UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE SISTEMA PARA EL CONTROL DE UN BRAZO ROBÓTICO IMPRESO EN 3D MEDIANTE API DE PROGRAMACIÓN E INTERFAZ GRÁFICA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ALBERTO ABARZÚA POBLETE

PROFESOR GUÍA:
LUCIANO RADRIGAN FIGUEROA

MIEMBROS DE LA COMISIÓN:

NANCY HITSCHFELD KAHLER

PATRICIO INOSTROZA FAJARDIN

SANTIAGO DE CHILE
2023

Resumen

El presente informe describe el desarrollo de un sistema de control para un brazo robótico impreso en 3D con seis grados de libertad. El objetivo del proyecto fue diseñar una solución de control accesible y fácil de implementar, adecuada para usuarios con diferentes niveles de conocimientos técnicos.

El proyecto introduce una interfaz web intuitiva, pensada para usuarios no técnicos, que facilita el movimiento libre del brazo robótico y la creación de rutinas de movimiento. Simultáneamente, se desarrolló una biblioteca de control en Python, brindando a los usuarios con habilidades de programación la posibilidad de interactuar de manera más detallada y flexible con el brazo.

Un componente crucial es el firmware para el microcontrolador ESP32, que garantiza una comunicación efectiva entre el hardware del brazo robótico y la plataforma de control. Este firmware es fundamental para la interoperabilidad y funcionalidad total del sistema.

Además, se destacan los tipos de movimientos que el brazo robótico puede realizar, que incluyen la manipulación precisa de objetos, movimientos de rotación y alcance, y la capacidad de simular gestos humanos. Estas funciones permiten aplicaciones en tareas variadas, desde ensamblaje industrial hasta educación y investigación.

El enfoque del proyecto se centró en la facilidad de uso y adaptabilidad. Tras una etapa de validación que incluyó pruebas de usabilidad con usuarios de distintos ámbitos y conocimientos en robótica, se confirmó la eficacia y accesibilidad de la plataforma.

En conclusión, este proyecto logra su objetivo de proporcionar una plataforma de control versátil y accesible para el brazo robótico, demostrando su aplicabilidad y potencial para promover la inclusión y el aprendizaje en el ámbito de la robótica.

Tabla de Contenido

1. Introducción

Adentrarse en el mundo de la electrónica y la programación de microcontroladores puede resultar intimidante para aquellos que son nuevos en el área de la computación. Uno de los elementos que puede incentivar a estos principiantes es la posibilidad de ver los resultados físicos de su trabajo programático, lo cual puede generar un sentido de logro tangible y de progresión. Sin embargo, para suscitar interés en este universo tan complejo, es crucial crear ambientes de aprendizaje amigables y de fácil uso para estos. Idealmente, estos entornos deberían permitirles aumentar la dificultad de manera gradual, manteniéndose dentro de un mismo ambiente, a medida que van adquiriendo confianza y habilidades.

Un área particularmente interesante y visualmente atractiva en electrónica y programación es la de los brazos robóticos. Estos ofrecen un atractivo visual y presentan muchas aristas en cuanto a su uso, diseño y control. Los brazos robóticos son ampliamente utilizados tanto en diversas empresas de manufactura a gran escala como en contextos educativos y pueden ser precisos, rápidos e interactivos, facilitando tanto tareas completamente automatizadas como las que requieren interacción humana.

Un brazo robótico es un dispositivo mecánico diseñado para emular la funcionalidad y movilidad del brazo humano. Estos brazos, compuestos por segmentos interconectados, generalmente por articulaciones, pueden realizar movimientos y tareas con precisión y eficiencia. Cada articulación del brazo robótico se asemeja a las articulaciones humanas, como el codo o la muñeca, permitiendo así movimientos que imitan de cerca las capacidades del brazo humano.

Los brazos robóticos se usan en varias aplicaciones, desde la automatización industrial hasta la investigación científica, e incluso en tareas cotidianas que requieren precisión y repetibilidad. Por ejemplo, en el ámbito industrial, los brazos robóticos realizan tareas como ensamblaje, pintura y soldadura, mientras que en el campo médico pueden asistir en procedimientos quirúrgicos delicados.

La eficacia de un brazo robótico radica en su capacidad para realizar movimientos precisos y controlados, lo que se logra mediante un complejo sistema de control. Este sistema integra software y hardware, incluyendo motores y sensores, que trabajan en conjunto para interpretar las instrucciones de movimiento y ejecutarlas con exactitud. La programación de estos brazos robóticos es una tarea compleja que involucra el entendimiento profundo de la cinemática, permitiendo así que el brazo realice acciones que van desde las más simples hasta las extremadamente complejas y delicadas.

En lo que respecta a su control, los brazos robóticos normalmente requieren la interacción con motores para mover sus articulaciones. Un ejemplo común es el brazo robótico de 6 grados de libertad, muy popular tanto en la industria como en las opciones educati-

vas y los kits de construcción. Este atractivo se debe a su versatilidad de movimiento y a su modelo matemático de movimiento relativamente simple. En general, para quienes desean adentrarse en el mundo de la robótica, específicamente en los brazos robóticos, hay un conjunto claro de opciones disponibles. Sin embargo, estas opciones a menudo se ven limitadas por factores como el costo y los conocimientos técnicos requeridos. En general, se pueden identificar dos tipos de soluciones.

Por un lado, están los kits que permiten ensamblar un brazo robótico sencillo. Aunque estos kits son accesibles en términos de precio, por lo general, emplean un microcontrolador con firmware propietario y proporcionan un software de control muy básico y poco extensible. Esto limita la solución a ser poco más que un dispositivo aislado, solo capaz de realizar movimientos simples. Por otro lado, existen las instrucciones y archivos 3D para construir e imprimir uno mismo un brazo robótico preexistente. Con esta opción, la parte de control y software queda en manos del usuario. Aunque existen muchas soluciones disponibles, estas requieren un gran conocimiento técnico para su implementación. Además, estas soluciones suelen ser muy amplias, ya que están diseñadas para usarse en varios proyectos de robótica, no solo en brazos robóticos.

Por otro lado, si una institución está interesada en organizar talleres para atraer a estudiantes o a personas potencialmente interesadas en el campo de la robótica, existen opciones de un nivel más elevado, dada la capacidad institucional para invertir en equipamiento. En este escenario, se encuentran disponibles soluciones de brazos robóticos profesionales de nivel educativo que incluyen software de control avanzado con muchas posibilidades. Sin embargo, estas soluciones pueden ser costosas, difíciles de mantener (en términos de mantenimiento físico del robot en caso de fallas) y limitan el entorno de control a lo definido por el fabricante.

