Universidad Tecnológica de Pereira

Ecosistema de simulación y control dinámico de un manipulador robótico de 6 GDL

Soberanía tecnológica con software libre (Python & PyBullet)

Elias Escobar Pereira

Universidad Tecnológica de Pereira
Semillero "Huellas sobre Sistemas Digitales y Mantenibilidad"
GIGSEEA

7 de Mayo, 2026

Manipuladores robóticos y grados de libertad

¿Qué es un manipulador robótico? Es una *cadena cinemática* de eslabones rígidos unidos por articulaciones. Tareas de alta repetitividad, alta precisión y condiciones adversas.

¿Por qué 6 GDL? (Estándar Industrial)

- **3 GDL de Posición:** Alcanzan cualquier punto (X, Y, Z).
- **3 GDL de Orientación:** Definen la actitud (*Roll, Pitch, Yaw*).

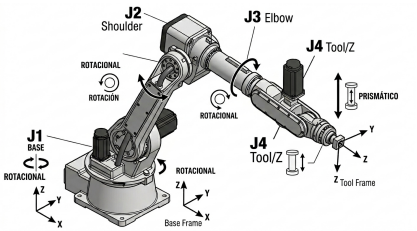

Figura 1: Estructura de un brazo de 4 GDL.

Analogía humana

El brazo humano tiene 7 GDL. Los 6 GDL son la cantidad mínima para dominar cualquier pose en el espacio 3D.

Cinemática

El lenguaje del movimiento

- **Cinemática directa:** Dados los ángulos motor, calcula (X, Y, Z, R, P, Y) del efector final.
- **Cinemática inversa:** Dados (X, Y, Z) deseados, determina los ángulos necesarios.

Figura 2: Manipulador PAROL 6.

Dato curioso

La cinemática inversa es mucho más compleja, ya que para un mismo punto el robot puede tener varias formas de llegar.

¿Por qué surge este proyecto?

El costo prohibitivo del software privativo

Plataforma	Tipo	Costo (Aprox.)
RoboDK	Privativa	$4,990 USD
MATLAB + Robotics TB	Privativa	$2,150 USD/año
ABB RobotStudio	Privativa	$3,200 USD/año
Siemens NX	Privativa	>$20,000 USD
Este proyecto	**Libre**	**$0**

**Precios estimados para licencias industriales/profesionales.*

El riesgo físico y económico

- **Hardware costoso:** Un servomotor industrial promedia entre $200 y $800 USD.
- **El factor error:** Sin un simulador de alta fidelidad, un error de sintaxis o lógica se traduce en una colisión real y pérdida de equipo.
- **Barrera regional:** Estos costos frenan la innovación en la academia y la pequeña industria local.

Nuestra solución

Aporte principal:
Construcción de un ecosistema de **nivel industrial** para control, simulación y telemetría desarrollado íntegramente con **Software Libre**.

Metodología de Validación Técnica

La robustez del sistema se evaluó bajo tres pilares fundamentales en hardware convencional:

- **Estabilidad Visual:** Medición de cuadros por segundo (FPS) para garantizar una experiencia de usuario fluida y sin fatiga.
- **Determinismo Físico:** Análisis de latencia y tiempos de ciclo para asegurar que la simulación responda en tiempo real.
- **Viabilidad Económica:** Comparativa de costos frente a licencias privativas para validar la democratización del acceso.

Evaluación de rendimiento basada en estándares de simulación robótica.

Sistema propuesto: PAROL 6

Configuración de los 6 Ejes

Eje	Función Cinemática
J1	Base (Giro horizontal)
J2 - J3	Posicionamiento (Brazo y codo)
J4 - J5	Orientación (Muñeca esférica)
J6	Efector Final (Herramienta)

¿Por qué el PAROL 6?

