler, practical testing has

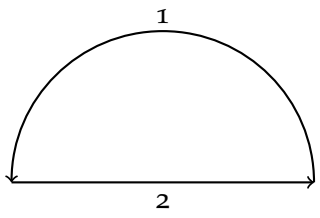


Figure 35: Stride 1

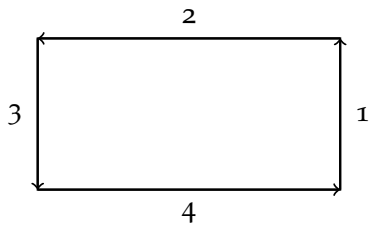


Figure 36: Stride 2

shown that the second type is more effective, as it reduces the likelihood of the robot stumbling. Another type of stride was also tested, similar to the first but with the difference that instead of a linear return movement after the semicircular phase, a second, less eccentric downward semicircle was used to theoretically improve ground adherence. However, this method also proved to be unstable and was therefore discarded.

5.2 GAIT

The movement of the robot's legs and the sequence of lifting and placing its four feet on the ground is referred to as gait. Different types of gait affect the robot's stability and speed, making them a crucial factor in the development of the control method.
Various gait types are used in practice, primarily classified by the number of legs that are simultaneously off the ground for a short period during movement. The higher this number, the greater the robot's sustained speed, but the lower its stability.
Due to construction and implementation constraints, only one type of gait will be implemented in this project [5].

5.2.1 *Trot*

In this gait, a diagonal pair of legs is lifted and placed simultaneously. This type of gait is significantly faster and more dynamic compared to a walking gait but is also slightly less stable.
The trot and walking gait are the only implementable gaits in this project. Other gait types are extremely dynamic, requiring high speeds and, in some cases, involving more than two legs lifted off the ground simultaneously, making them impractical for this system.

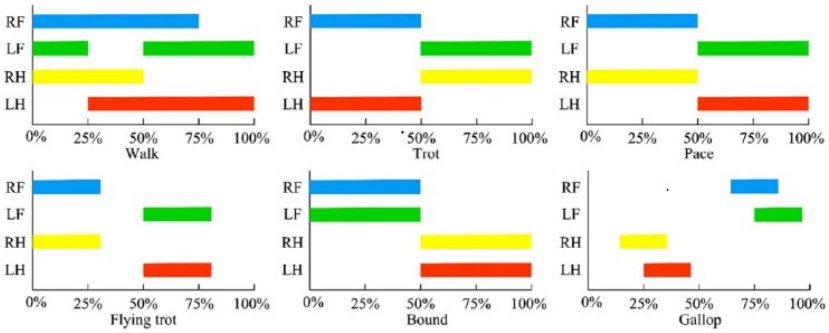


Figure 37: Gaits [5]

5.3 MOVEMENTS

In this project, six different movements have been implemented, three of which are complementary to each other. Three movements are considered complementary because one can be derived from another with minor modifications, and vice versa.

5.3.1 *Standing-Sitting*

The starting position is one in which the second straight segment of the leg is parallel to the ground, and the motors do not need to exert any torque to keep the robot in position. From this initial position, the robot can transition to a standing posture and return to the ground once its operation is complete.
However, this movement places the highest stress on the motors and requires the greatest torque to overcome the system's initial inertia, considering the robot's total weight of 3 kg. Another challenge is that the elbow-like leg configuration shifts the center of gravity backward relative to the geometric center of the torso, as the knee motors are positioned toward the rear.
During testing, this characteristic caused excessive strain on the rear motors, which frequently risked stalling. To address this issue, the robot shifts its torso forward during the standing movement to compensate for the imbalance.

Figure 38: Sitting.

Figure 39: Standing.

5.3.2 *Forward-Backward Movement*

Another pair of complementary movements, perhaps the most important, is the forward and backward movement. This movement is fundamental, allowing the robot to navigate through space at a speed determined by the stepping frequency commanded by the system.
The trot gait has proven to be highly effective, maintaining a reasonable level of stability while minimizing vibrations during movement. The backward movement is derived by reversing the motor control sequence used for the forward movement.

