Design, Manufacture and Control of an Inverted RRS Delta Robot

Carlos E. Contreras Martínez, Angel D. González Jasso
A01754923@tec.mx, A01747954@tec.mx
Instituto Tecnológico y de Estudios
Superiores de Monterrey

***Abstract*—This report presents the design, manufacture, programming, and control of an inverted Delta robot developed as a final project. The main objectives of the robot was to balance a ball on a mobile platform and also being capable of supporting a load of up to one kilogram. The system integrates mechanical design, embedded programming, and control strategies to achieve motion. The control system uses feedback from cameras to get the position of the ball and the inverse kinematics model corrects the platform's orientation to maintain the ball's position. Experimental results demonstrate the robot's effectiveness in achieving its goals, validating the proposed design and control approach.**

***Index Terms*—Delta robot, Parallel manipulator, Kinematics, Control, Robotics.**

I. Introduction

A Delta robot is a type of parallel manipulator known for its high speed, precision, and structural rigidity. Unlike traditional serial robots, delta robots consist of a fixed base and a mobile platform connected by three arms, usually arranged in a triangular configuration. These arms operate in parallel, which allows for quick and dynamic motions. The structure of the robot is particularly suitable for pick-and-place tasks, sorting systems, and operations requiring delicate positioning at high frequency.

The origin of the Delta robot takes back to the early 1980s, when Professor Reymond Clavel of the École Polytechnique Fédérale de Lausanne (EPFL) proposed a new parallel kinematic mechanism in his doctoral thesis. His aim was to develop a manipulator capable of performing fast and accurate motions. This concept came to life in the early 1990s when the Swiss company Demaurex implemented the Delta robot in industrial settings, focusing on high-speed pick-and-place operations. [1]

The basic design of a Delta parallel robot uses either revolute motors, linear actuators or a combination of them; these have a fixed orientation with respect to an input link. The most common configuration includes three actuators, three degrees of freedom. However, the existence of other parallel robots with more DOF have introduced new research avenues.

The specific problem addressed is the design and integration of a mechanical structure, embedded control system, and feedback sensors to maintain the ball's position on a moving platform. The control strategy uses computer vision and processing, enabling the robot to react to external disturbances and preserve stability. A key component of the system is the implementation of inverse kinematics to compute the necessary joint angles to correct the platform's orientation.

Fig. 1. ABB Delta Robot (IRB 360 FlexPicker)

Recent research in parallel manipulators and robotic control systems highlights the potential of such platforms for tasks requiring high accuracy and responsiveness. The system developed integrates insights from different studies and practical constraints, implementing a control architecture based on the ROS, two ESP32 microcontrollers, and Jetson Nano for image processing.

The rest of this article is structured as follows: Section II presents the theoretical framework behind the key components of the system; Section III details the development process including design, sensors, and control implementation; Section IV discusses the experimental results; and Section V provides conclusions and future work suggestions.

II. Theoretical Framework

A. *PID Controller*

A Proportional, Integrative, Derivative (PID) controller is a widely used feedback control algorithm in robotics and automation projects. It consists on calculating an error from the difference between a desired reference and a feedback signal. This applies a correction based on the three elements of PID

- Proportional. Reacts to the present error.
- Integral. Reacts to the accumulation of past errors.
- Derivative. Reacts to the rate of change of the error.

The control signal looks like this:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

Where:

- $u(t)$: Control signal.
- $e(t)$: Error signal,
- K_p: Proportional gain.
- K_i: Integral gain.
- K_d: Derivative gain.

This control signal can then be used to control actuators.

B. *Motor driver*

A motor driver acts as a power stage between low-power control signals and high-power motors, performing three key roles.

- Amplification. Takes the control signals from a microcontroller and amplifies them to power the motor.
- Direction. Using H-bridge circuits, a motor driver can set the direction of rotation.
- Speed control. Usually using Pulse Width Modulation (PWM) signals, modulates the motor speed by changing duty cycles

C. *Encoder*

An encoder is a sensor used to measure the position, speed, or direction of a rotating shaft. There are two main types of encoders,

- Incremental Encoders These generate pulses as the shaft rotates, providing a way to measure position and speed.
- Quadrature Encoders These generate two-phase outputs, A and B, that are slightly shifted. This enables the use logic to measure position, speed, and direction.

Some encoders also come with a Z phase that emits a pulse when a full rotation of the shaft has been reached.