Alternativamente, las instituciones pueden optar por la ruta de la creación propia, aprovechando proyectos de código abierto que proporcionan instrucciones para construir brazos robóticos impresos en 3D. Nuevamente, la elección del control y el software recae en la institución. Las opciones más comunes suelen ser ROS y GRBL, aunque la implementación de estas puede complicar el uso de la herramienta y requerir una considerable cantidad de conocimiento técnico.

1.1.1. Estado del Arte

Para la definición de este proyecto, es crucial investigar las soluciones actuales, tanto electrónicas como de software, para un brazo robótico y su plataforma de control. Al examinar las soluciones existentes, se considera su funcionalidad, eficacia y el potencial de mejora a través de un enfoque de código abierto y bajo costo.

En términos de accesibilidad, la nueva plataforma busca ser inclusiva, con una interfaz intuitiva que sea fácilmente comprensible para usuarios no técnicos. Esto implica simplificación en la configuración y operación, así como una documentación clara y didáctica.

En cuanto a la facilidad de uso, se enfatiza en la reducción de complejidad para tareas básicas, permitiendo a usuarios de cualquier nivel interactuar con el brazo robótico sin necesidad de conocimientos profundos en programación o electrónica.

Esta investigación permitirá determinar las necesidades y brechas existentes, estableciendo una base para el desarrollo de una plataforma de control accesible, fácil de usar y adaptable a l

1.1.2. Solución Propuesta y Metodología

Tras analizar las opciones actuales, se evidencia un vacío en lo que respecta a una solución de control que sea fácil de usar, que ofrezca posibilidades tanto para el control básico como para aspectos más avanzados y que proporcione un entorno para el crecimiento y la complejidad en un marco extensible y de código abierto. Este crecimiento y complejidad se refieren al desarrollo progresivo de habilidades en programación y robótica, así como a la capacidad de la plataforma para adaptarse a proyectos más sofisticados y técnicamente demandantes. A partir de esta observación surge el planteamiento de esta memoria:

> « La creación de una plataforma de control completa para un brazo robótico, incluyendo una interfaz gráfica orientada a usuarios con poco o ningún conocimiento de programación o electrónica, y una API programática para usuarios con conocimientos de programación, pero no de electrónica, con el objetivo de fomentar el crecimiento en un entorno de código abierto y extensible.»

Por ejemplo, un usuario con poca experiencia podría empezar utilizando la interfaz gráfica para realizar movimientos básicos del brazo robótico, como la manipulación de objetos simples o la ejecución de rutinas predefinidas. A medida que adquiera más experiencia, podría comenzar a experimentar con la API programática, desarrollando rutinas más complejas o integrando el brazo con otros sistemas y sensores. El crecimiento en este contexto implica pasar de un control elemental a la implementación de soluciones más sofisticadas, como la programación de tareas automatizadas o el uso del brazo en aplicaciones de aprendizaje de máquinas.

Este planteamiento propone llenar la brecha entre las soluciones avanzadas y técnicamente exigentes. Al ser de código abierto, permite su uso y expansión por parte de una amplia gama de usuarios. Además, este enfoque promueve la inclusión, al permitir que personas con varios niveles de experiencia y conocimientos puedan aprender y contribuir al

mundo de la robótica. Asimismo, promueve la innovación al ser extensible, lo que significa que la plataforma puede evolucionar y adaptarse a nuevas necesidades y avances tecnológicos.

Metodología

Para alcanzar este planteamiento, se llevó a cabo un desarrollo estructurado en tres iteraciones. La primera iteración consistió en la creación de la biblioteca y el firmware. La segunda iteración se centró en el desarrollo de la interfaz, y finalmente, la tercera iteración se dedicó a la integración y el perfeccionamiento del sistema. Entre la segunda y la tercera iteración, se realizó una etapa de validación crucial. Esta fase incluyó una actividad específica y la aplicación de una encuesta para recoger retroalimentación esencial.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo principal de este proyecto es desarrollar una plataforma de software de control que sea accesible para un amplio rango de usuarios, desde aquellos con experiencia en programación hasta los que no la tienen. Esta plataforma, deberá ser fácil de implementar en hardware existente, constará de una interfaz web visual para usuarios no técnicos y una API de programación o SDK para los usuarios técnicos. De esta manera, se busca que el control de un brazo robótico sea una tarea intuitiva y accesible para todos, independientemente de su nivel de habilidades técnicas.

1.2.2. Objetivos Específicos

I. **Interfaz visual de fácil uso**

El primer objetivo específico es el desarrollo de una plataforma de control visual basada en la web. Esta permitirá a los usuarios sin conocimientos de programación controlar el brazo robótico de manera intuitiva y eficiente. La interfaz se diseñará para ser amigable y fácil de usar, proporcionando un acceso sencillo a las funciones básicas y avanzadas del brazo robótico.

II. **Librería de Control**

El segundo objetivo es desarrollar una librería de control programática para los usuarios con conocimientos de programación. Esta permitirá a los usuarios experimentados interactuar con el brazo robótico de manera más directa y flexible, proporcionando las herramientas necesarias para crear, modificar y optimizar las rutinas de movimiento del brazo robótico.

III. **Firmware extensible**

El tercer objetivo es desarrollar un firmware base para el ESP-32, enfocado en minimizar el código dependiente de hardware. Esto permitirá adaptarlo fácilmente a otros microcontroladores. El firmware proporcionará funciones esenciales para controlar el brazo robótico, asegurando una interacción fluida entre el hardware y la plataforma de control.

IV. **Orientado a la fácil implementación**

El último objetivo específico es la implementación de la plataforma de control en hardware existente. El estudiante ya cuenta con una base y un prototipo de brazo robótico similar a los presentados en el estado del arte de brazos robóticos de código abierto. Esto permitirá

probar y validar la plataforma en un entorno real y tangible, garantizando su funcionalidad y eficacia en escenarios prácticos.

La solución propuesta para el control de un brazo robótico se estructura en torno a tres servicios principales, diseñados para optimizar tanto la accesibilidad como la eficiencia técnica. Primero, la interfaz, que será una aplicación web, favoreciendo su accesibilidad y facilidad de uso. Esta interfaz utilizará React, un marco de trabajo moderno y ampliamente reconocido, para asegurar una experiencia de usuario fluida y una integración eficaz con diversas funcionalidades.

En segundo lugar, se desarrollará una librería en Python, un lenguaje de programación conocido por su simplicidad y popularidad. Esta librería permitirá el control programático del brazo robótico, facilitando a los usuarios con conocimientos técnicos manipular y programar el brazo de manera más detallada y específica, adaptándose a sus necesidades y proyectos particulares.