5.3.3 *Right-Left Rotation*

The final pair of movements allows the robot to rotate right and left. This pair has proven to be by far the most complex to implement. Unlike other movements, which exclude the use of four motors (those directly attached to the robot's body), lateral movements require the

use of all 12 motors instead of just 8. This detail significantly increases the complexity of the movement.
Additionally, rotation is more prone to instability compared to forward and backward movement, making the robot more likely to fall. During the development process, a key consideration was whether it would be more effective to take small steps at a high frequency or longer steps at a lower frequency. After extensive testing, the chosen approach was to use long but slow steps, minimizing torso oscillation.
Torso oscillation occurs because the motors are rigidly coupled to the legs, but due to mechanical tolerances, there is some play in their connections rather than a perfectly rigid coupling. This characteristic has both advantages and disadvantages: on the one hand, it helps absorb shocks and vibrations, but on the other, it can cause excessive oscillation during movement if not properly controlled. To mitigate this, the play between the motors is compensated to maintain accurate control and prevent performance degradation.

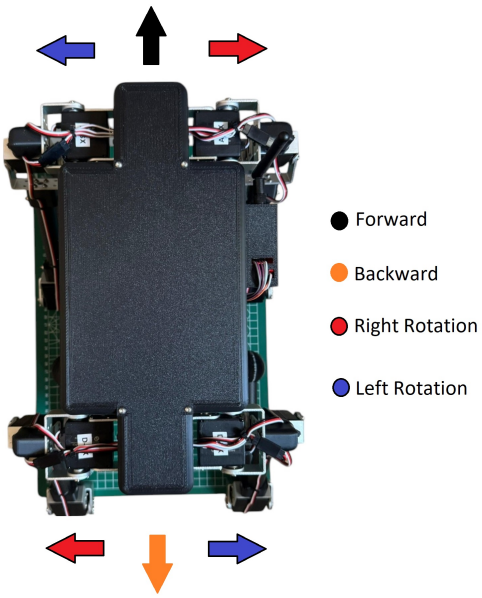


Figure 40: Movements.

6 SERIAL COMMUNICATION

6.1 CONFIGURATION

From a computational perspective, the most demanding workload in the system comes from the inverse kinematics calculations required to determine the angles for controlling the robot's motors. Since these calculations involve between 500 and 1500 trigonometric operations per second, depending on the type of movement, performing them on the Arduino would be inefficient. Even with the use of the Arduino Mega, which has greater computational capacity than the base model (Arduino UNO), the system would not be sufficiently performant to handle the required number of calculations.
One of the primary reasons for Arduino's inefficiency in executing such trigonometric calculations, particularly floating-point operations, is the absence of a Floating-Point Unit (FPU), which typically manages these computations. Without an FPU, Arduino is forced to perform floating-point operations using software emulation, which is highly inefficient for intensive workloads. In contrast, the Raspberry Pi 4 has an integrated FPU and is capable of executing significantly more operations per second than Arduino.
Another advantage of integrating the Raspberry Pi into the project is its ability to interface with more complex peripherals, such as LiDAR sensors or cameras for computer vision. This allows for a distinction between high-level control (complex calculations, vision) managed by the Raspberry Pi and low-level control (sensors, motors) handled by Arduino, with a communication protocol coordinating the two systems.
A factor to consider is the latency introduced by the need for Arduino to query the Raspberry Pi via a serial connection whenever a motor command is required. Arduino must send a request with the coordinates for computing the three angles needed to control a leg's motors, wait for the Raspberry Pi to process and return the values, and only then actuate the motors. However, this approach has proven to be significantly faster and more efficient than offloading all computational tasks to Arduino, thanks to the Raspberry Pi's high processing speed and the low communication latency between the two devices.
The system is configured to communicate at a baud rate of 115200 (115200 bits per second), which provides an optimal balance between communication reliability and processing speed. A higher communication frequency could lead to packet loss or data corruption if the hardware and software are not properly configured. The chosen li-

brary for managing serial communication with Arduino is WiringPi, which allows the Raspberry Pi to interface with the UART. On the Arduino side, communication is handled through its primary serial port (Serialo).