Criterio	Justificación
Open Hardware	URDF + STL disponibles públicamente
Configuración	6 GDL estándar industrial (ISO 9283)
Reproducibilidad	Fabricable con impresión 3D
Comunidad	Activa en GitHub, documentación completa

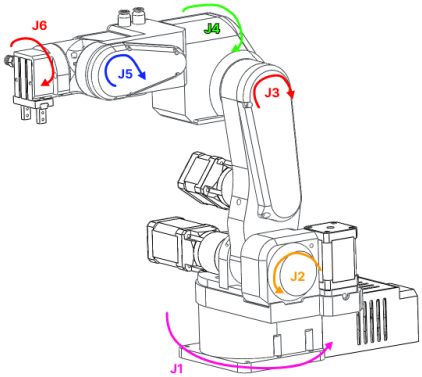

Figura 3: Configuración de los ejes del PAROL 6.

Archivos URDF y STL

Estándar Abierto (XML)

- **Cinemática y Dinámica:** Masas, inercias, centros de gravedad y restricciones de cada articulación.
- **Hard-limits:** Rangos angulares reales de cada motor.
- **Geometría Visual:** Enlaza mallas `.STL` de alta resolución para el renderizado en pantalla.
- **Geometría de Colisión:** Mallas `.STL` simplificadas para el motor de física.

Patrón de Diseño

La descripción del robot (URDF + STL) es la "estrategia" intercambiable, mientras que los módulos de control, física y visualización son la "lógica estable" que la consume.

Al cargar un nuevo URDF el sistema adapta automáticamente:

- Sliders de control y sus límites.
- Motor de colisiones y física.
- Tablas de telemetría (RPY, historial).

Stack Tecnológico: Python · PyBullet · PyQt5

Python

Ecosistema científico (NumPy, Matplotlib). Estándar en IA y robótica. Integración modular sin fricción.

PyBullet

Motor de física computacional. Simula gravedad, inercia y colisiones. Adoptado por DeepMind y OpenAI.

PyQt5

Framework para HMI industrial. Sliders, barras de progreso, tablas dinámicas e indicadores LED profesionales.

Cinemática: estructura matemática del robot

Lenguaje de la Robótica

- **Matrices Homogéneas (4×4):** Combinan rotación y traslación para ubicar cada parte del robot.
- **Encadenamiento:** Cada eslabón se posiciona respecto al anterior.
- **Cálculo del TCP:** La punta del robot se halla multiplicando todas las matrices:

$$T_{base}^{tcp} = \prod_{i=1}^{n} T_i^{i-1}$$

Del URDF a Denavit-Hartenberg

El software extrae el "ADN" del archivo URDF y genera los parámetros D-H automáticamente:

- **Variables (θ_i, d_i):** Los ángulos que mueven los motores.
- **Geometría (a_i, α_i):** El diseño físico y longitudes del brazo.

"Pasamos de una geometría estática al cálculo de fuerzas dinámicas."

Motor físico: Algoritmo de Featherstone $O(n)$

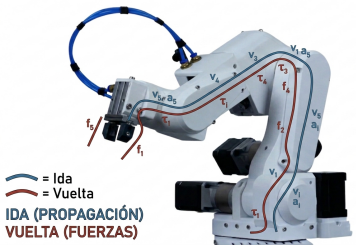

Figura 4: Propagación recursiva de fuerzas y velocidades.

¿Por qué la complejidad es $O(n)$?

- **Método clásico — Euler-Lagrange $O(n^3)$:** Construye la matriz de inercia del sistema ($n \times n$) y la invierte en cada paso. Invertir matrices es $O(n^3)$. Para $n = 6$: **216 operaciones**.
- **Featherstone — RNEA (2 pasadas $O(n)$):**
  - **Ida** (*Propagación*): velocidades y aceleraciones de base a efector — n pasos.
  - **Vuelta** (*Acumulación*): fuerzas y torques de efector a base — n pasos.
  - Total: $2n$ pasos. Para $n = 6$: **12 operaciones**.

Impacto en Rendimiento

La reducción de **216 → 12 operaciones** por paso de tiempo permite ejecutar **4 pasos de física por ciclo** manteniendo los **FPS** Constantes. Con Euler-Lagrange, la misma tasa de refresco sería imposible en hardware convencional.

