Polydog – Quadruped Robot, with a Three-degree-of-freedom Parallelogram System

Hugo Durand, Younes Bazi, Raphaël Anjou

***Abstract*—This article presents the development and implementation of a robot dog called Polydog. Several versions have been created, according to our vision and understanding of the robot. It's equipped of a motor driver SSC32 powering 12 servomotors TD8135MG controlled by a Esplora remote. The article highlights the process of a dog robot's development and the limitations of electronics components and of some methods.**

***Index Terms*—Robot dog, legged robot, parallelogram system, Arduino, servomotor, PlateformIO, inverse kinematics, Coppeliasim simulation.**

I. Introduction

THE use of robot dogs, also known as quadruped robots, offers several advantages over traditional wheeled robots in difficult terrain. It is known that's a subject of research and competition between large companies such as Boston Dynamics [1] or Unitree Robotics [2] which invest in this kind of project because they see in it an incredible automation of tasks in the future. They can be trained to perform a variety of tasks, such as assisting people with disabilities, perform boring or dangerous tasks for humans, and even serving as a security measure. On our scale we aim to develop low-cost, DIY and an open-source version that can be built in any fablab, without any increased computer knowledge. The advantage of their lower cost is that they are more easily inclined to be sent for small valuable missions, such as reconnaissance in the army without fear of a big loss.

By focusing on a low-cost, DIY, and open-source approach, our project aims to contribute to the democratization of robot dog technology, making it more accessible for individuals and smaller organizations to explore its vast potential and spur further innovation.

The rest of this article is organized as follows. In Section II, the designed research of a leg and the construction of the robot. In Section III, the electronic components used inside the Polydog. In Section IV, we present the algorithm to bring it to life. In Section V, the methods for calculating the positions and rotations of our joint model are described. In Section VI, we will discuss the simulation of the robot on the CoppeliaSim software. In the final Section, the conclusions are presented.

H. A. N. Durand, University Côte d'Azur, Robotics Department, Polytech'Nice, 930 Route des Colles, 06410 Biot (e-mail: hugo.durand@etu.univ-cotedazur.fr).

Y. Bazi, University Côte d'Azur, Robotics Department, Polytech'Nice, 930 Route des Colles, 06410 Biot (e-mail: younes.bazi@etu.univ-cotedazur.fr).

R. Anjou, University Côte d'Azur, Computer Science Department, Polytech'Nice, 930 Route des Colles, 06410 Biot (e-mail: raphael.anjou@etu.univ-cotedazur.fr).

II. Building the robot

A. Parallelogram system

Based on Nina Schaller's research on the walking of an ostrich and the bones composition [3], we opted for a modeling of the leg integrating the parallelogram system which allows to keep the mass of the leg closer to the body and limit the inertia of the leg.

The parallelogram configuration of the robot dog leg involves four links or bars connected by pivot points, creating a parallelogram shape. In this configuration, two servo motors are employed to control the movement of the upper and lower parts of the leg. Figure 1 provides a visual representation of this configuration.

In terms of the body construction, two solutions were tested: 3D printing and aluminum. Initially, a 3D printed body was used, but it was later replaced with an aluminum body. While the weight of the body did not decrease with the aluminum construction, it offered significant improvements in terms of strength and durability.