Por último, la implementación base del sistema se realizará utilizando un microcontrolador ESP32. Se ha elegido este hardware específicamente por su gran popularidad en el ámbito de la robótica DIY, su facilidad de uso, su bajo costo y, especialmente, por incluir conectividad Wi-Fi. Esto permitirá una mayor flexibilidad en la comunicación y control del brazo robótico, además de facilitar la integración con otros componentes y sistemas en un entorno de desarrollo abierto y extensible.

En conjunto, estos tres elementos conforman una solución versátil, que busca cerrar la brecha entre usuarios con distintos niveles de habilidades técnicas y conocimientos en programación y robótica, democratizando el acceso a la tecnología de brazos robóticos y fomentando la innovación y el aprendizaje en este fascinante campo.

1.3. Alcance y Limitaciones

En el desarrollo de la plataforma de control para el brazo robótico, es crucial considerar tanto los alcances como las limitaciones inherentes al proyecto. Un aspecto primordial es la simplicidad de la interfaz, que, pese a ser intuitiva y fácil de usar, encuentra su límite en la complejidad intrínseca del brazo robótico. Esta complejidad se refleja también en la librería de control, cuya efectividad y versatilidad están condicionadas por las características y capacidades técnicas del brazo.

Además, la adaptabilidad de la plataforma a diferentes modelos de brazos robóticos se ve limitada por las especificaciones del hardware utilizado, incluyendo el tipo de microcontrolador, sensores, compatibilidad con el modelo matemático y motores. Esta variabilidad en el hardware puede restringir la compatibilidad universal de la plataforma, exigiendo ajustes o modificaciones para adaptarse a cada modelo específico de brazo robótico.

Otra limitación significativa radica en la comprensión de la complejidad operativa del brazo. Conceptos como la cinemática inversa, que es crucial para el movimiento preciso del brazo, y la localización de la herramienta en el espacio, utilizando coordenadas cartesianas y ángulos de acercamiento o Euler, pueden resultar desafiantes para usuarios sin experiencia previa en robótica. Estos conceptos, aunque se abordarán y simplificarán lo posible con la interfaz de usuario y la documentación, requerirán comprensión y familiarización del usuario.

Por lo tanto, si bien el objetivo es hacer la plataforma lo más accesible y amigable posible, es importante reconocer que ciertos aspectos técnicos inherentes al control de un brazo robótico de esta naturaleza pueden presentar una curva de aprendizaje para los usuarios finales. La documentación y el diseño de la interfaz buscarán minimizar estos desafíos, pero estos factores deben tenerse en cuenta al evaluar la accesibilidad y la facilidad de uso de la solución propuesta.

2. Marco y Estado del Arte del Problema

2.1. Soluciones de Nivel Profesional

En primera instancia, nos enfocaremos en las soluciones de alto nivel dirigidas a instituciones o empresas con capacidad para invertir en la solución y su mantenimiento. A través de diversas búsquedas, conocimiento previo e investigación, se llegaron a los siguientes resultados.

Estas soluciones generalmente consisten en productos ya ensamblados y terminados, equipados con componentes de alto nivel para lograr una excelente precisión y consistencia de movimiento.

En términos de soluciones completas, el rango de precios más bajo es de entre 1500 a 3000 USD para las opciones más básicas. Además, es importante considerar que la provisión de soporte o incluso la entrega de sistemas como estos suele ser compleja, ya que el alcance de estas soluciones generalmente no se extiende a países como Chile.

Los brazos robóticos con integración de software tienen un precio base de 1500 USD, siendo el mejor ejemplo y el más recomendado el de Dobot. Este incluye en su solución una plataforma de control y un SDK para desarrollo propietario. Algunas referencias acerca de las búsquedas y valores. [1], [2]

Figura 1:

Brazo robot Dobot Magician.

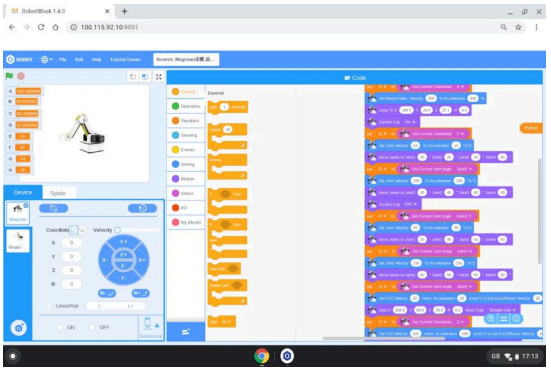

Figura 2:

Software de Control Dobot Magician.

2.2. Soluciones Open Source

Por otro lado, existen soluciones de menor costo tanto para instituciones como para individuos. Estas suelen consistir en la publicación de sets de instrucciones y archivos de diseño para ser impresos en 3D, como en los casos de MoveoRos, MoveoArm y Arctos. Generalmente, estos proyectos tienen un costo de fabricación por debajo de los 500 USD y están creados con piezas que, en su mayoría, son fáciles de conseguir e imprimir en 3D. Así, su creación y mantenimiento dependen del implementador, lo que resulta factible tanto para individuos como para instituciones.

En cuanto al control de la solución (aspecto de software), también queda completamente en manos del implementador. Cabe destacar que el precio y las funcionalidades de estas soluciones son variables debido a la naturaleza de código abierto de estos proyectos y la versatilidad de la impresión 3D. La ruta general para su construcción incluye la impresión de los modelos (esqueleto del brazo) y la compra independiente de los componentes electrónicos y motores.

Algunas soluciones de código abierto conocidas son:

- BCN3D-Moveo [3]
- THOR [4]
- Arctos [5]

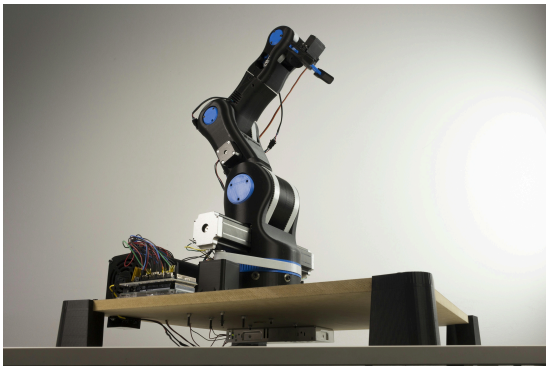

Figura 3:

Brazo robot Open Source BCN3D MOVEO. [3]

Figura 4:

Brazo robótico Arctos. [5]

2.3. Software de Control

En el contexto del estado del arte para brazos robóticos Open Source, el desafío más significativo radica en el control y la programación de estos dispositivos. La implementación efectiva de sistemas de control para brazos robóticos no solo requiere conocimientos avanzados en programación, sino también una comprensión profunda de la electrónica y la mecánica involucrada. Los usuarios deben manejar complicados algoritmos para controlar con precisión los movimientos del brazo, lo que implica una sólida base en matemáticas y lógica de programación.

