XVI Encuentro Departamental RREDSI

Semilleros de Investigacion - Risaralda

Proyecto Finalizado - PF-030

Informacion General

Código:
PF-030

Fecha de Registro:
2026-03-19 20:43:26

Semillero de Investigación:
Huellas sobre sistemas digitales y mantenibilidad

Ponente 1

Rol Ponente 1:
Estudiante / Aprendiz

Nombre Ponente 1:
Elias Escobar Pereira

Institucion Ponente 1:
Universidad Tecnológica De Pereira

Ciudad Ponente 1:
Pereira

Lider / Tutor 1

Nombre Lider/Tutor 1:
Ana Julieth Marín Hurtado

Institucion Lider/Tutor 1:
Universidad Tecnológica De Pereira

Ciudad Lider/Tutor 1:
Pereira

Clasificacion

Area del Conocimiento:
Ingeniería y Tecnología

ODS:
4. Educación de Calidad

Contenido del Proyecto

Titulo del Proyecto:
Ecosistema de simulación para el control dinámico y monitoreo de seguridad de un manipulador robótico de 6 GDL

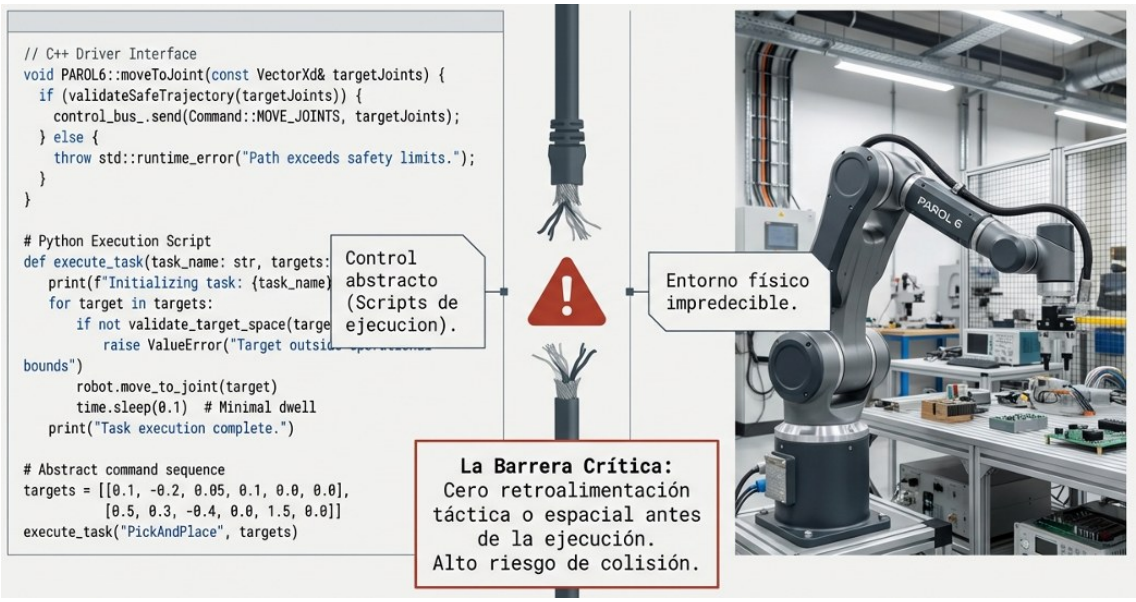
Palabras Clave:
Simulación Dinámica, Manipulador Robótico, Interfaz Humano Máquina, Cinemática, Telemetría

Resumen:
El presente proyecto detalla el diseño y la implementación de una plataforma de software integrada para el control, simulación y monitoreo de un manipulador robótico de seis grados de libertad (6 GDL). El objetivo principal de este trabajo es desarrollar una interfaz gráfica de usuario (GUI), utilizando el lenguaje de programación Python y la librería PyQt, que permita cerrar la brecha entre la programación abstracta y la operación táctica de sistemas robóticos complejos.
La metodología se centra en tres pilares fundamentales: el diseño industrial de una interfaz hombre-máquina (HMI), la integración de un motor de física de alto desempeño y el análisis de datos en tiempo real. En primer lugar, se desarrolla una GUI que facilita el control individual de cada articulación del manipulador a través de comandos manuales intuitivos. En segundo lugar, se integra el entorno de simulación PyBullet, el cual permite representar el comportamiento dinámico del robot, gestionando de forma eficiente la cinemática y las colisiones, visualizando el movimiento de manera sincronizada con los comandos del usuario y por último, se implementan herramientas de telemetría y monitoreo que permiten la representación gráfica del estado del robot. Estas herramientas incluyen la visualización histórica de la trayectoria del efector final y el seguimiento de los ángulos de Euler (Roll, Pitch, Yaw) para la determinación de la orientación y posición.
Un aspecto crítico del desarrollo es la seguridad operacional; para ello, la interfaz integra indicadores visuales dinámicos que alertan sobre la proximidad a los límites físicos de operación, permitiendo un análisis preventivo del comportamiento del robot. Los resultados demuestran una plataforma estable que garantiza una baja latencia en la visualización y una alta precisión en la monitorización de variables articulares. Esta herramienta constituye un entorno de experimentación y validación versátil, sentando las bases para aplicaciones futuras en teleoperación y control autónomo de manipuladores robóticos.