GUI — Panel izquierdo: control articular y seguridad

Tipo de Control: Control de Posición por PD

El sistema implementa un **controlador PD** (Proporcional-Derivativo) por articulación. Se puede cambiar a control de **velocidad** o **torque** modificando únicamente el parámetro de modo, lo que demuestra la modularidad del sistema.

Control Intuitivo de Movimiento

- **Sliders con Límites Reales:** Cada slider replica los *hard-limits* del URDF: J1 ± 1.70 rad, J2 -0.98/+1.00 rad, J3 -2.00/+1.30 rad, J4–J5 ± 2.00 rad, J6 ± 3.10 rad.

Sistema de Semáforo (Seguridad Pasiva)

- **Verde (15 %–85 %):** Operación normal y segura.
- **Naranja (6 %–14 % y 86 %–94 %):** Proximidad a los límites mecánicos (Precaución).
- **Rojo ($< 5\,\%$ y $> 95\,\%$):** Límite alcanzado. Alerta visual activa para evitar colisiones.

Figura 5: Control directo de ejes.

GUI — Panel central: visualización en tiempo real

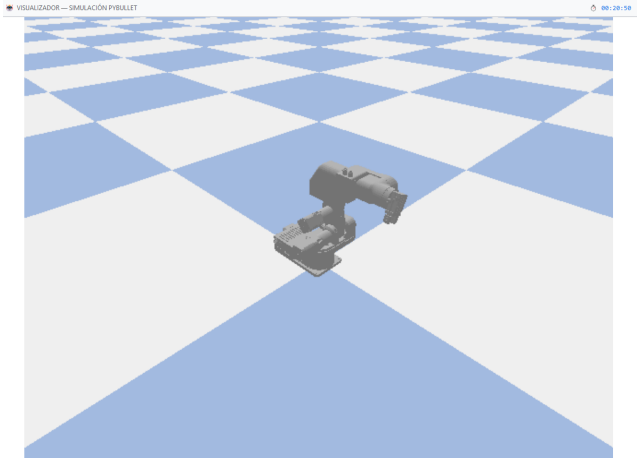

Figura 6: Entorno virtual del PAROL 6.

Simulación Dinámica

- **Cámara Virtual:** Resolución fija de 640×480 px para evitar el crecimiento indefinido de la ventana y garantizar estabilidad gráfica.
- **Estabilidad Gráfica:** El renderizado está optimizado para mantener una fluidez constante de 60 cuadros por segundo.
- **Fidelidad Física:** El modelo 3D responde a gravedad, inercias y colisiones calculadas por PyBullet.

GUI — Panel central: telemetría de precisión

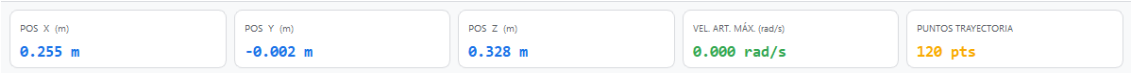

Figura 7: Dashboard de telemetría: Monitoreo en tiempo real del estado del robot.

Localización del Efector (TCP)

- **Monitoreo Espacial:** Posición extraída directamente del motor de simulación con `getLinkState`.
- **Precisión Métrica:** Coordenadas (X, Y, Z) en **metros (m)**, referenciadas al centro de la base del robot.
- **Sincronización:** Datos actualizados a 60 Hz, sincrónicamente con el ciclo de física.

Estado Dinámico del Sistema

El panel despliega 5 tarjetas con información crítica:

- **Ubicación:** Coordenadas cartesianas exactas (m).
- **Seguridad:** Velocidad de los motores en **rad/s**.
- **Memoria de Trazado:** Puntos acumulados para la trayectoria.

GUI — Panel derecho: trayectoria y rastro

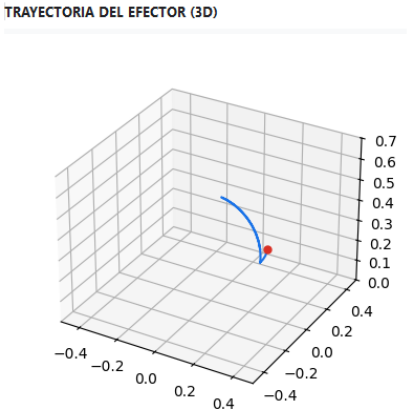

Figura 8: Visualización del rastro espacial.