Los softwares de control típicamente realizan acciones como la coordinación de movimientos articulares, la gestión de trayectorias y la manipulación de objetos con precisión. También están involucrados en la sincronización y control de actuadores y sensores para lograr movimientos fluidos y precisos. Además, estos sistemas suelen incluir funciones para la programación de tareas repetitivas y la ejecución de secuencias de movimientos complejos.

Además, la configuración y sincronización de componentes electrónicos, como motores y sensores, exigen habilidades específicas en electrónica. Por último, el diseño de una interfaz de usuario intuitiva y eficaz para la interacción con el brazo robótico es crucial, lo que añade otra capa de complejidad a estos sistemas. Estos requisitos técnicos hacen que el control de brazos robóticos de código abierto sea un área especializada, a menudo inaccesible para principiantes o entusiastas sin formación técnica avanzada.

En cuanto a las soluciones que requieren la implementación o uso de software de control, generalmente existen dos caminos: una implementación desde cero utilizando un microcontrolador y un computador para el control, o la utilización de algún framework o plataforma de control de robótica grande y conocida. Entre estas últimas, se pueden encontrar GRBL y ROS.

GRBL es un firmware de código abierto para el control de movimiento de máquinas que se mueven, como las máquinas CNC de 3 ejes. Ha sido diseñado para funcionar en microcontroladores Arduino, lo que proporciona una solución de control de bajo costo y de fácil implementación para la robótica.

ROS (Robot Operating System) es un marco flexible para escribir software de robots. Es una colección de herramientas, bibliotecas y convenciones cuyo objetivo es simplificar la tarea de crear un comportamiento de robot complejo y robusto en plataformas robóticas. [6]

Sin embargo, esta misma generalidad puede convertirse en un aspecto negativo para aquellos con menos experiencia o conocimientos específicos. ROS, al ser una plataforma

tan universal, puede resultar confuso y difícil de utilizar para principiantes o para aquellos que buscan una solución más sencilla y directa para un problema específico. Puede requerir un gran conocimiento de programación y sistemas robóticos para usarse eficazmente, y su amplio alcance puede ser abrumador para usuarios nuevos.

Por ejemplo, al realizar una búsqueda exhaustiva utilizando varias palabras clave como «software de control para brazo robótico», los resultados más relevantes únicamente contenían simulaciones y representaciones gráficas de modelos de brazos robóticos de 3 grados de libertad [7]. Es importante destacar que, aunque existen paquetes y firmwares específicos desarrollados para ciertos brazos robóticos bajo ROS, como es el caso de Jesseweisberg/moveo_ros [8] que puede ser utilizado para controlar el brazo robótico BCN3D Moveo tanto en simulación como en la vida real, no se ha encontrado una solución completa que englobe tanto el firmware como el software de control genérico. Esto resalta la necesidad de desarrollar una plataforma más accesible para el control de brazos robóticos.

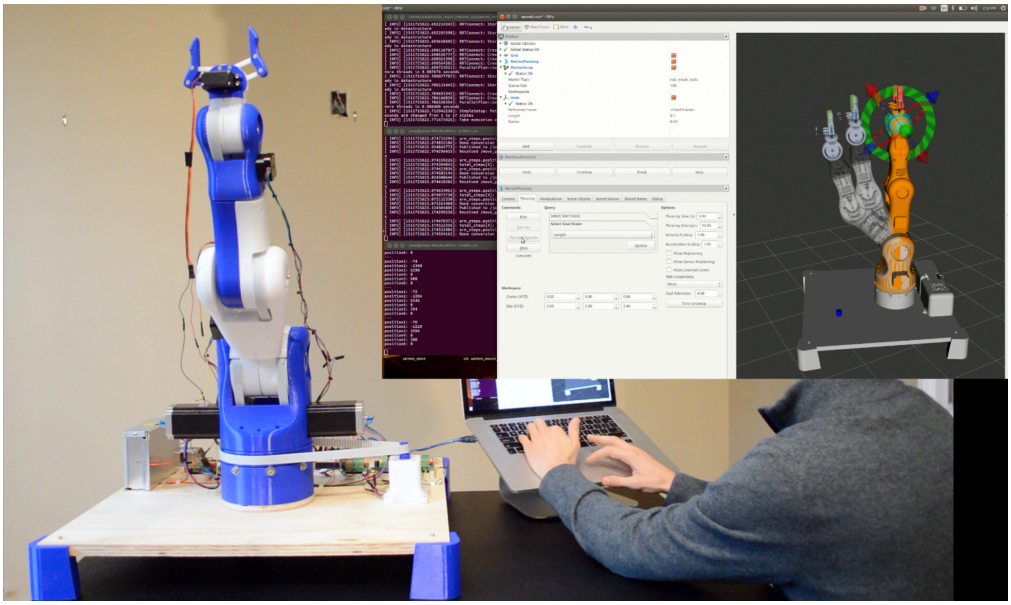

Figura 5:

Operando Moveo con ROS. [8]

2.4. La Falta de una Solución Completa y Accesible

En el análisis del estado del arte relativo al control de brazos robóticos, se ha identificado una clara carencia de soluciones que sean al mismo tiempo completas y accesibles. Las opciones existentes de alto nivel, aunque ofrecen soluciones completas, conllevan altos costos y una complejidad significativa en su mantenimiento. Este mantenimiento suele requerir asistencia directa de la marca específica, con repuestos y procedimientos de reparación que pueden ser muy complejos y costosos. Esto limita su aplicabilidad, especialmente para instituciones de menor envergadura o individuos. Además, la dependencia de una marca

específica para el mantenimiento y las actualizaciones puede resultar en una menor flexibilidad y mayores costos a largo plazo. Esta situación resalta la necesidad de desarrollar una solución que, además de ser relativamente fácil de implementar y usar, también permita la evolución dentro del control de brazos robóticos de código abierto y extensible.

Por otro lado, se observa que, aunque hay soluciones de código abierto disponibles, estas a menudo son incompletas o poco accesibles para aquellos que no poseen un conocimiento técnico avanzado. Estas soluciones suelen ser demasiado amplias y no se centran específicamente en brazos robóticos, lo que incrementa la dificultad para aquellos que buscan una solución más enfocada y accesible. Esta realidad subraya la falta de opciones que faciliten el control de brazos robóticos de manera sencilla y accesible para un público más amplio.