D. *Microcontroller*

In mechatronic systems, a microcontroller is the central unit that is in charge of:

- Reading sensor data.
- Executing the control system.
- Sending control signals to the power phase.
- Interfacing with other components, such as buttons.
- Communicating with external systems using different types of communication, like UART, I2C or Wi-Fi

E. *ROS*

Robotic Operating System (ROS) is a framework for developing robot software, facilitating modular design by using:

- Nodes.
  These are executable programs that perform tasks, such as vision processing or motor control.
- Topics.
  These act as channels for unidirectional communication. The publisher-subscriber model.
- Services.
  These offer request-response communication between nodes.
- Actions. Used for long-lasting tasks. These offer the possibility to receive feedback while the task is being completed and allows cancellation of it midway.

III. Development

A. *System Overview*

The robot that will be developed is an inverted delta manipulator designed to balance a ball on a flat platform. The system makes quick movements to compensate for the ball's position and maintain it on a target point or trajectory. The mechanical structure is composed of three arms connected to DC motors, each arm is connected to the platform with a spherical joint, and at the elbow has a revolute joint.

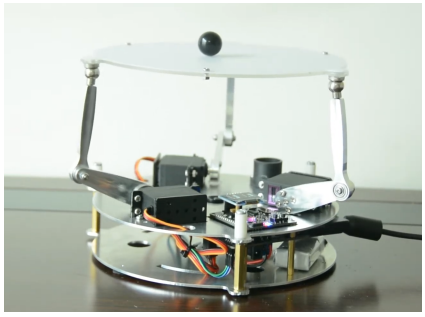

Fig. 2. Inverse RRS Delta

The electrical components include the DC motors responsible for positioning the platform, as well as a pair of cameras that detect the ball's position. The image processing will be made by a Jetson NANO board. These components are integrated with a control system based on two ESP32 microcontrollers.

B. *Datasheets*

In the following link, the datasheets of the components of the project were gathered: Link

C. *Sensor Analysis*

- Encoders:
  The ESP32 was programmed to read pulses from the encoder's A and B phases using interrupts. By detecting changes in both signals, the board did full quadrature reading to estimate the position and determine the direction of rotation for each motor.
- Potentiometers:
  A potentiometer was added to each motor to determine its initial position at startup, since the encoders used were relative rather than absolute. (Fig. 4)
- Cameras:

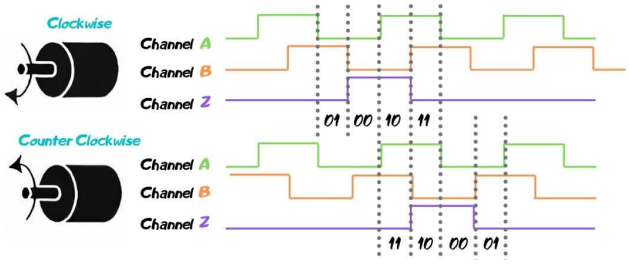

Fig. 3. Full quadrature encoder reading.

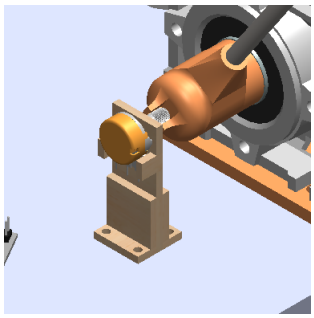

Fig. 4. Potentiometer-Motor coupling.

A Python program was developed to detect the position of the ball using two digital cameras: one positioned laterally and the other placed underneath. (Fig. 5)

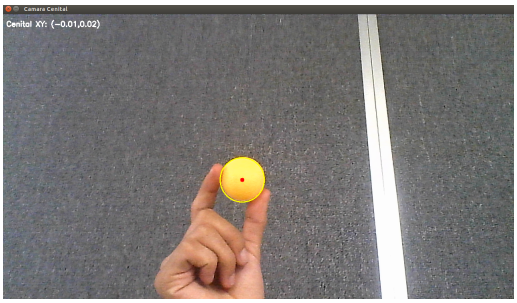

Fig. 5. Bottom camera reading.