6.2 COMPARISON

During the development of the project, a test was conducted to determine whether it was more efficient to perform inverse kinematics calculations on the Raspberry Pi[6] or if executing them on Arduino[1] would be an optimal solution.
To measure execution time, a parallel test was performed on both devices, where each was tasked with executing a significant number of operations (2000–3000) using the equations derived from inverse kinematics. The results were then divided by the number of iterations performed to obtain a sufficiently reliable estimate of the execution time for a single iteration on both devices.
Subsequently, the total time required for data transmission from Arduino to Raspberry, computation, and the return of results was measured to compare it with the execution time of running the calculations locally on Arduino.
As expected, the Raspberry Pi requires less than 0.5% of the time taken by Arduino for a single iteration of calculations. Even when considering the transmission and reception time, the serial communication latency performing the calculations on Raspberry Pi is still 87% faster than executing them locally on Arduino.

Table 2: Comparison between execution time.

Execution Time	**Arduino Mega**	**Raspberry PI 4**
Single iteration	928 µs	4 µs
Single Iteration + Serial Communication	928 µs*	124 µs

* This number is equal to the previous one because, if the calculations were performed on the Arduino, there would be no latency due to serial communication, as everything would be executed locally.

7 CONTROL METHOD

7.1 INTRODUCTION

The control method for a quadruped robot is a crucial component that enables the robot to move freely while fully utilizing the engineering choices and mechanical implementations incorporated into its design. Within the control method, both trajectory planning and gait planning are defined.
In practice, numerous methods exist for implementing trajectory planning, often relying on complex algorithms of various types, such as:

- **Central Pattern Generator (CPG)**
  The CPG mimics natural neural networks to generate rhythmic movements, enabling a smooth and adaptive gait.
- **Spring-Loaded Inverted Pendulum (SLIP)**
  This simplified model describes the robot's behavior as a spring-loaded pendulum, utilizing elastic energy to represent running and jumping motions.
- **Zero Moment Point (ZMP)**
  The ZMP method ensures that the resultant force of gravity and inertial forces remains within the robot's support polygon, maintaining balance and stability.

All these methods are too complex from both a mathematical and computational standpoint for this project. Additionally, the available time is insufficient to implement them effectively [3].

7.2 IMPLEMENTATION

To implement an effective and simple control method, this project will adopt a Model-Free Control approach [3]. This method does not rely on complex mathematical algorithms to simulate and predict the robot's dynamics but instead utilizes a non-adaptive control strategy. In this project, the robot's movement patterns will be predefined rather than dynamically adjusted.
The system, as previously explained, relies on serial communication between Arduino and Raspberry Pi for motor control and angle computation.
Once powered on, the Arduino initializes all peripherals and sets the motor positions so that the robot starts in a sitting position without requiring immediate communication with the Raspberry Pi. This is

achieved using a lookup table, which stores the predefined motor positions for the robot's powered-off state, preventing delays caused by waiting for the Raspberry Pi to boot and initialize serial communication.

The Raspberry Pi, once powered on, takes approximately 20–25 seconds to boot properly and execute the script that enables serial communication. This script performs a handshake with the Arduino by sending and receiving a message to ensure that both devices are ready for serial communication.

Once initialized, the Raspberry continuously listens for coordinate packets from the Arduino. Upon receiving these packets, they are processed in a separate script and then sent back with the computed motor angles before returning to listening mode.

The Arduino, after setting the initial motor values, starts the serial interface and establishes radio communication with the remote controller through the dedicated module. The microcontroller then enters a loop, listening for signals from the remote controller that, depending on the received signal, trigger the execution of a specific movement.