Las opciones actuales se pueden resumir en tres categorías principales:

1. **Sistemas de Control de Robótica General como ROS**: Aunque ROS (Robot Operating System) es capaz de controlar eficazmente un brazo robótico, su implementación y uso son complejos. Diseñado para una amplia gama de aplicaciones robóticas, ROS demanda conocimientos técnicos avanzados y una comprensión profunda de la robótica, lo que lo hace inaccesible para usuarios sin una formación técnica especializada.
2. **Adaptación de Sistemas de Control General como GRBL**: Originalmente diseñados para otras aplicaciones (como máquinas CNC), sistemas como GRBL pueden ser adaptados para controlar brazos robóticos. Sin embargo, estas adaptaciones no son soluciones ideales, ya que no están específicamente diseñadas para brazos robóticos y pueden requerir modificaciones técnicas considerables.
3. **Control Desde Cero Utilizando Arduino o Similares**: Esta opción implica desarrollar un sistema de control desde cero, utilizando plataformas como Arduino. Aunque ofrece flexibilidad, esta aproximación requiere habilidades de programación y electrónica avanzadas, y puede ser un proceso lento y laborioso.

	Sistemas Existentes	Solución Propuesta
Facilidad de Implementación	Generalmente compleja	Diseñada para ser más directa y sencilla
Conocimientos Previos	Requieren conocimientos avanzados en robótica, electrónica y programación	Básicos en programación y electrónica; más accesibles
Costo en Tiempo	Alto, debido a la necesidad de adaptación y personalización	Reducido, con una implementación más rápida
Potencial y Flexibilidad	Alto, pero puede requerir adaptaciones significativas para brazos robóticos específicos	Focalizado en brazos robóticos, con menor necesidad de adaptación
Accesibilidad para Usuarios	Limitada para usuarios no técnicos o principiantes	Mejorada para un rango más amplio de usuarios, incluyendo principiantes
Costo Económico	Puede ser alto, especialmente con sistemas profesionales	Generalmente más bajo, utilizando componentes y software de código abierto
Capacidad de Personalización	Alta, pero requiere habilidades avanzadas	Moderada, con énfasis en la facilidad de uso y accesibilidad
Integración con Otras Tecnologías	Compleja pero muy flexible	Simplificada y menos versátil

Tabla 1: Comparación de las opciones de control existentes con la solución propuesta.

La tabla comparativa resalta las diferencias entre las soluciones existentes y la solución propuesta en términos de facilidad de implementación, conocimientos previos requeridos, costo en tiempo y potencial y flexibilidad. Se observa que, mientras las soluciones existentes pueden ser difíciles de implementar, es decir, complicadas de instalar y configurar en el hardware del brazo robótico, y requieren conocimientos técnicos avanzados, además de ser costosas en términos de tiempo, la solución propuesta busca ser más simple y accesible en estos aspectos. Además, se centra exclusivamente en brazos robóticos, lo cual podría simplificar aún más el proceso de implementación en comparación con soluciones más generales.

Por tanto, se identifican tres limitaciones principales en las alternativas existentes:

- La **dificultad de implementación** de sistemas de control de robótica general, que a menudo requieren conocimientos técnicos avanzados.
- La **ausencia de una solución completa y dedicada exclusivamente a brazos robóticos**, lo que limita la eficacia de las soluciones existentes para esta aplicación específica.
- La **dificultad de uso** de estos sistemas, que excluye a usuarios sin conocimientos técnicos avanzados.

En conclusión, hay una necesidad clara de una solución de control más específica, sencilla y accesible para brazos robóticos Open Source que pueda ser utilizada eficientemente por un espectro más amplio de usuarios.

3. Fundamentos y Conceptos Clave

3.1. Modelo matemático

El modelo matemático de un brazo robótico es esencial para su diseño y operación, y consta de dos partes fundamentales: la cinemática directa y la cinemática inversa. Estos conceptos son cruciales para entender cómo un brazo robótico interactúa con su entorno y cómo se controlan sus movimientos.

La cinemática directa se refiere a la relación entre los ángulos de las articulaciones del brazo robótico y la posición de la herramienta o del efector final en el espacio. En otras palabras, partiendo del conocimiento de los ángulos en cada una de las articulaciones, la cinemática directa nos permite calcular la ubicación exacta y la orientación de la herramienta en un espacio tridimensional. Este proceso implica la utilización de ecuaciones matemáticas que describen la geometría y la mecánica del brazo, proporcionando así un mapa detallado de cómo los movimientos de las articulaciones se traducen en el posicionamiento del efector final.

Por otro lado, la cinemática inversa aborda el problema desde la perspectiva opuesta. Se centra en determinar los ángulos necesarios en las articulaciones del brazo para alcanzar una posición y orientación específicas de la herramienta en el espacio. Este proceso es más complejo que la cinemática directa, ya que a menudo implica resolver sistemas de ecuaciones no lineales. La cinemática inversa es particularmente importante en la programación y control de brazos robóticos, ya que permite al operador especificar el punto final deseado para la herramienta y luego calcular automáticamente los ángulos de articulación necesarios para lograr esa posición.

En el desarrollo de las secciones correspondientes a la cinemática directa e inversa, fue fundamental la contribución del curso “Industrial Robotics” [9] disponible en la plataforma Udemy. Este curso proporcionó una comprensión sólida de los conceptos básicos y una visión general de cómo se calculan la cinemática directa e inversa en un brazo robótico. A continuación, se detallarán las cinemáticas directa e inversa de un brazo robótico de seis grados de libertad, inspirándose en el contenido presentado en el curso mencionado. Como el curso ofrece una visión general sin entrar en detalles, hay que profundizar y presentar de manera más detallada los conceptos de cinemática directa e inversa, así como su cálculo práctico, para una comprensión más completa.

3.2. Cinemática Directa

La cinemática directa es un aspecto crucial en el control y la programación de brazos robóticos, ya que establece la relación entre los ángulos de las articulaciones y la posición de la herramienta o efector final en el espacio. El cálculo de la cinemática directa se realiza a través de la matriz de transformación homogénea, que es una herramienta matemática fundamental en la robótica y la mecánica de precisión.