Planteamiento y Justificacion

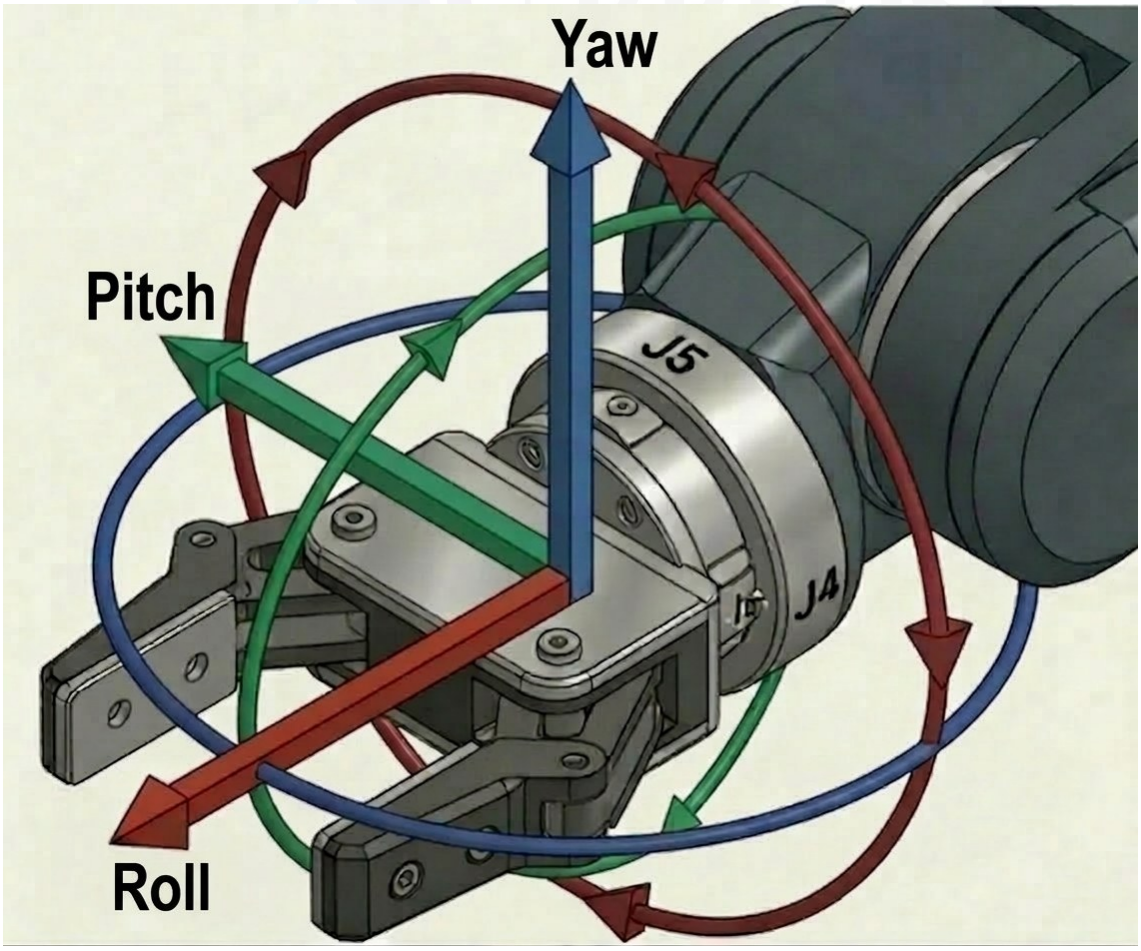
Planteamiento del Problema:
La robótica de manipulación representa uno de los pilares fundamentales de la automatización industrial y la investigación académica. Los manipuladores de seis grados de libertad (6 GDL) son ampliamente utilizados debido a su capacidad para replicar la movilidad del brazo humano, permitiendo alcanzar posiciones y orientaciones complejas en un espacio de trabajo tridimensional (Craig, 2018). Sin embargo, la operación y el estudio de estos sistemas enfrentan una barrera crítica; la desconexión entre el entorno de control algorítmico y la visualización tangible del comportamiento dinámico del robot.

El problema central radica en que muchos de los entornos de desarrollo robótico carecen de interfaces hombre-máquina (HMI) integradas que permitan una transición fluida entre la teoría y la práctica; normalmente, los investigadores y operadores se ven obligados a trabajar con líneas de comandos o scripts aislados que no ofrecen una retroalimentación visual inmediata sobre las variables de estado del sistema. Como se ilustra en la figura



, esta carencia de visualización en tiempo real incrementa el riesgo de colisiones accidentales, daños en los servomotores por exceder límites físicos (hard-limits) y una comprensión deficiente de la cinemática del robot (Siciliano & Khatib, 2016).

Específicamente, en el caso de manipuladores de alta precisión como el PAROL 6, surge el desafío de monitorear simultáneamente múltiples flujos de datos: ángulos articulares individuales, coordenadas cartesianas del efector final (TCP), la posición y orientación espacial. En la figura

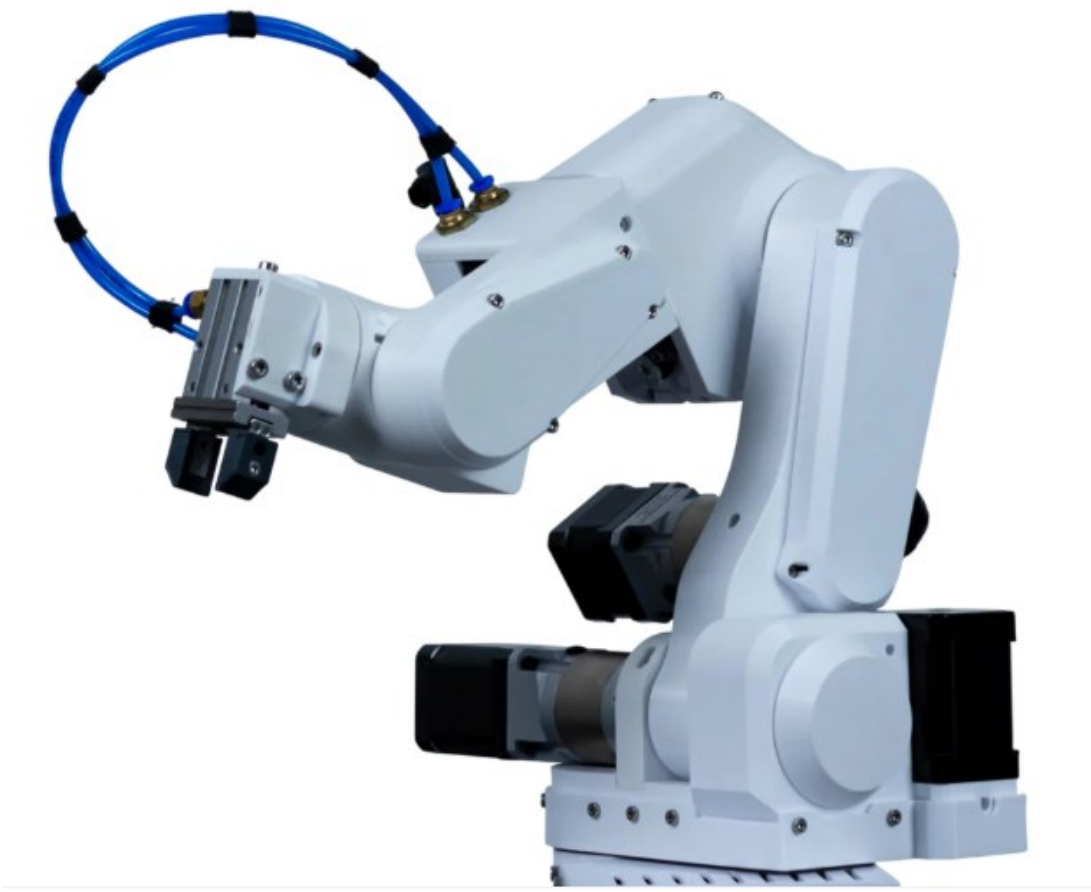


se detalla la complejidad de esta última, definida por los ángulos de Euler (Roll, Pitch, Yaw), sin una herramienta centralizada, el análisis de la trayectoria del efector final se convierte en una tarea compleja de post-procesamiento, lo que impide realizar ajustes correctivos durante la ejecución de una maniobra (Spong et al., 2020).