Depending on the movement requested by the system, the Arduino

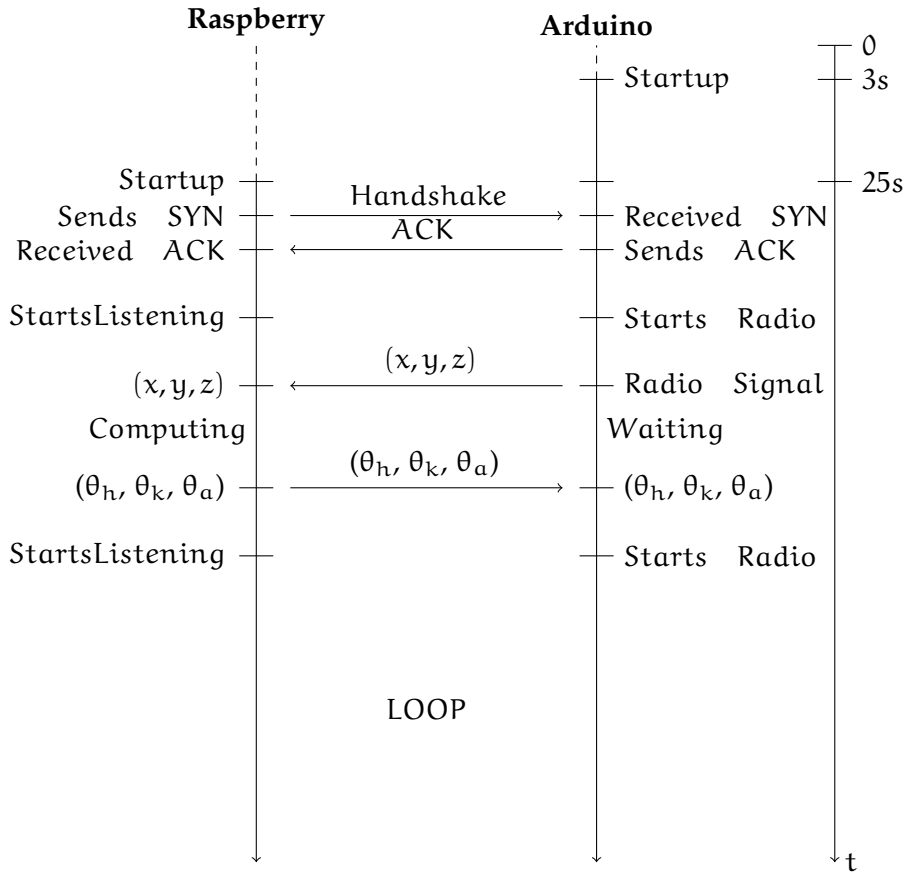


Figure 41: Control Flow.

processes a sequence of commands to control the motors accordingly. If no signals are received from the remote controller, the control system halts any ongoing movement sequence, and the robot returns to its default standing position, waiting for new commands.
To execute the previously defined stride movements, it is not sufficient to provide only the terminal positions of the vectors to achieve smooth and continuous motion.
The issue that would arise if only the coordinates of the four vertices were given is that there would be no guarantee that the robot's legs would follow a straight path between each vertex. This is crucial for maintaining ground contact and ensuring effective locomotion.
Therefore, it is necessary to compute a series of intermediate positions for the leg to follow, ensuring that the terminal part of the leg adheres precisely to the desired trajectory.

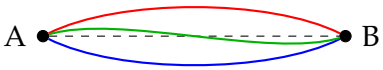


Figure 42: Movement with Two Coordinates.

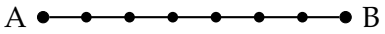


Figure 43: Movement with Intermediate Steps.

7.3 AUTONOMOUS MOVEMENT

The ability of a quadruped robot to move and orient itself in an unknown environment, adapting its movements to its surroundings, represents the turning point from a human-operated machine to a system capable of autonomous movement.
It is essential for the robot to leverage onboard sensors to orient itself, understand the terrain, and detect obstacles. For this reason, a gyroscope was chosen for this project. After an initial calibration, this type of sensor provides the system with inclination measurements along the three axes, allowing the robot to assess whether it is in an unstable position for example, if one of its four legs is resting on an elevated surface compared to the others.
The challenge in this project is that during the trot gait, the robot experiences frequent vibrations and mechanical stress, making it difficult to interpret the gyroscope data reliably. A minor vibration caused by walking could be misinterpreted as an instability, leading the system to apply corrections for a problem that does not actually exist.
At the beginning of this project, I envisioned using the gyroscope as an aid for autonomous movement, implementing an adaptive stability control system based on inclination measurements during locomotion. The idea was to allow the robot to autonomously adjust the position of its four legs, maintaining a horizontal posture without interfering with the ongoing movement.
This type of control is classified as adaptive control, where the system continuously adjusts its movement based on real-time sensor feed-