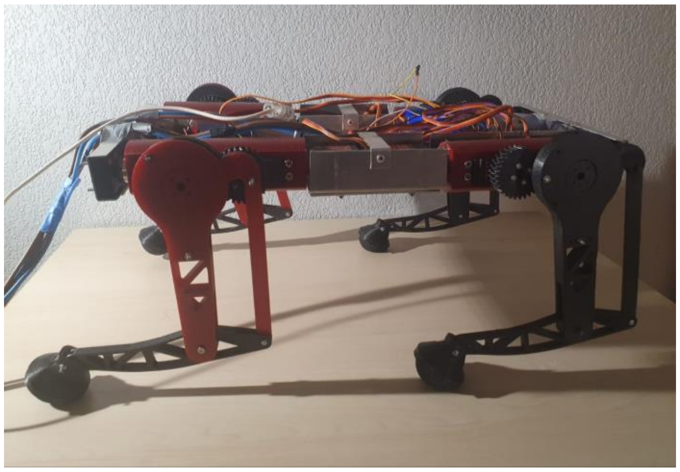

Fig. 1. Legs with 3 degrees of freedom. A servo fixed to the body rotates the cylinder attached to the other end by a ball bearing; the second one moves the upper part of the leg back and forth through the gear and the last one controls the lower part with the parallelogram system.

We have tested several types of gears. In the first robot version [4], we tried spur & helical gears, modifying the module (reduction of the pitch respecting the precision limits of a 3D printer nozzle) and the gear ratios (increasing the output torque while keeping a sufficient amplitude of movement). In the latest version, we opted for herringbone gear [5] with a 1.3 mm module and 30 teeth sprockets. In comparison to single-helical gears, herringbone gears offer notable advantages, including a higher load carrying capacity, a larger total contact ratio, and a reduced axial force. We also tried to reduce the backlash between the servo motor housing and the servo horn, with our own hand spinner shaped servo horn, printed in Greypro resin and Draft_v2. However, they were not strong, and the backlash was not completely erased due to the internal gear backlash of the servo motor.

B. *Modeling and printing*

We modeled the whole animal on the Onshape software (used in Peip1 course), facilitating the management of the links between the 8 designed parts per leg (3D printing, PETG, 20% filling), the body (made first in 3d printing and then in aluminum) and the servo horns.

The feet used to be made of ninja flex filament, with a rubber coating, but now they are made of squash ball and have a better grip on the ground. It is easier for the inverse kinematics calculations to have a circular foot rather than an unconventional foot shape.

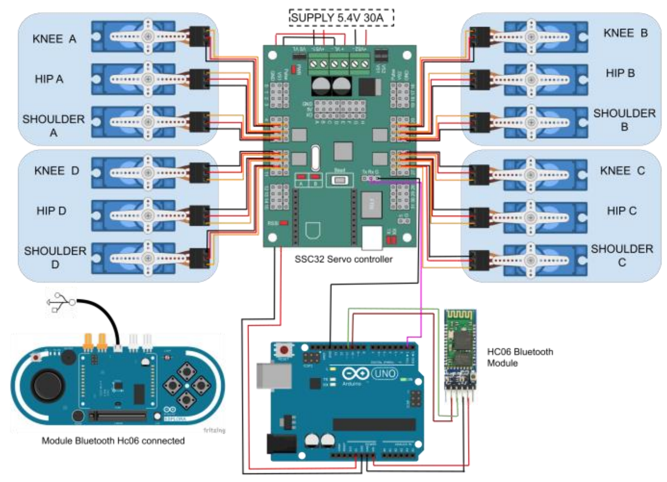

Fig. 2. Electronic diagram of the robot. This allows to give a global vision of the whole and to observe the links between the different components. This has been designed on Draw.IO [6].

III. ELECTRONIC COMPONENTS

A. *The central control unit*

The Polydog uses an Arduino Uno microcontroller as its central processing unit, which allows for the integration of various sensors and actuators. Chosen for the moment for its simplicity of use and its pre-programmed modules. It is however limited in the number of serial communication, as we only have two Rx Tx I/O, we used the SoftwareSerial library replicating the same functionality on other pins.

B. *Motors and driver*

The robot dog is equipped with 12 TD8135MG servo motors 35kg/cm of torque, which are controlled by a motor driver SSC 32. The reading of the user's guide was essential for us to understand the functioning of the card, especially the power options (VS1, VS2, VL). We supply the logic circuit VL with the Arduino board and we connect the two power channels VS1 and VS2 under 5.4 V for the servo motors with a power supply providing up to 50A [7].