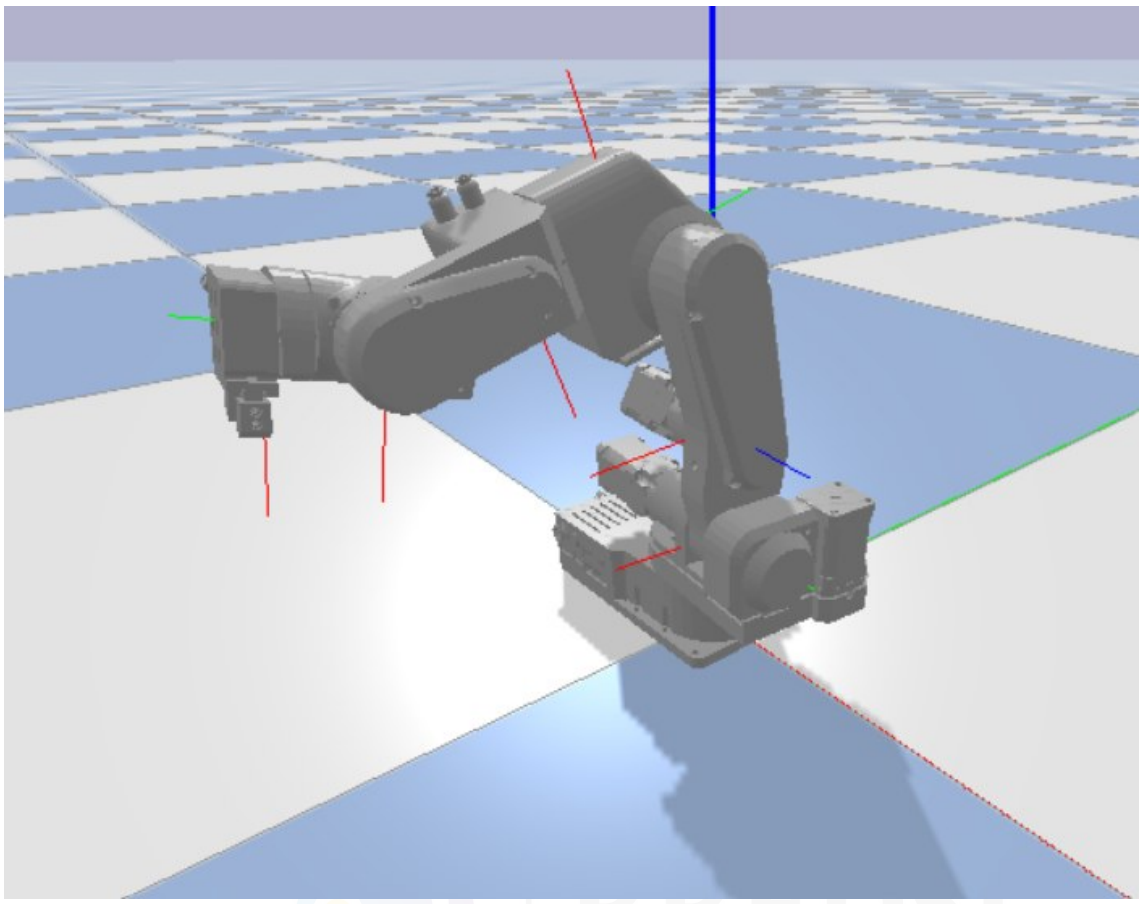
Existen diversas aproximaciones metodológicas para abordar esta validación y control. En el ámbito comercial, se han empleado sofisticados paquetes de software como RoboDK (Garbev & Atanassov, 2020), Siemens Process Simulate o DELMIA (Polden et al., 2011), los cuales ofrecen amplias capacidades de simulación y programación fuera de línea, pero su elevado costo y la dificultad de integrarlos con interfaces de usuario personalizadas representan algunas barreras. Por otro lado, se ha explorado el desarrollo de interfaces basadas en plataformas de código abierto como el Robot Operating System (ROS) con herramientas de visualización rviz (Quigley et al., 2009), que brindan flexibilidad pero conllevan una complejidad de integración considerable. No obstante, la implementación de gemelos digitales o simuladores dinámicos frecuentemente requiere software propietario costoso o plataformas de alta demanda computacional que son difíciles de integrar con interfaces de usuario personalizadas. La ausencia de un entorno de simulación dinámico accesible que pueda representar fielmente el comportamiento físico limita la capacidad de validar trayectorias antes de ser ejecutadas en el hardware real (Coumans & Bai, 2016).

Por lo tanto, se identifica la necesidad de explorar otra metodología de software propia que integre un motor de física con una interfaz de control intuitiva. Esta propuesta busca resolver la falta de una herramienta que consolide, en una sola ventana de operación, el control manual, la simulación dinámica y la telemetría avanzada, garantizando la operatividad del manipulador mediante el monitoreo de sus límites operacionales y facilitando el análisis del comportamiento espacial del robot.

Justificacion:
La implementación de este proyecto nace por la necesidad crítica de garantizar la integridad física de los sistemas robóticos de alta precisión, como el manipulador PAROL 6, durante el desarrollo y prueba de nuevos algoritmos de control. En el ámbito académico y de investigación, la interacción directa con el hardware sin una etapa previa de validación virtual representa un riesgo elevado de daños mecánicos costosos. Bajo este contexto, la implementación de un entorno de simulación dinámica permite la validación de trayectorias antes de su ejecución física, tal como se observa en la figura [

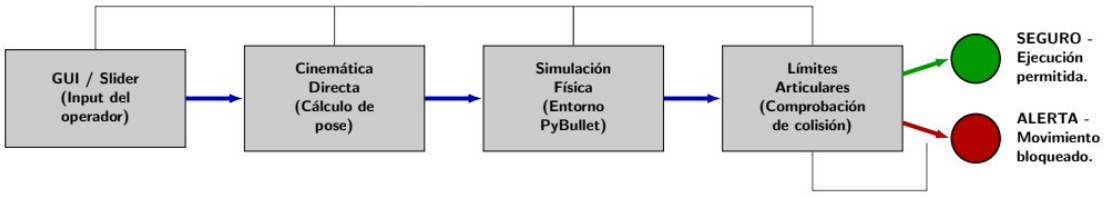


,



], donde se integra el comportamiento del hardware en un entorno virtual controlado. Según (Siciliano y Khatib (2016)), la seguridad operacional en robótica depende directamente de la capacidad de la interfaz para comunicar de forma clara y oportuna los límites de trabajo al operador. Por lo tanto, el desarrollo de un sistema de monitoreo preventivo con retroalimentación visual inmediata es fundamental para reducir la incidencia de fallos durante las fases de experimentación.