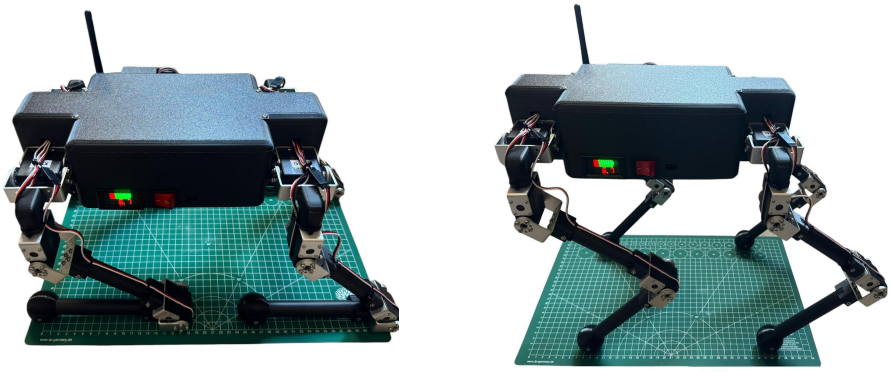
back. However, in this project, adaptive control presents a significant issue: when the system experiences unexpected forces or terrain variations, it attempts to correct its gait, potentially conflicting with its current movement execution.
An alternative approach, predictive control, is based on mapping the environment to anticipate the most suitable gait for the terrain before movement occurs. In my opinion, this is a far more effective approach. A system that reacts to disturbances will always struggle with stability and fluidity, compared to a system that predicts and adjusts its movements in advance.
Additionally, predictive control can help the robot recognize impassable terrain and stop before falling, whereas adaptive control would attempt to correct its gait after the fact, potentially resulting in missteps such as placing a foot into a deep hole and subsequently falling.
During development, a static balancing system based on a gyroscope was implemented, which adjusts the leg posture according to the type of inclination. This system considers the inclination along the two vertical axes and utilizes these values through a simple algorithm to autonomously determine the posture of the four legs. Therefore, the gyroscope can be used to achieve stability, but only in static conditions, which is of limited usefulness when the robot is in motion.
In conclusion, while the gyroscope is an interesting tool for controlling the robot, it cannot serve as a central element for the robot's spatial orientation and overall stability [5].

8

EXPERIMENTAL RESULTS

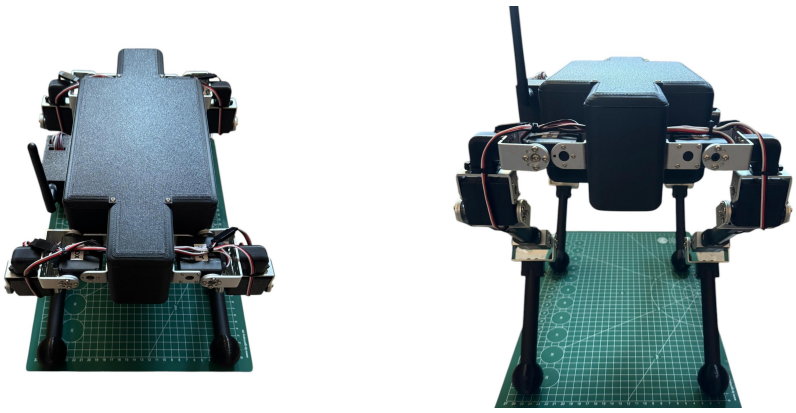
8.1 IMAGES OF THE IMPLEMENTED SYSTEM

Figures 44 and 45 present visual results of the experiment, demonstrating the robot's ability to attain seated and standing from different view.



(a) Seated Position. (b) Standing Position.

Figure 44: Side View.



(a) Seated Position. (b) Standing Position.

Figure 45: Front View.