After noticing momentary voltage drops, to which the addition of capacitors was not enough to store enough electrical energy to overcome the electrical resistance of the cable. By increasing the cross-section of the power cable and manually changing the output voltage of the power supply, we solved the servo operation problems.

C. *Esplora board controller*

Finally, we used an Esplora remote to control the Polydog, which was equipped with the Bluetooth module Hc06 (Slave). The module Bluetooth Hc05 (Master) [7] connected to the Arduino, via software pins 10 & 11.

The robotic dog was programmed to respond to various inputs from the Esplora controller, such as button presses, joystick movements, and sensor readings. For example, we can tilt the body of the robot back and forth, to one side or the other, following the values of the accelerometer on the remote.

IV. ALGORITHMS

The algorithm employed initiates with the transmission of commands to the SSC32 servo controller through serial communication. These commands are sent as string data in a specific format, namely "#<motor_pin>P<angle between 500-2500 microseconds>T<time in milliseconds>." The Arduino Uno microcontroller is responsible for sending these commands, which are then converted into control signals for the servomotors.

To facilitate the management of command transmission, a custom class named CustomServo was implemented. This class encapsulates the necessary functions for sending commands to the servo controller, ensuring efficient and organized control of the servomotors. By leveraging the functionalities of this class, we gain the flexibility to specify the angle unit of the command, allowing for the choice between degrees or micro-seconds for more precise control over the servo motors.

In addition, a Leg class was developed to instantiate each leg of the robot dog. This class serves multiple purposes, including the correction of imprecisions and offsets between the legs. By applying appropriate adjustments, the Leg class ensures that the four legs of the robot form a stable rectangular configuration on a table surface. This step is crucial for maintaining stability and balance during locomotion.

Furthermore, the PolyDog class has been designed to record empirically constructed movement patterns. These patterns, such as trot_walk() and stand_up(), consist of a sequence of position commands. Each position command specifies the desired movement for a particular leg, usually in the form "legA.move_hip(95)". By storing these movement patterns, the PolyDog class reduce the amount of code in the main function.

The development process of the robot dog utilized PlatformIO, an open-source Integrated Development Environment (IDE) specifically designed for embedded systems. PlatformIO offers a range of features that enhance the development workflow, including comprehensive dependency management. This feature simplifies the installation of necessary libraries, such as Esplora, which are crucial for interfacing with the hardware components of the robot dog. Additionally, PlatformIO provides an integrated debugger, facilitating the identification and resolution of software bugs during the development phase.

V. INVERSE KINEMATICS

A. *In search of angles*

Inverse kinematics (IK) is the mathematical process of computing motions that achieve a given set of *tasks*, such as placing a foot on a surface or moving the center of mass (CoM) by calculating the variable joint parameters required to place the end of the kinematic chain.

The parallelogram system makes the calculation of the angles of the two central servo motors of the leg more complex since it is a question of finding the angle between the tendon and the tibia that has been actuated by the knee servo indirectly (joint A on the Figure 3).

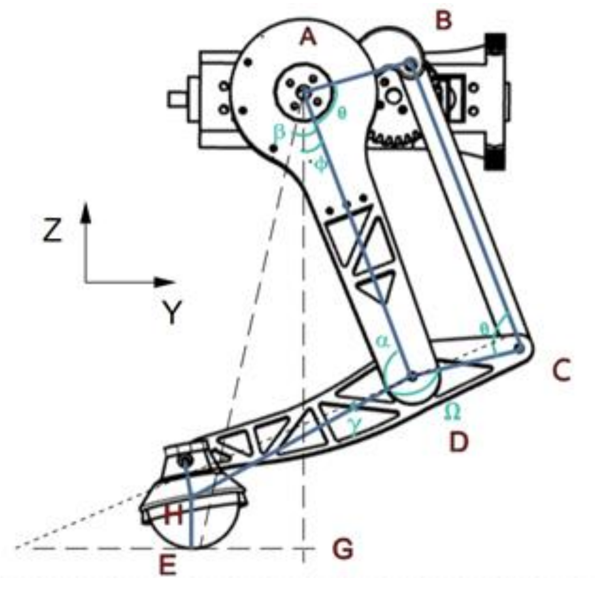

Fig. 3. Modeling of the leg by segments between the pivot links allowing the visualization of triangles, points, and different configurations.