La matriz de transformación homogénea es una matriz de 4x4 que representa no solo la posición, sino también la orientación de un objeto en el espacio tridimensional. Esta matriz combina una matriz de rotación 3x3, que define la orientación del objeto, y un vector de posición 3x1, que define su ubicación, en una sola entidad matemática. La última fila de esta matriz, generalmente fijada en [0 0 0 1], permite mantener la coherencia en las operaciones matemáticas.

Para calcular la posición y orientación del efector final de un brazo robótico, se procede de la siguiente manera: primero, se calcula la matriz de transformación homogénea para cada articulación del brazo. Esta matriz toma en cuenta tanto la rotación como la traslación que proporciona cada articulación al sistema. Posteriormente, estas matrices individuales se multiplican secuencialmente, comenzando por la matriz correspondiente a la base del brazo y avanzando a lo largo de cada articulación. La multiplicación secuencial de estas matrices permite incorporar el efecto acumulativo de cada movimiento de articulación en la posición y orientación finales del efector.

El resultado de este proceso es la matriz de transformación homogénea del efector final del brazo robótico. Esta matriz puede ser descompuesta para extraer la posición y la orientación específicas del efector final en el espacio. La posición se obtiene directamente de los tres primeros elementos de la última columna de la matriz, mientras que la orientación se puede deducir de los primeros tres elementos de las primeras tres columnas.

De manera específica, en la imagen que sigue se ilustran los ejes de rotación de cada articulación y las distancias que las separan. En un brazo robótico típico con seis grados de libertad, encontramos seis articulaciones distintas. Cada una de estas articulaciones se asocia con una matriz de transformación homogénea, resultando así en seis matrices de transformación homogénea para el brazo completo.

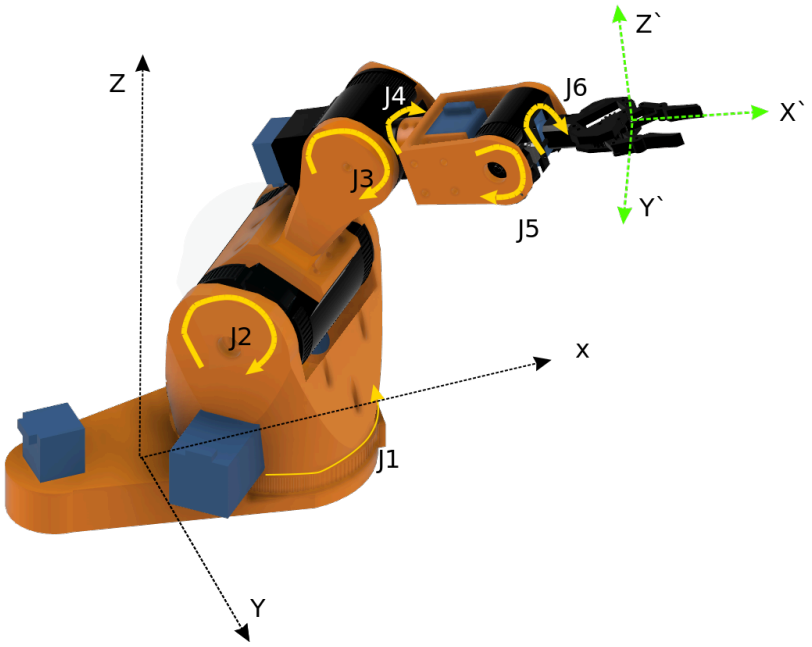

Figura 6:

Ejes de rotación de las articulaciones de un brazo robótico de 6 grados de libertad.

En la figura presentada, es posible visualizar las seis articulaciones del brazo robótico, identificadas desde J1 hasta J6.

Para comenzar con el análisis, es esencial calcular la matriz de transformación homogénea que parte de la base hasta alcanzar la primera articulación, J1. Este cálculo implica dos componentes principales: la distancia desde el punto de origen hasta el eje de rotación de J1, y la rotación propia de esta primera articulación.

En el caso específico de J1, observamos que su rotación ocurre alrededor del eje Z. Además, la distancia desde el origen hasta el eje de rotación de J1 es nula en los ejes X y Y (ya que tomamos como origen este), y presenta un valor específico, denominado a1z, en el eje Z.

Para la construcción de la matriz de transformación homogénea, es fundamental entender primero cómo se configura la matriz de rotación en torno al eje Z.

$$R_1 = \begin{pmatrix} \cos(j_1) & -\sin(j_1) & 0 \\ \sin(j_1) & \cos(j_1) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

El ángulo de rotación de la primera articulación se representa como j_1

En el contexto de las matrices de transformación homogénea, que combinan tanto rotación como traslación, la estructura general de estas matrices se presenta de la siguiente manera:

$$T = \begin{pmatrix} \text{Rotacion} & ... & ... & \text{Traslacion} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

En este contexto, «Rotación» corresponde a una matriz de rotación 3x3, y «Traslación» a un vector de traslación 3x1.

Con estos elementos, podemos construir la matriz de transformación homogénea siguiente:

$$T_1 = \begin{pmatrix} \cos(j_1) & -\sin(j_1) & 0 & 0 \\ \sin(j_1) & \cos(j_1) & 0 & 0 \\ 0 & 0 & 1 & a_1 z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Para facilitar la notación en las matrices subsiguientes, es importante reconocer que las matrices de rotación para cada eje son las siguientes

$$R_{x(\theta)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} R_{y(\theta)} = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} R_{z(\theta)} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A continuación, definimos que el vector de traslación a_i correspondiente a cada articulación i es:

$$a_i = \begin{pmatrix} a_i x \\ a_i y \\ a_i z \end{pmatrix}$$

$$\mathrm{Th}(R, A) = \begin{pmatrix} R & ... & ... & A \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Donde A representa el vector de traslación de la articulación y R es la matriz de rotación de la articulación.