Trazado del Efector Final

- **Memoria de Movimiento:** El sistema almacena los últimos 120 puntos del recorrido del TCP en el espacio 3D.
- **Análisis Visual:** Permite verificar si el robot está siguiendo la ruta deseada o si existen desviaciones.
- **Indicadores:** Una línea continua muestra el rastro recorrido; un marcador rojo resalta la posición actual en tiempo real.

GUI — Panel derecho: actitud espacial (Orientación)

ACTITUD ESPACIAL (ROLL, PITCH, YAW)

	Roll (rad)	Pitch (rad)	Yaw (rad)
1	-0.000	-0.000	0.490
2	-1.571	0.000	0.490
3	1.571	-1.291	0.490
4	0.000	1.291	-2.652
5	-1.571	1.291	-2.652
6	-0.000	1.291	-2.652

Figura 9: Orientación del cabezal.

Entendiendo la Orientación (Roll, Pitch, Yaw)

- **Interpretación Humana:** Traducimos los cálculos complejos del robot a ángulos fáciles de entender para el operador.

Movimiento	Efecto en el Robot
Roll (Alabeo)	Giro sobre el propio eje de la herramienta.
Pitch (Cabeceo)	Inclinación hacia arriba o hacia abajo.
Yaw (Guiñada)	Giro hacia los lados (izquierda/derecha).

GUI — Panel derecho: historial de telemetría

HISTORIAL ÚLTIMAS POSICIONES EE

	X (m)	Y (m)	Z (m)
1	0.2552	-0.0020	0.3280
2	0.2552	-0.0020	0.3280
3	0.2552	-0.0020	0.3280
4	0.2552	-0.0020	0.3280
5	0.2552	-0.0020	0.3280
6	0.2552	-0.0020	0.3280
7	0.2552	-0.0020	0.3280

Figura 10: Registro de coordenadas.

Análisis de Repetibilidad

La interfaz mantiene un registro de los últimos estados para asegurar que el robot cumple con los puntos de destino.

- **Registro Dinámico:** Tabla con las últimas 8 coordenadas visitadas por el robot.
- **Identificación Visual:** El sistema resalta la posición más reciente para facilitar la lectura de los datos.
- **Utilidad:** Fundamental para procesos de control de calidad o depuración de tareas repetitivas.

Ciclo de ejecución del sistema

1. **Entrada:** Lectura del slider → nuevo ángulo destino enviado al controlador PD de PyBullet.
2. **Física (×4):** Cuatro pasos de simulación con Featherstone RNEA, actualizando fuerzas y torques.
3. **Seguridad:** Validación de límites articulares; activación del semáforo visual si corresponde.
4. **Gestión de Recursos:** La cámara, Matplotlib y telemetría se actualizan a frecuencias reducidas (*throttling*) para no saturar la CPU.

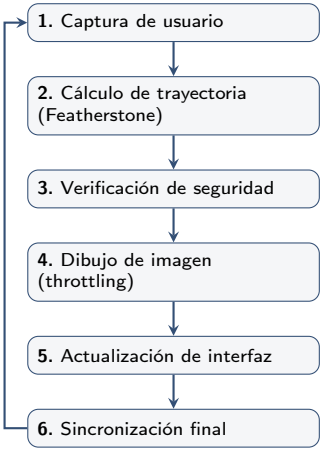

Figura 11: Flujo continuo de respuesta inmediata.

Validación del sistema

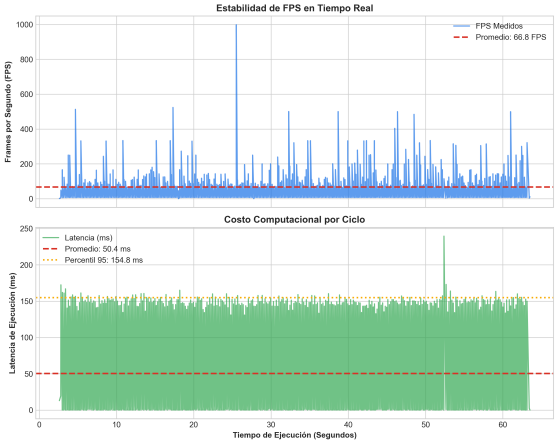

Figura 12: Prueba A: 60s de operación.