By providing input parameters such as the desired lifting height, movement extent, and speed, the algorithm is capable of autonomously controlling the leg's motion in the specified direction. This functionality is achieved through a straightforward function call, allowing for seamless integration and intuitive leg movement control.

B. *Joints and coordinates calculation*

We started by representing the leg of the robot by drawing a technical diagram of the assembly on Onshape. By importing it on the EdrawMax software, we represented the angles and the stick structure of the leg.

It was crucial to represent the dependencies between the different parts, especially to reveal the gamma angle, essential in the calculations.

With the help of figure 3, we obtain according to the Al Kashi's theorem:

$$CH^2 = CD^2 + DH^2 - 2 \times DH \times CD \times \cos(\Omega) \text{ So } \Omega = cos^{-1}\left(\frac{DE^2+CD^2-CE^2}{2\times CD\times DE}\right)$$

(Valid for both configurations) $$\gamma = \pi - \left(\frac{\pi}{2} - \theta\right) - \Omega = \left(\frac{\pi}{2} + \theta\right) - \Omega \qquad (1)$$

$$\alpha = \frac{\pi}{2} + \gamma \qquad (\gamma \text{ is negative or positive}) \qquad (2)$$

Knowing α, I can get the AH to find EG and AG:

$$AH = \sqrt{DA^2 + DH^2 - 2 \times DA \times DH \times \cos(\alpha)} \qquad \Phi = cos^{-1}\left(\frac{-DH^2 + DA^2 + AH^2}{2 \times DA \times AH}\right)$$

$$\begin{cases} EG = \sin\left(abs(\beta - \Phi)\right) \times AH \\ AG = \cos\left(\text{abs}(\beta - \Phi)\right) \times AH + HE \end{cases} \qquad (3)$$

We define hip_angle = Ħ and knee_angle = κ
Conversely, with the coordinates of the leg AE, we have:

$$HA = \sqrt{EG^2 + (AG - HE)^2} \qquad \varepsilon = cos^{-1}\left(\frac{(AG-HE)}{AH}\right) \qquad (4)$$

$$\Phi = cos^{-1}\left(\frac{-DH^2 + DA^2 + AH^2}{2 \times DA \times AH}\right) \qquad \alpha = cos^{-1}\left(\frac{DH^2 + DA^2 - AH^2}{2 \times DA \times DH}\right)$$

- $\text{Ħ} = \frac{\pi}{2} - \Phi - \varepsilon \; or \; \text{Ħ} = \frac{\pi}{2} - \Phi + \varepsilon \qquad (5)$

  Depends on if the leg is in front or behind the hip

- $\kappa = \theta - \text{Ħ}$

  $\kappa = \Omega - \frac{\pi}{2} - \text{Ħ} + \gamma \quad or \quad \kappa = \Omega - \frac{\pi}{2} - \text{Ħ} - \gamma$

  $\kappa = \Omega + \alpha - \pi - \text{Ħ} \quad or \quad \kappa = \Omega - \alpha - \text{Ħ} \qquad (6)$

The previous joint calculation (5), (6) is placed in a simplified configuration with the shoulder perpendicular to the ground and with a positive knee angle.

By adding a function to PolyDog, it can hold itself in a position by inputting the bust height and the distance of the foot from the hip axis. We wrote two functions f(coordinates) and f (joints angles) to check if the result is correct.

VI. Simulation

CoppeliaSim [8] was selected as the simulation platform for our dog robot due to its comprehensive features, realistic physics engine, and user-friendly interface.