Desde una perspectiva investigativa y técnica, la desconexión entre la formulación teórica de la cinemática y su validación física suele ralentizar el ciclo de desarrollo de aplicaciones robóticas. La integración de herramientas de análisis espacial y la visualización de trayectorias en tiempo real conforman una metodología práctica para evaluar lógicas de control de manera segura (Spong et al., 2020). Para asegurar esta protección durante las pruebas, el sistema procesa los comandos a través de la figura



, que detalla la lógica encargada de verificar los límites articulares antes de permitir la ejecución de cualquier movimiento; esta capa de validación facilita la identificación de singularidades y el análisis del espacio de trabajo del robot, aspectos esenciales para la investigación en manipulación avanzada.

Finalmente, el uso de herramientas de código abierto como Python, PyQt y PyBullet se propone como una alternativa viable y complementaria a los entornos de simulación comerciales. Si bien el software propietario de la industria ofrece ecosistemas altamente estandarizados, la adopción de estas tecnologías libres (Coumans & Bai, 2016) proporciona una plataforma modular y de gran flexibilidad. Esto permite la personalización de la interfaz y la integración profunda de algoritmos propios según las necesidades específicas de cada estudio, consolidando una herramienta funcional para la experimentación robótica sin incurrir en elevados costos de licenciamiento.

Objetivos

Objetivo General:

Desarrollar una interfaz de control y visualización para un manipulador robótico de 6 grados de libertad, integrando simulación dinámica, monitoreo de variables articulares y herramientas de análisis del comportamiento del robot.

Objetivos Especificos:

Implementar un ecosistema de simulación dinámica que permita el renderizado del comportamiento físico del manipulador PAROL 6 en un entorno virtual controlado.

Desarrollar un algoritmo de manipulación controlada que gestione las trayectorias articulares y garantice la seguridad operativa.

Diseñar una interfaz gráfica (GUI) para la ejecución de comandos manuales y el monitoreo espacial del comportamiento del robot.

Marco Teorico y Metodologia

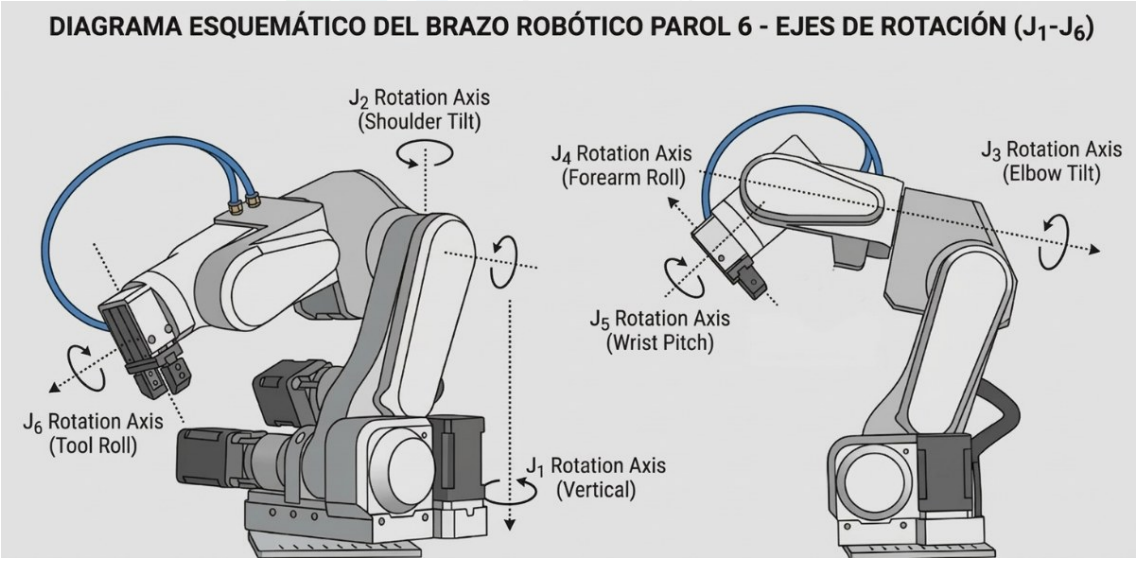
Referente Teorico:

Morfología y Grados de Libertad en Manipuladores

Un manipulador robótico industrial se define mecánicamente como una cadena cinemática de cuerpos rígidos denominados eslabones, conectados por articulaciones que permiten el movimiento relativo. En el caso de un robot de 6 grados de libertad (6 GDL), esta configuración permite que el efector final alcance cualquier posición (x, y, z) y orientación (, ,) dentro de su espacio de trabajo, lo cual es esencial para tareas que requieren destreza similar a la humana. Según Fu et al. (1987), la estructura de 6 GDL es el estándar para la versatilidad operativa, ya que desacopla la posición de la orientación mediante los tres últimos ejes que conforman la muñeca esférica.

Cinemática y Representación Espacial

La cinemática es la piedra angular del control robótico. Se divide en cinemática directa, que determina la pose del efector final a partir de los ángulos articulares, y cinemática inversa, que calcula los ángulos necesarios para alcanzar una pose deseada. Como se observa en la figura



, la morfología del manipulador de 6 GDL permite definir con precisión estos movimientos sobre cada una de sus articulaciones.

Para la descripción de la orientación, este proyecto utiliza los Ángulos de Euler (Roll, Pitch, Yaw). A diferencia de las matrices de rotación que poseen una redundancia de nueve elementos para representar tres grados de libertad, los ángulos RPY ofrecen una representación mínima y altamente intuitiva para el operador humano, facilitando la interpretación de la actitud del robot en entornos de monitoreo (Spong et al., 2020).

Interfaces Hombre-Máquina (HMI) y Usabilidad Cognitiva

El diseño de interfaces para sistemas robóticos ha evolucionado hacia la reducción de la carga cognitiva del usuario. Una

HMI efectiva no solo debe presentar datos, sino transformarlos en información procesable. En este sentido, la utilización de indicadores de estado tipo LED y barras de progreso dinámicas responde a principios de diseño industrial que buscan una respuesta visual inmediata ante situaciones críticas. De acuerdo con Shneiderman et al. (2016), la retroalimentación visual continua y la manipulación directa (sliders) son estrategias fundamentales para que el usuario mantenga un modelo mental preciso del estado actual de la máquina, previniendo errores de operación en zonas de límites físicos o "hard-limits".