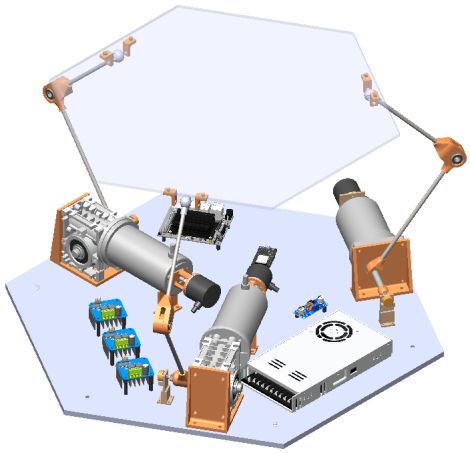

Fig. 6. Final CAD design

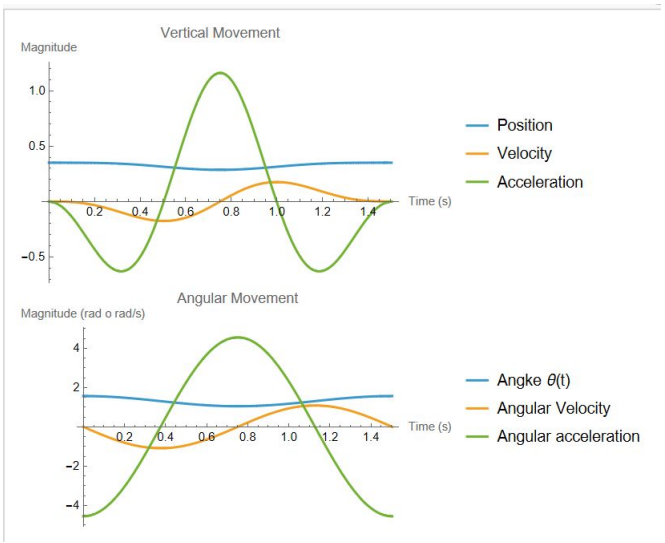

Fig. 7. Motion analysis of the moving platform. Vertical movement (top) shown in meters (m), meters per second (m/s), and meters per second squared (m/s²). Angular movement (bottom) shown in radians (rad), radians per second (rad/s), and radians per second squared (rad/s²).

D. *Design Process*

The mechanical design of the delta robot was developed focusing on obtaining stability and efficiency in the motion of the robot. In Fig 6 the final CAD design is presented.

E. *Slider Crank Analysis*

While in the design phase, a motion analysis was conducted to calculate the position, velocity, and acceleration of the platform in relation to the angular movement of the motors (Fig 7). This analysis allowed the determination of the best link lengths to maximize the platform's acceleration. The lengths were set to $l_1 = 25cm$ and $l_2 = 20cm$, achieving the highest possible response.

F. *Singularities*

Due to design restrictions, there are two singularity points in the robot, positions in which the robot cannot transmit motion to the end effector. One of these singularities happens when the lower link of the arm is perpendicular to the base (Fig. 8). The other singularity happens when the upper link of the arm is parallel to the base (Fig. 9). With a geometry analysis the angles of these singularities are obtained as $\theta_{s1} = 36.86$ & $\theta_{s2} = 90$.

G. *ROS Implementation*

The main objective of the robot is to balance a ball on a platform. ROS (Robot Operating System) will be used to organize the different parts of the system, like reading the cameras, processing the images, controlling the robot, and sending commands to the microcontroller. ROS will help us make all the nodes communicate efficiently and also make the system modular. Since Ubuntu 24.04 is not supported yet in the Jetson Nano, it was flashed with Ubuntu 18.04. The ROS distribution for that OS is ROS Melodic. The architecture

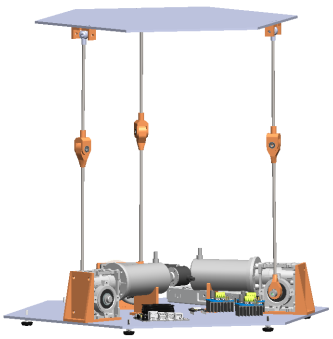

Fig. 8. Singularity 1: Arm links parallel to each other

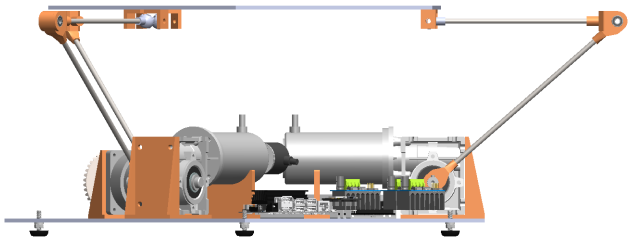

Fig. 9. Singularity 2: Upper links parallel to platform

established in Table I and figure [10] were proposed for the project.

H. *Inverse Kinematics*

The inverse kinematics of a robot refer to the process of obtaining the joint angles necessary to obtain the set of coordinates of the end effector. In this case, the end effector is the platform, by defining a desired position for the platform, each motor position has to be obtained.

4.1. Manipulator Geometry

The 3-RRS Parallel Manipulator consists of a fixed base, a moving platform and 3 identical limbs connecting the base and the platform. The limbs lie on separate planes. Each limb plane is normal to the parallel R joint axes and passes through the S joint centers associated with the limb. Each limb is composed of three joints:

- an active R joint connected to the base represented by points O_{0i} on the limb plane,
- a passive R joint between upper and lower limbs represented by points O_{ij} on the limb plane,
- a S joint between upper limb and platform represented by points for $i = 1, 2, 3$ and $j = i + 3$.