The robot was programmed using the Lua language, enabling it to perform various movements within the simulated environment. The implemented functionalities include forward walking, backward walking, sideways walking, turning around itself, and executing in-place maneuvers such as yawing, pitching, and rolling. To ensure the robot's stability and balance during its movements, we integrated a PID (Proportional-Integral-Derivative) control system into its operation. This control system allowed the robot to make rapid adjustments, effectively maintaining its equilibrium even when faced with unexpected conditions.

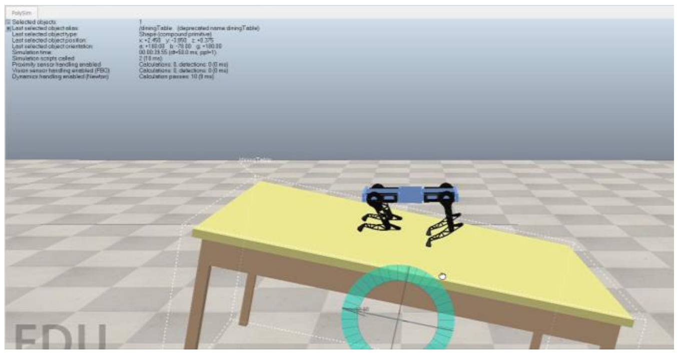

Fig. 4. The Polydog maintaining a horizontal body position while rotating the table due to the implementation of the PID control system.

Additionally, we successfully programmed the robot in the simulation to respond to the Arduino Esplora controller. Consequently, the user can now control the robot using the remote controller, enhancing accessibility and user-friendliness of the system.

Finally, the simulation played a crucial role in our research as it served as a testing ground for our code before implementing it on the physical robot. This step was invaluable in identifying and rectifying any potential issues or errors, ultimately contributing to the successful deployment and performance of the robot. The simulation not only provided a safe and controlled environment for code validation but also accelerated the development process, reducing risks and costs associated with direct experimentation on the physical platform.

VII. Conclusion

The development of a robot dog represents a major technical challenge, particularly in terms of stability and biomimetic locomotion. The progression from maintaining balance on four legs to the transition to three and finally two legs, mirroring the locomotion of four-legged animals, is particularly difficult from our point of view. Although the current iteration of the Polydog is able to perform walking movements, but it still wimps a little.

For the moment, we've built the robot without testing the real strength of the servomotors and their power in relation to the weight they must support at each corner of the body. We think the next step is to go back to the design of the leg and determine the weight of the body at the four points of the hip to better choose the servo motors or the transmission of mechanical force from the servo motor head to the force transmitted to the ground by the rubber foot.

We'll also soon be able to retrieve the right angles from the simulation and transmit them to the Arduino, so we'll be able to render the robot's movements in a way that's more in line with our expectations, and control them more finely.

References

[1] Boston Dynamics official website: https://www.bostondynamics.com/
[2] Unitree Robotics official website: https://www.unitree.com/en/
[3] N. Schaller, “Birds on the run: what makes ostriches so fast?” in the in the scienceinschool.org website, November 22, 2011
[4] R. Anjou, H. Durand, Polydog_version1 in Instructables, a website specialized in DIY projects, https://www.instructables.com/Polydog-Arduino-Project/
[5] R Keith Mobley, 39 - Gears and Gearboxes, in Plant Engineer's Handbook, 2001
[6] Gaudenz Alder, Draw. Io, diagrams.net, a free and open-source cross-platform graphic drawing software developed in HTML5 and JavaScript.
[7] RF”, http://users.polytech.unice.fr/~pmasson/Enseignement.htm
[8] CoppeliaSim user manual: https://www.coppeliarobotics.com/helpFiles/

Hugo Durand received a high school diploma from the Lycée Paul Valery, in Cali, Colombia. Since 2020, he’s student at Polytech Nice Sophia, currently in the Robotics Department.

Younes Bazi, received a high school diploma from the H.S the Residence, in Casablanca, Morocco. and he’s currently in the Robotics Department at Polytech Nice Sophia.

Raphaël Anjou, currently a student in Computer Science at Polytech in a work-study program at Orange Business Service. He is one of the co-designers, building the first version of PolyDog.