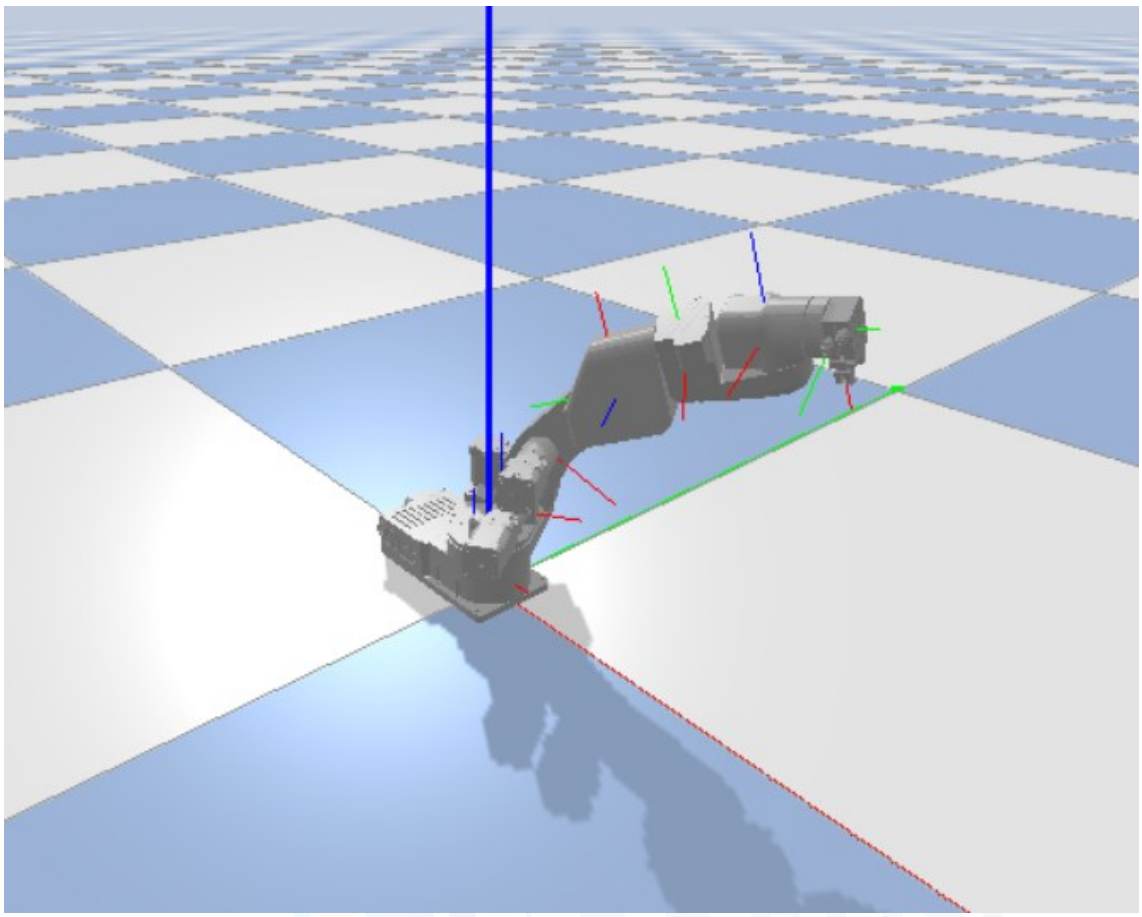
Simulación Dinámica y Motores de Física de Tiempo Real
La simulación de robots ha pasado de ser una herramienta de visualización estética a ser un motor de validación física robusto. Motores de física como PyBullet utilizan algoritmos de dinámica de cuerpos múltiples que consideran la gravedad, la inercia y la fricción. Para resolver estas interacciones, PyBullet implementa el algoritmo de Featherstone, una formulación matemática diseñada para calcular la aceleración de sistemas de cadenas articuladas (como un brazo robótico) de manera recursiva. A diferencia de los métodos tradicionales que invierten matrices de inercia de gran tamaño, este algoritmo se basa en la cinemática y dinámica de cuerpos espaciales, lo que le permite operar con una complejidad computacional de O(n). En este contexto, n representa el número de grados de libertad del robot; una complejidad lineal (O(n)) significa que el tiempo de cálculo aumenta de forma proporcional a los eslabones del robot, y no de forma exponencial.
Esta eficiencia matemática garantiza que, incluso en manipuladores complejos de 6 GDL como el PAROL 6, la sincronización entre el comando del usuario y la respuesta visual sea instantánea, manteniendo una alta tasa de refresco (refresh rate) necesaria para una validación fidedigna del hardware real (Coumans & Bai, 2016).

Desarrollo de Software con Python y el Framework PyQt
La elección de Python como lenguaje de desarrollo se justifica por su ecosistema de librerías científicas y su facilidad para la integración de sistemas. Sin embargo, para interfaces de monitoreo de alta frecuencia, es necesario el uso de programación multihilo (multithreading). El framework PyQt, basado en las librerías Qt en C++, permite que el hilo principal gestione la interfaz gráfica (GUI) mientras que los hilos secundarios procesan los cálculos de telemetría y la comunicación con el motor de física. Según Summerfield (2010), esta arquitectura evita que la interfaz se "congele" durante procesos intensivos de cálculo, manteniendo una tasa de refresco constante que es crítica para la seguridad del monitoreo.

Visualización de Datos y Telemetría
La telemetría en robótica implica la medición y transmisión remota de datos sobre el estado de las articulaciones y el efector final. La integración de herramientas como Matplotlib dentro de entornos PyQt permite la representación gráfica del historial de posiciones. El trazado de trayectorias en 3D permite al analista verificar la suavidad del movimiento y la precisión del seguimiento de rutas, identificando anomalías que no son perceptibles mediante la observación directa del robot (Siciliano & Khatib, 2016).