In Figure 11, the fixed coordinate frame $O_0{-}xyz$ is attached on the base, where the origin O_0 is at the center of the three fixed revolute joints. The radius of the base circle is b. The vectors from O_0 to O_{0i} are $\vec{b}_i$ and the x-axis is chosen to be along $\vec{b}_1$. The angle between $\vec{b}_1$and $\vec{b}_2$ is $\alpha_{12} = 120°$. The angle between $\vec{b}_1$ and $\vec{b}_3$ is $\alpha_{13} = 240°$.

A coordinate frame $O_7 - uvw$ is attached on the moving platform, where the origin O_7 is the center of the three S joints O_{7j} for $j = 4, 5, 6$. The radius of the platform circle is p. The vectors from O_7 to O_{7j} are $\vec{p}_j$. The u-axis is chosen to be along $\vec{p}_4$. The angle between $\vec{p}_4$ and $\vec{p}_5$ is $\alpha_{45} = 120°$. The angle between $\vec{p}_4$ and $\vec{p}_6$ is $\alpha_{46} = 240°$.

The position vector that defines the location of the platform with respect to the fixed coordinate frame is $\vec{r}_P = \overrightarrow{O_0O_7}$. $\vec{r}_i = \overrightarrow{O_{0i}O_{ij}}$ are the lower limb vectors and $\vec{r}_j = \overrightarrow{O_{ij}O_{7j}}$ are the upper limb vectors.

4.3. Inverse Kinematics

From ref. [3], the equation that solves for the angles of the active revolute joint is:

$$\theta_i = 2\tan^{-1}\left(\frac{-B_i \pm \sqrt{{A_i}^2 + {B_i}^2 - {C_i}^2}}{C_i - A_i}\right) \tag{2}$$

where:

$$A_i = 2l_1 c\alpha_{1i}\left(bc\alpha_{1i} - O_{7j,x}\right) \tag{3}$$

$$B_i = 2l_1 O_{7j,z} c^2\alpha_{1i} \tag{4}$$

$$C_i = {O_{7j,x}}^2 - 2bO_{7j,z}c\alpha_{1i} + c^2\alpha_{1i}\left(b^2 + {l_1}^2 - {l_2}^2 + {O_{7j,z}}^2\right) \tag{5}$$

IV. Results

The final assembly of the robot is shown in Fig (12):

In the following link all the programming involved in the project is stored in a github repository: Link

In the following link there is a demostration video of the final prototype: Link

During testing, the robot was not able to fully achieve the goal of keeping the ball balanced on the platform. While the control system, sensors, and communication between modules worked as expected, the main issue was mechanical because the structure wasn't rigid enough to hold the platform in position when the motors were active.

In addition, the arms were designed to be relatively long to improve acceleration and range of motion, but this ended up reducing stability. The platform tended to wobble and lose alignment, which made it hard for the control system to make accurate corrections.

V. Conclusions

The project demonstrated the integration of PID control based on image processing, inverse kinematics, and embedded communication using ROS and microcontrollers. However, the mechanical limitations of the prototype prevented the full validation of the control strategy. The selected DC motors and drivers provided sufficient torque and speed, but the lack of structural rigidity and the extended length of the robot's arms caused platform instability, affecting the performance.

These issues highlighted the importance of balancing mechanical design with control precision. For future iterations, shorter and stiffer links are recommended, along with a reinforced frame to ensure consistent platform positioning. Although the original objective of maintaining the ball in position was not fully met, the project successfully validated key components of the system and provides a great foundation for future improvements.

TABLE I
COMPONENTS OF THE DELTA ROBOT SYSTEM

Node	Function	Purpose	Publishes to...	Subscribes to...
/cameras	Sensing & Processing	It calculates the current ball position, based on the image of the cameras.	/position_measurement	–
/reference	Trajectory	It publishes the desired position reference of the platform, the different trajectories of the ball are selected by publishing on the /select_reference topic.	/reference	/select_reference
/control	Control	It calculates the new desired platform position based on the current reference and measurement.	/platform_position	/position_measurement /reference
/IK	Processing	Converts the desired platform position into the joint angles of the robot, with the inverse kinematics.	/joint_angle	/platform_position
/UART	Communication	Receives the joint angles and sends them to the esp32 via UART communication.	–	/joint_angle