Con esta definición, podemos reescribir la matriz de transformación homogénea de la primera articulación de la siguiente manera:

Base a Articulacion 1

$$T_1 = \begin{pmatrix} R_{z(j_1)} & ... & ... & a_1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathrm{Th}\Big(R_{z(j_1)}, a_1\Big)$$

Articulacion 1 a Articulacion 2

Aquí, es importante observar que tenemos una rotación alrededor del eje Y.

$$T_2 = \begin{pmatrix} R_{y(j_2)} & ... & ... & a_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathrm{Th}\Big(R_{y(j_2)}, a_2\Big)$$

Así, se procede sucesivamente hasta alcanzar la última articulación. Así, llegamos a la matriz de transformación homogénea del efector final. Es relevante destacar que para la articulación 1, la rotación se realiza en torno al eje Z; para las articulaciones 2, 3 y 5, en torno al eje Y; y para las articulaciones 4 y 6, en torno al eje X. Por tanto, podemos obtener la matriz de transformación homogénea del efector final multiplicando secuencialmente las matrices de transformación homogénea de cada una de las articulaciones.

$$T_{\text{final}} = \mathrm{Th}\Big(R_{z(j_1)}, a_1\Big)\ \mathrm{Th}\Big(R_{y(j_2)}, a2\Big)\ \mathrm{Th}\Big(R_{y(j_3)}, a3\Big)$$
$$\mathrm{Th}\Big(R_{x(j_4)}, a4\Big)\ \mathrm{Th}\Big(R_{y(j_5)}, a5\Big)\ \mathrm{Th}\Big(R_{x(j_6)}, a6\Big)$$

Luego para obtener la posición y orientación del efector final, se extraen los valores de la matriz de transformación homogénea del efector final.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = T_{\text{final}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Luego para encontrar la orientación del efector final, se extraen los valores de la matriz de transformación homogénea del efector final. La forma más simple de esto es a partir de la matriz de rotación, la cual se obtiene de la matriz de transformación homogénea del efector final, extrayendo los valores de las primeras tres columnas.

$$R = \begin{pmatrix} T_{\text{final}}11 & T_{\text{final}}12 & T_{\text{final}}13 \\ T_{\text{final}}21 & T_{\text{final}}22 & T_{\text{final}}23 \\ T_{\text{final}}31 & T_{\text{final}}32 & T_{\text{final}}33 \end{pmatrix}$$

Luego para obtener los ángulos de Euler, se puede usar la siguiente formula [10]:

$$\text{roll} = \text{atan2}(-T_{\text{final}}32, T_{\text{final}}33)$$

$$\text{pitch} = \text{asin}(T_{\text{final}}31)$$

$$\text{yaw} = \text{atan2}(-T_{\text{final}}21, T_{\text{final}}11)$$

Un caso borde ocurre cuando $R_{31} = 1 \vee R_{31} = -1$, en este caso $\text{pitch} = \pm\frac{\pi}{2}$. En este caso

se puede fijar el valor de $\text{yaw} = 0$ y calcular el valor de roll a partir de la matriz de rotación de la siguiente forma:

$$\text{roll} = \text{atan2}(T_{\text{final}}12, T_{\text{final}}13)$$

$$\text{pitch} = \text{asin}(T_{\text{final}}31)$$

$$\text{yaw} = 0$$

3.3. Cinemática Inversa

La cinemática inversa en el contexto de brazos robóticos es el proceso de determinar los ángulos de las articulaciones requeridos para alcanzar una posición y orientación específicas del efector final o herramienta en el espacio. A diferencia de la cinemática directa, que parte de los ángulos de las articulaciones para encontrar la posición de la herramienta, la cinemática inversa comienza con una posición cartesiana deseada (x, y, z) y una orientación específica (roll, pitch, yaw) y trabaja de manera inversa para calcular los ángulos necesarios en las articulaciones.

Para simplificar este cálculo, muchos brazos robóticos se diseñan de manera que los ejes de rotación de sus últimas tres articulaciones (comúnmente referidas como la muñeca del brazo robótico) coincidan en un punto único, esto facilita el cálculo ya que de esta forma la ubicación de la muñeca del brazo robótico depende únicamente de las primeras tres articulaciones. Así esta configuración permite separar el problema de la cinemática inversa en dos partes más manejables:

Calcular la Posición del Punto de Intersección: La primera tarea consiste en determinar la posición del punto donde convergen los ejes de rotación de las últimas tres articulaciones. Esta posición se calcula en función de la posición deseada de la herramienta y su orientación, teniendo en cuenta la longitud y geometría del brazo. Se busca el punto donde la muñeca del brazo debería estar para que la herramienta alcance la posición y orientación deseadas.

Calcular los Ángulos de las Primeras Tres Articulaciones: Una vez identificada la posición del punto de intersección, el siguiente paso es calcular los ángulos de las primeras tres articulaciones del brazo (que comúnmente corresponden a la base, el hombro y el codo del brazo robótico). Estos cálculos permitirán posicionar correctamente el punto de intersección de las últimas tres articulaciones. Este proceso implica resolver ecuaciones matemáticas que relacionan la geometría del brazo con la posición deseada, lo que puede ser un reto debido a la naturaleza no lineal y a veces redundante de los sistemas robóticos.

Ahora, para realizar el cálculo de la cinemática inversa, es fundamental calcular primero la posición del punto de intersección. Esto implica determinar la ubicación de la última articulación del brazo robótico y, posteriormente, calcular la posición de la muñeca del mismo.

Considerando una configuración espacial conocida como TP, definida por las coordenadas (x, y, z) y una orientación específica descrita por los ángulos “roll”, “pitch” y “yaw”, nuestro objetivo es identificar la ubicación exacta del punto denominado WP (punto de la muñeca).

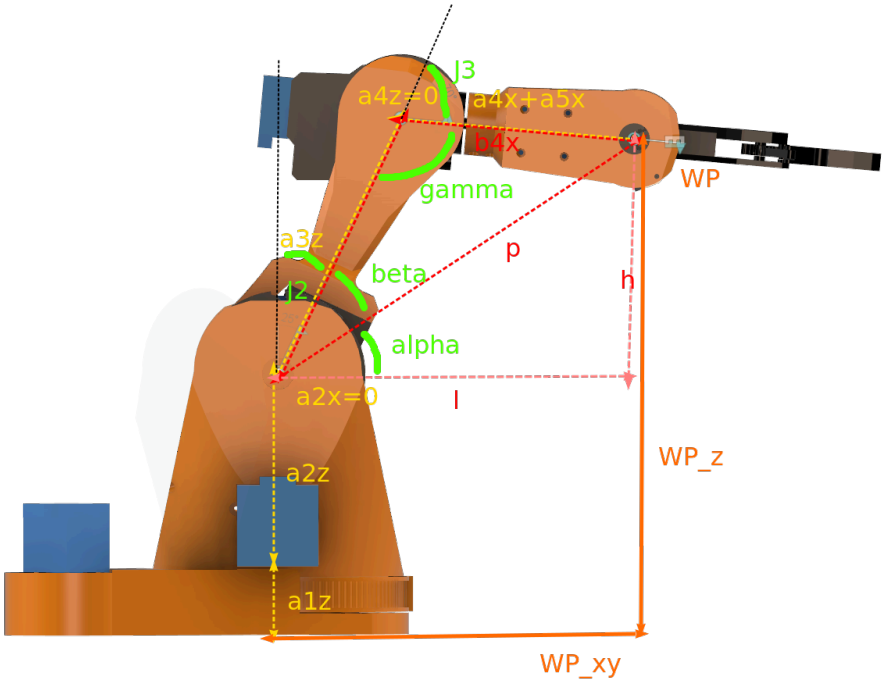

Figura 7:

Vista lateral para el cálculo de la posicion de la muñeca del brazo robótico.

$$\text{WP} = \text{TP} - a6x\hat{x}$$

Esto corresponde a la posición desde las coordenadas (x, y, z) de TP menos la distancia a_6x en la dirección del marco de referencia de la última articulación.