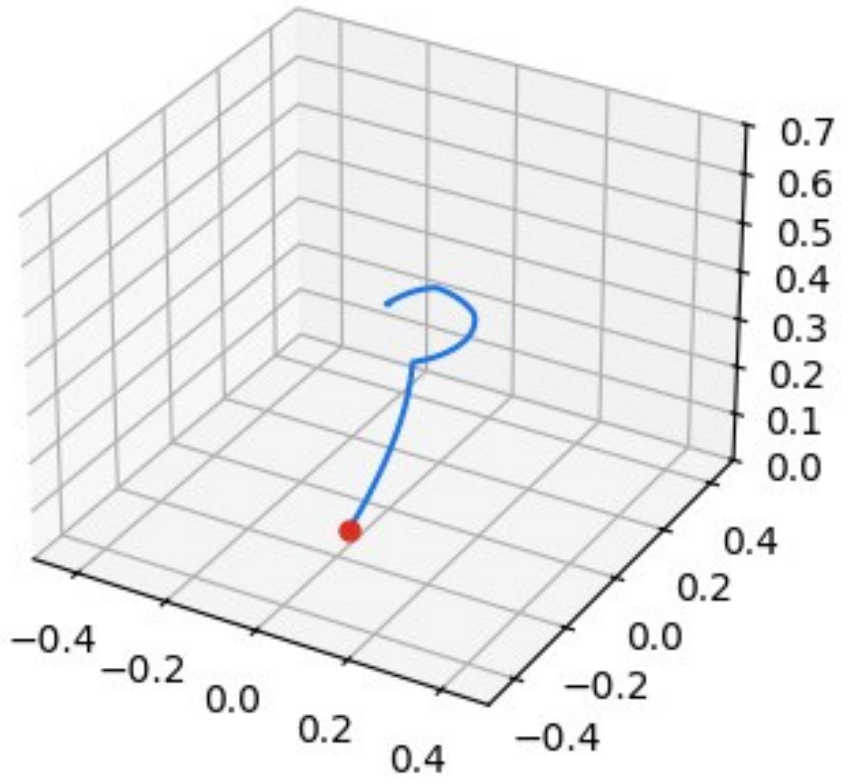
Metodologia:
La metodología del proyecto se estructuró bajo un enfoque de ingeniería de software para sistemas ciberfísicos. El desarrollo se dividió en tres fases secuenciales y modulares, las cuales responden directamente a los objetivos específicos y garantizan una transición fluida desde la física computacional hasta la interacción humana.

Fase 1: Implementación del Ecosistema de Simulación Dinámica Esta primera etapa construyó el núcleo físico en PyBullet para representar fielmente el comportamiento mecánico del manipulador. El entorno virtual configurado se ilustra en la figura



.
Actividad 1.1: Configuración del Motor de Física: Se inicializó el entorno de simulación procesando el archivo URDF del PAROL 6. Se definieron las masas de los eslabones, los ejes articulares, las restricciones de movimiento, la gravedad y las mallas de colisión para el motor de dinámica.
Actividad 1.2: Sincronización y Temporización: Para asegurar la fluidez temporal, se implementó un temporizador de alta frecuencia (QTimer). Este componente sincroniza las instrucciones de la interfaz con los avances algorítmicos del simulador (stepSimulation), garantizando una ejecución a tasa constante.
Actividad 1.3: Renderizado de Cámara Sintética: Se configuró un sensor de visión virtual en el espacio de trabajo. La captura se procesa dinámicamente mediante arreglos de NumPy, permitiendo la observación visual en tiempo real de las reacciones mecánicas del robot.

Fase 2: Desarrollo del Algoritmo de Control y Seguridad Operativa Una vez validado el entorno, se programó la lógica matemática responsable de gestionar las trayectorias articulares y salvaguardar la integridad del sistema. La extracción de estos datos espaciales se observa en la figura



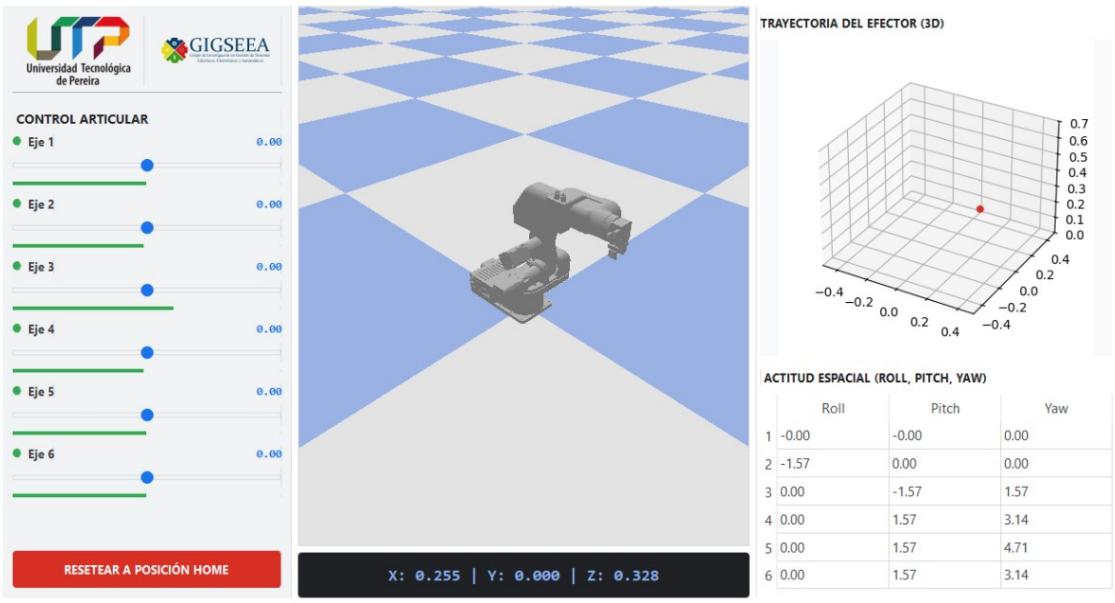
.

Actividad 2.1: Lógica de Manipulación: Se desarrolló una arquitectura basada en señales y ranuras (signals and slots) para traducir las entradas numéricas del usuario en comandos de control de posición. Esto permite actualizar el estado del robot con una resolución de centésimas de radián.

Actividad 2.2: Cinemática Directa (TCP y RPY): Se integraron algoritmos analíticos para calcular la posición cartesiana del efector final (Tool Center Point) y su orientación en el espacio mediante ángulos de Euler (Roll, Pitch, Yaw), lo cual es fundamental para la validación teórica del movimiento.

Actividad 2.3: Sistema de Seguridad y Límites: Se programó una función de monitoreo continuo que compara la posición articular instantánea contra los límites mecánicos (hard-limits) del URDF. Esta restricción algorítmica condiciona el movimiento para prevenir autocolisiones.

Fase 3: Diseño e Integración de la Interfaz Gráfica (GUI) La fase final centralizó las herramientas de control y telemetría en una Interfaz Humano-Máquina (HMI), diseñada bajo principios de ergonomía cognitiva, como se presenta en la figura



.

Actividad 3.1: Diseño del Layout Visual: Utilizando PyQt5, se estructuró una rejilla (QGridLayout) que divide la pantalla en tres módulos funcionales: control articular, cámara central y telemetría. Esta distribución estratégica minimiza la carga visual y facilita la operación intuitiva.

Actividad 3.2: Representación de Telemetría Espacial: Se acopló un lienzo interactivo de Matplotlib para graficar en 3D la trayectoria del efector final. Paralelamente, se dispuso una tabla de datos dinámicos para la lectura ininterrumpida de los valores cinemáticos.

Actividad 3.3: Retroalimentación Preventiva: Se aplicaron hojas de estilo (QSS) para un acabado industrial. Adicionalmente, se programaron indicadores LED virtuales que, enlazados a la lógica de la Fase 2, alertan mediante un código de colores (verde, ámbar, rojo) sobre la proximidad a los límites de riesgo.