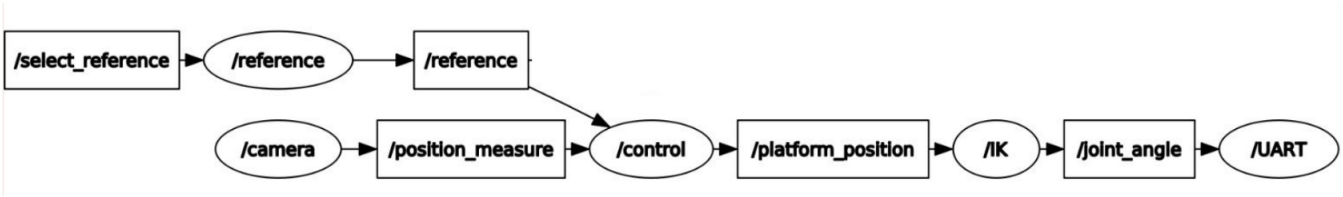

Fig. 10. ROS diagram of nodes (ovals) and topics (squares).

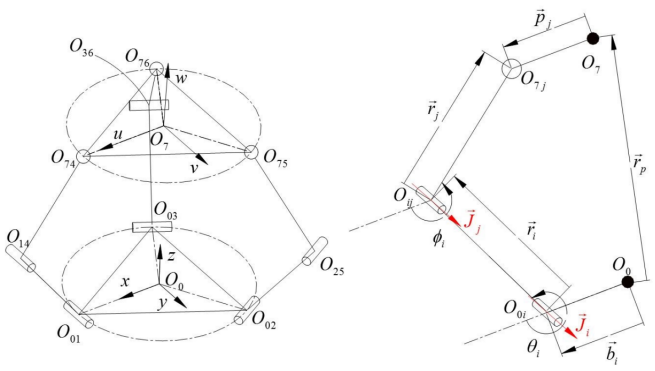

Fig. 11. Kinematic Diagram of the 3-RRS PM.

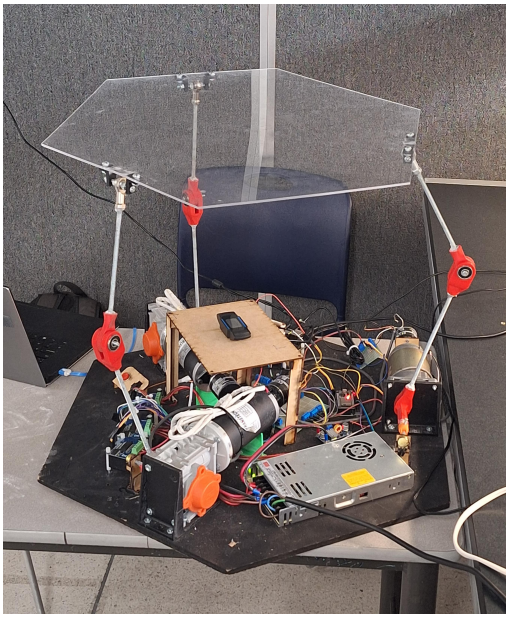

Fig. 12. Final assembly.

REFERENCES

[1] A. Gasparetto and L. Scalera, "From the Unimate to the Delta Robot: The Early Decades of Industrial Robotics," in Explorations in the History and Heritage of Machines and Mechanisms: Proceedings of the 2018 HMM IFToMM Symposium on History of Machines and Mechanisms, B. Zhang and M. Ceccarelli, Eds. Cham, Switzerland: Springer, 2019, pp. 284–295.

[2] E. Artuc, P. Bastos, A. Copestake, y B. Rijkers, "Robots and Trade: Implications for Developing Countries," en Robots and AI: A New Economic Era, G. M. Grossman y L. Y. Ing, Eds. Oxon and New York: Routledge, 2022, p. 240.

[3] H. Tetik, Modelling and Control of a 3-RRS Parallel Manipulator, M.S. thesis, Dept. Mech. Eng., İzmir Institute of Technology, İzmir, Turkey, 2016.

[4] Dynapar, "Encoder Basics," Dynapar, [Online]. Available: https://www.dynapar.com/knowledge/encoder-basics. [2025].

[5] ID Explained, "PID Controller Explained," PID Explained, [Online]. Available: https://pidexplained.com/pid-controller-explained/. [2025].

[6] echTarget, "What is a microcontroller?," IoT Agenda, [Online]. Available: https://www.techtarget.com/iotagenda/definition/microcontroller. 2025].

[7] eeksforGeeks, "Introduction to ROS (Robot Operating System)," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/introduction-to-ros-robot-operating-system/. [2025].