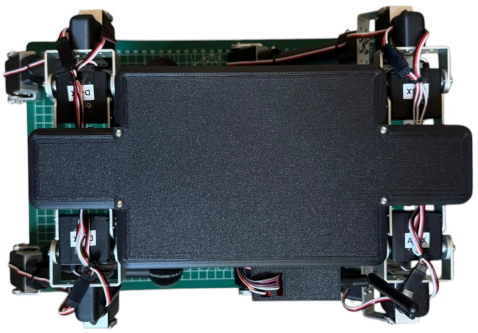


Figure 46: Top View.

8.2 RESULTS

Overall, the project has successfully achieved its objective: the robot moves through space, and the gyroscope serves as an excellent aid for the system's static stability.
The movements are smooth and stable in terms of balance, while the body and legs are highly robust, maintaining a reasonable level of lightness due to the use of carbon fiber. However, the control system represents the most significant improvement compared to previous versions of the robot, enabling highly accurate and precise movement control.
Video of the Robot on YouTube: All Movements, Standing-Sitting.
(https://youtube.com/shorts/rValXnHnFNA,
https://youtube.com/shorts/EczIbvVzH1g).

8.3 TECHNICAL SPECIFICATIONS

Table 3: Specifications

Dimensions	**Locomotion**	**Environment**
Height: 250 mm	Speed: 0,1 m/s	Op. Temp: 10°C to 30°C
Width: 285 mm	Max Load(Static): 4,5-5 kg	Ingress Protection: IP10
Length: 460 mm	Max Load(Dynamic): <0,5 kg	
Standing Height: 367 mm	Max Slope ±10°	
Weight: 3 kg		

Note: The indicated values may vary depending on the battery charge level.

9

CONCLUSIONS AND FUTURE WORK

Several critical aspects could be improved.
One of the main limitations is the type of motors used. While servomotors provide high precision and torque, they have significant constraints in terms of speed and continuous motion. As a result, the robot is limited in performing more dynamic and turbulent movements, such as jumping, and has a relatively low movement speed.
Another constraint is the available space within the robot's body, which is just sufficient for the necessary components and does not allow for the addition of extra elements.
The most critical issue, however, is the robot's vision. The system is essentially "blind", relying only on the gyroscope, which provides information about the inclination of the robot's body and acceleration data that are not particularly useful for this project. The gyroscope alone is insufficient to help the robot orient itself in space.
A fundamental improvement would be the integration of a LiDAR sensor, which would enable the robot to map its surroundings, avoid obstacles, and understand the terrain's morphology. This capability is essential for adapting the gait to irregular surfaces or obstacles in real-time.
Another major limitation is the control system. Advanced quadruped robots developed by universities and companies utilize sophisticated machine learning algorithms that allow them to dynamically adapt their gait to the environment. This feature is crucial for making the robot more versatile and capable of operating in complex environments.
Despite these limitations, all these issues can be addressed in a future version of the quadruped robot without major difficulties.

BIBLIOGRAPHY

[1] Arduino. Arduino mega 2560. https://store.arduino.cc/products/arduino-mega-2560-rev3.

[2] Priyaranjan Biswal and Prases K. Mohanty. Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, 12(2):2017–2031, 2021. ISSN 2090-4479.

[3] Hui Chai, Yibin Li, Rui Song, Guoteng Zhang, Qin Zhang, Song Liu, Jinmian Hou, Yaxian Xin, Ming Yuan, Guoxuan Zhang, and Zhiyuan Yang. A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach. *Biomimetic Intelligence and Robotics*, 2 (1):100029, 2022. ISSN 2667-3797.

[4] Boston Dyamics. Spot. https://bostondynamics.com/products/spot/.

[5] Yanan Fan, Zhongcai Pei, Chen Wang, Meng Li, Zhiyong Tang, and Qinghua Liu. A review of quadruped robots: Structure, control, and autonomous motion. *Advanced Intelligent Systems*, 6(6): 2300783, 2024.

[6] Raspberry. Raspberry pi 4 b. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[7] Muhammed Arif Şen, Veli Bakırcıoğlu, and Mete Kalyoncu. Inverse kinematic analysis of a quadruped robot. *International Journal of Scientific Technology Research*, 6, 10 2017.