Resultados y Conclusiones

Resultados y Analisis:

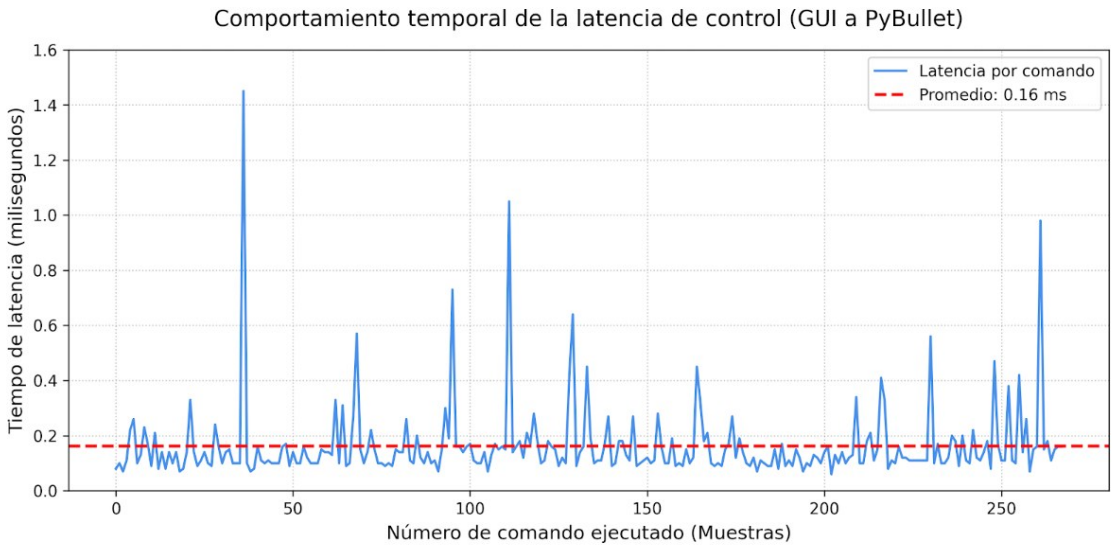
El desarrollo de la plataforma permitió consolidar un entorno funcional donde la sincronización entre la interfaz de usuario y el motor de física fue evaluada mediante el registro de tiempos de ejecución. Los resultados revelaron un contraste significativo entre la eficiencia del control algorítmico y la carga computacional del renderizado gráfico.

En cuanto al control articular, se logró una respuesta casi instantánea. Como se detalla en la tabla

Tabla 1: Evaluación de rendimiento del entorno de simulación.

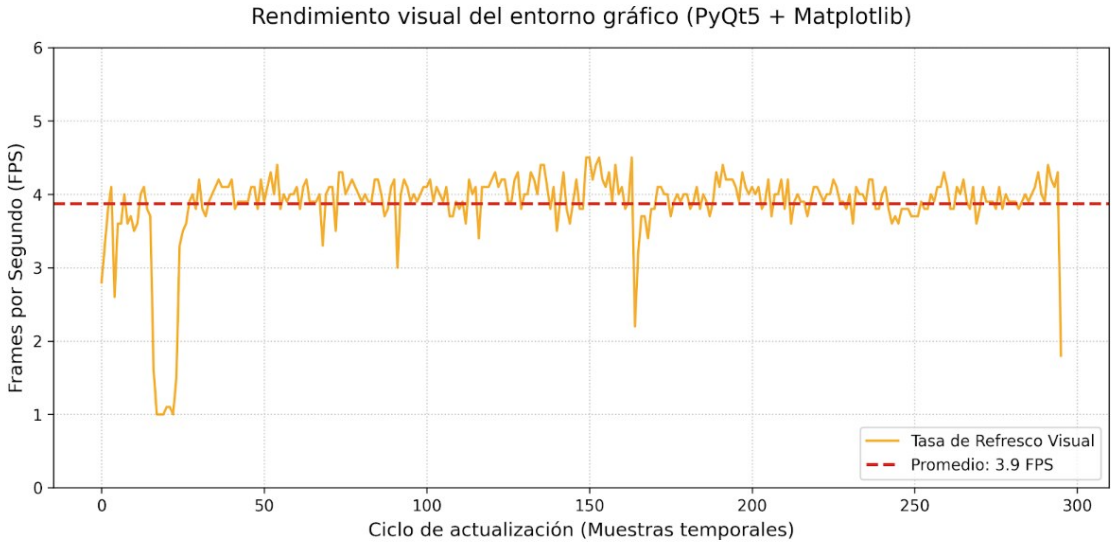
Métrica de Desempeño	Valor Medido (Promedio)	Desviación Estándar (σ)
Latencia de comando (GUI a Motor)	0.16 ms	$\pm$ 0.08 ms
Picos máximos de latencia (Outliers)	1.45 ms	-
Tasa de refresco visual GUI (Renderizado)	3.9 FPS	$\pm$ 0.3 FPS

, el tiempo transcurrido desde el accionamiento de los deslizadores en la HMI hasta la actualización del comando de posición en el motor físico promedió los 0.16 milisegundos, con picos máximos que no superaron los 1.5 ms, comportamiento que se puede observar de manera detallada en la figura



. Al contrastar este desempeño con la literatura, estudios como el de Maruyama et al. (2016) indican que entornos tradicionales basados en ROS 1 integrados con simuladores como Gazebo pueden presentar latencias inter-procesos de entre 30 y 50 ms. Este hallazgo valida que la arquitectura directa propuesta elimina el sobrecosto computacional de los middlewares de red, logrando una sincronización en tiempo real superior para operaciones de teleoperación.

Por otro lado, el análisis del renderizado visual (simulación dinámica y telemetría 3D) reflejó una tasa de refresco promedio de 3.9 FPS. Como se evidencia en el registro temporal de la figura



, el rendimiento se mantiene estable a pesar de ser inherente a la arquitectura síncrona de las librerías gráficas utilizadas (PyQt5 y Matplotlib). La extracción continua de la matriz de píxeles mediante la función de cámara virtual de PyBullet, sumada al redibujado de la trayectoria espacial tridimensional en el mismo hilo de ejecución principal, genera un cuello de botella en el procesamiento visual. No obstante, es fundamental destacar que esta limitación visual no afecta la frecuencia de actualización de la física subyacente, la cual sigue respondiendo de gran manera y fiel a las leyes de inercia y colisión (Coumans & Bai, 2016).