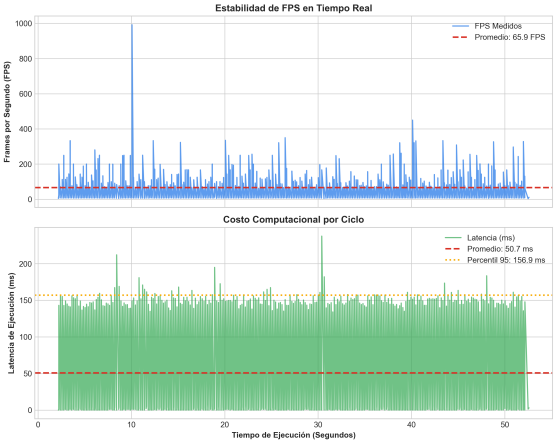

Figura 13: Prueba B: 60s de operación.

Análisis de métricas (Rendimiento)

Métricas Estadísticas — 5 ejecuciones × 60 s

Métrica	Media	Desv. Est.
FPS Promedio	65.80 FPS	±1.2 FPS
Latencia Promedio	50.80 ms	±0.7 ms
Picos latencia (P95)	155.90 ms	±1.1 ms
Estabilidad Temporal	99.2 %	—

Conclusión de Estabilidad

La desviación estándar inferior al **2 %** en todas las pruebas confirma que el sistema es **determinista**. No hay fugas de memoria ni degradación de rendimiento tras uso prolongado.

Impacto del proyecto

- **Impacto Tecnológico:**
  - Integración exitosa de control PD, física de Featherstone y telemetría.
  - Rendimiento de **65.8 FPS**, superando a estándares como Gazebo en hardware convencional.
- **Impacto Educativo:**
  - Creación de un "Laboratorio Virtual" de fallo seguro.
  - Democratización tecnológica: herramientas de nivel industrial sin costo de entrada.
- **Impacto Económico:**
  - Ahorro proyectado de **$5,000 a $20,000 USD** por puesto de trabajo.
  - Mitigación de riesgos mecánicos mediante el monitoreo de *hard-limits*.

Trabajo futuro

Control en espacio de tarea

1. **Cinemática Inversa Adaptativa:** Pasar del movimiento articular manual a la definición de objetivos en coordenadas (X, Y, Z) mediante resolución analítica y numérica.

2. **Aprendizaje por Refuerzo (Reinforcement Learning):** Implementar agentes que aprendan trayectorias óptimas de manera autónoma, reduciendo la necesidad de programación manual para tareas de ensamble.

3. **Gemelo Digital en Tiempo Real:** Sincronización bidireccional completa con el hardware físico para monitoreo predictivo de fallos.

Referencias I

[1] R. Featherstone, *Robot Dynamics Algorithms*. Springer, 1987. doi: 10.1007/978-1-4899-7560-7

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009. doi: 10.1007/978-1-84628-642-1

[3] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005.

[4] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016–2021. [Online]. Available: `http://pybullet.org`

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[6] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, 1992.

[7] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson, 2005.

[8] PCrnjak, "PAROL6 Desktop Robot Arm," GitHub, 2022. [Online]. Available: `https://github.com/PCrnjak/PAROL6-Desktop-robot-arm`

Universidad Tecnológica de Pereira

El conocimiento no debería tener licencia

Ecosistema de simulación y control dinámico

de un manipulador robótico de 6 GDL

Elias Escobar Pereira

Soberanía Tecnológica con Software Libre (Python & PyBullet)

Universidad Tecnológica de Pereira (UTP)
Semillero "Huellas sobre Sistemas Digitales y Mantenibilidad"
GIGSEEA
elias.escobar@utp.edu.co