Por último, las herramientas de telemetría arrojaron datos cruciales para la seguridad. El sistema de alertas visuales (LEDs) respondió de manera determinista ante los hard-limits definidos en el archivo URDF, bloqueando movimientos prohibidos sin verse afectado por la tasa de refresco visual. Además, la tabla de orientación RPY demostró una buena precisión consistente con la cinemática directa. Estos resultados demuestran que, si bien existen oportunidades de mejora en el uso de subprocesos para optimizar los FPS de la interfaz, el sistema cumple íntegramente su propósito investigativo: garantizar un control rápido y seguro del manipulador PAROL 6.

Conclusiones:

El desarrollo del presente proyecto permitió la consolidación de un ecosistema de control y simulación para el manipulador PAROL 6. A partir del análisis de los resultados obtenidos y contrastados con los objetivos iniciales, se extraen las siguientes conclusiones:

En primer lugar, la integración del motor físico PyBullet demostró ser una plataforma altamente fidedigna para la validación de cinemática y dinámica. Contar con un entorno de simulación previo a la operación real actúa como un filtro de seguridad crítico, reduciendo significativamente la probabilidad de colisiones articulares y optimizando los tiempos de experimentación al permitir el ensayo de trayectorias en un espacio digital controlado.

En segundo lugar, la instrumentación del algoritmo de control reveló un desempeño sobresaliente en la gestión de comandos. La arquitectura directa implementada en Python logró una latencia de respuesta promedio de 0.16 ms entre la interfaz y el motor físico. Esta eficiencia demuestra que es posible prescindir de middlewares de red complejos para tareas de teleoperación local, garantizando una reacción casi instantánea de la lógica de seguridad ante la proximidad de los límites articulares mecánicos.

En cuanto a la interfaz gráfica (GUI) y el análisis espacial, el sistema logró traducir exitosamente datos numéricos abstractos (como matrices de transformación y cuaterniones) en representaciones visuales intuitivas (telemetría RPY y estelas 3D). Si bien se identificó que la actualización síncrona de gráficos tridimensionales genera un cuello de botella en la tasa de refresco visual de la interfaz (estabilizada en 3.9 FPS), se concluye que esta limitación gráfica no degrada ni interfiere con la alta frecuencia de cálculo de la física subyacente, cumpliendo íntegramente con el propósito de monitoreo seguro.

En definitiva, la arquitectura de software desarrollada ratifica que es posible construir herramientas de investigación de alto nivel utilizando exclusivamente recursos de código abierto (PyQt5, Matplotlib, PyBullet). Frente a las soluciones cerradas de software propietario, este ecosistema ofrece una alternativa técnica y económicamente viable, brindando flexibilidad modular para el estudio de sistemas ciberfísicos complejos sin incurrir en costos de licenciamiento.

Impactos Alcanzados:

Impacto Tecnológico: El proyecto aporta una herramienta de software unificada que centraliza el control manual, la simulación física y la telemetría, mitigando la dependencia de múltiples programas aislados para la operación del manipulador. La extracción de datos cinemáticos en tiempo real y su visualización tridimensional dotan a la plataforma de características propias del estado del arte en programación fuera de línea (Offline Programming), facilitando una transición fluida entre el diseño algorítmico y la ejecución en hardware.

Impacto Educativo: El sistema fomenta un aprendizaje significativo al consolidarse como un laboratorio virtual permanente. Permite a los estudiantes e investigadores experimentar con la cinemática directa e inversa de un robot de 6 GDL en un entorno libre de riesgos mecánicos. Esta herramienta democratiza el acceso a prácticas de alta fidelidad, permitiendo validar lógicas de control y planificación de trayectorias sin el temor de comprometer equipos físicos de alto costo.

Impacto Económico: La adopción exclusiva de librerías de código abierto genera un ahorro sustancial para las instituciones académicas al eliminar las barreras económicas de las licencias de simulación industrial. Asimismo, el sistema de alertas tempranas y bloqueo por límites articulares ejerce una función de mantenimiento preventivo indirecto; al evitar colisiones y esfuerzos fuera de rango, se extiende significativamente la vida útil de los servomotores y eslabones impresos en 3D del PAROL 6, reduciendo los costos de reparación a largo plazo.

Impacto Investigativo: Al entregar una arquitectura de software modular y bien documentada, esta plataforma sienta una base técnica sólida para el desarrollo de futuras tesis y proyectos. Su código escalable permite que el ecosistema sea utilizado como punto de partida para integrar nuevas líneas de investigación avanzada, tales como algoritmos de control autónomo, visión artificial, aprendizaje por refuerzo (Machine Learning) o su futura integración con Robot Operating System (ROS).

Bibliografia

Bibliografia:

Craig, J. J. (2018). Introduction to robotics: Mechanics and control (4.ª ed.). Pearson.

Coumans, E., & Bai, Y. (2016). PyBullet, a Python module for physics simulation for games, robotics and machine

learning. http://pybullet.org

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). Robot modeling and control (2.ª ed.). Wiley.

Siciliano, B., & Khatib, O. (Eds.). (2016). Springer handbook of robotics (2.ª ed.). Springer.

Fu, K. S., Gonzalez, R. C., & Lee, C. S. G. (1987). Robotics: Control, sensing, vision, and intelligence. McGraw-Hill.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the user interface: Strategies for effective human-computer interaction (6.ª ed.). Pearson.

Summerfield, M. (2010). Rapid GUI programming with Python and Qt: The definitive guide to PyQt programming. Prentice Hall.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. ICRA workshop on open source software, 3(3.2), 5.

Garbev, A., & Atanassov, A. (2020). Comparative analysis of RoboDK and Robot Operating System for solving diagnostics tasks in off-line programming. 2020 International Conference Automatics and Informatics (ICAI), 1-5. IEEE.

Polden, J., Pan, Z., Larkin, N., & van Duin, S. (2011). Offline programming for a complex welding system using DELMIA automation. 2011 Eighth International Conference on Computer Graphics, Imaging and Visualization, 58-63. IEEE

Maruyama, Y., Kato, S., & Azumi, T. (2016). Exploring the performance of ROS2. Proceedings of the 13th International Conference on Embedded Software (EMSOFT), 1-10.

Autores

Autor 1 - Nombre:
Escobar Pereira, Elias

Autor 1 - Institucion:
Universidad Tecnológica De Pereira

Autor 2 - Nombre:
Ortega Quiñones, Kevin David

Autor 2 - Institucion:
Universidad Tecnológica De Pereira

Autor 3 - Nombre:
Holguín Londoño, Mauricio

Autor 3 - Institucion:
Universidad Tecnológica De Pereira

Autor 4 - Nombre:
Holguín Londoño, German Andres

Autor 4 - Institucion:
Universidad Tecnológica De Pereira

Documentos de Soporte

URL Carta de Aval:
https://drive.google.com/file/d/1bbYYpzdHkfaaeLhn-ozSLQbZs5MVRazn/view?usp=sharing