Para encontrar el valor de $\hat{x}$, debemos calcular la matriz de rotación que corresponde a la orientación de la herramienta, para esto se utilizan los ángulos de roll, pitch, yaw para generar una matriz de rotación. (Estos igual son conocidos como ángulos de Euler)

$$R = R_{z(\text{yaw})} x R_{y(\text{pitch})} x R_{x(\text{roll})}$$

Luego podemos extraer la primera columna de esta matriz para obtener la dirección del eje X del marco de referencia de la herramienta.

$$\hat{x} = R \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Con lo que ya tenemos la posición de la muñeca del brazo robótico, ahora debemos calcular los ángulos de las primeras tres articulaciones.

Articulación 1

Se puede notar que al realizar cualquier movimiento de la articulación 1, la posición de la muñeca del brazo robótico en el eje Z no cambia. Por lo tanto, se puede reducir el problema a un plano 2D en el eje X y Y. Con esto:

$$j_1 = \text{atan2}(\text{WP}y, \text{WP}x)$$

Aquí se destaca el uso de la función «atan2», la cual retorna el ángulo entre el eje X y el vector «WP». Esta función se elige por su mejor estabilidad numérica y se usa en el cálculo de cinemática inversa y en robótica en general.

Existe un caso borde donde $\text{WP}x = 0$ y $\text{WP}y = 0$. En esta situación, cualquier valor de j_1 es válido. Por lo general, se opta por utilizar el valor actual del ángulo de la articulación 1.

Articulaciones 2 y 3

De manera similar a la articulación 1, se observa que al mover las articulaciones 2 y 3, la muñeca del brazo robótico se desplaza en un plano 2D. Esto se debe a que las articulaciones 2 y 3 están alineadas en el mismo plano.

Con esta observación, es posible construir una serie de triángulos y calcular los ángulos de las articulaciones 2 y 3.

- $\text{WPxy} = \sqrt{\text{WP}x^2 + \text{WP}y^2}$: Esta es la distancia desde el origen hasta la muñeca del brazo robótico en el plano 2D (X,Y).

Luego, es posible formar un triángulo entre los puntos WP, Articulación 2 y Articulación 3.

Los lados de este triángulo son a3z, ρ y b4x.

Aquí, a3z es un valor conocido, ρ es la distancia entre la articulación 2 y el punto WP,

y b4x es la distancia entre la articulación 3 y el punto WP.

Para calcular ρ, necesitamos determinar h y l, que corresponden a la distancia en el eje Z entre la articulación 2 y el WP , y la distancia entre el WP y la articulación 2 en el plano XY, respectivamente.

$$h = \mathrm{WP}z - \mathrm{a1z} - \mathrm{a2z}$$

$$l = \mathrm{WPxy} - \mathrm{2ax}$$

De esta forma obtenemos ρ

$$\rho = \sqrt{h^2 + l^2}$$

Luego para obtener b4x utilizamos las medidas físicas del brazo:

$$\mathrm{b4x} = \sqrt{\mathrm{a4z}^2 + (\mathrm{a4z} + \mathrm{a5x})^2}$$

Aquí es donde podemos observar uno de los indicadores de que una posición está fuera de alcance para el brazo robótico, cuando el valor de ρ resulta ser mayor que la suma del resto de los lados del triángulo formado, por lo que es claro ver que:

$\rho < \mathrm{a3z} + \mathrm{b4x}$

Ahora que ya tenemos las dimensiones del triángulo generado, podemos encontrar los valores de j2 y j3, para esto primero encontramos valores de ángulos auxiliares.

- $\alpha = \mathrm{atan2}(h, l)$
- $\delta = \mathrm{atan2}(\mathrm{a4x} + \mathrm{a5x}, \mathrm{a4z})$
- $\cos(\beta) = \frac{\rho^2 + \mathrm{a3z}^2 - \mathrm{b4z}^2}{2\rho\ \mathrm{a3z}}$
- $\cos(\gamma) = \frac{\mathrm{a3z}^2 + \mathrm{b4x}^2 - \rho^2}{2\rho\ \mathrm{a3z}\ \mathrm{b4x}}$

Con esto tenemos que β y γ son:

$$\beta = \mathrm{atan2}\left(\sqrt{1 - \cos(\beta)^2}, \cos(\beta)\right)$$

$$\gamma = \mathrm{atan2}\left(\sqrt{1 - \cos(\gamma)^2}, \cos(\gamma)\right)$$

Así podemos obtener el valor de j_2 y j_3

$$j_2 = \frac{\pi}{2} - \alpha - \beta$$

$$j_3 = \pi - \gamma - \delta$$

Articulaciones 4, 5 y 6

Ahora que tenemos los valores de j_1, j_2, y j_3, podemos calcular la posición de la articulación 4, 5 y 6.

Sabemos desde la cinemática directa que para obtener la matriz de rotación de la última articulación debemos multiplicar las matrices de rotación de cada articulación. En este caso tenemos las matrices de rotación de las primeras tres articulaciones, la multiplicación de estas nos da la matriz de rotación R_{arm}.

$$R_{\text{arm}} = R_{z(j_1)} R_{y(j_2)} R_{y(j_3)}$$

Luego notamos que a partir de los ángulos de aproximación (roll,pitch y yaw) ya tenemos R la matriz de rotación de la articulación final, con lo que se tiene que:

$$R_{\text{arm}} R_{\text{wrist}} = R$$

$$R_{\text{wrist}} = R_{\text{arm}}^{-1} R$$

Como las matrices de rotación son ortogonales, la inversa de una matriz de rotación es su transpuesta, por lo que:

$$R_{\text{wrist}} = R_{\text{arm}}^{T} R$$

Luego notamos que R_{wrist} es generado por las rotaciones de las articulaciones 4, 5 y 6, es decir:

$$R_{\text{wrist}} = R_{x(j_4)} R_{y(j_5)} R_{x(j_6)} = R_{\text{arm}}^{T} R$$

$$R_{\text{wrist}} = \begin{pmatrix} c(j_5) & s(j_5)s(j_6) & s(j_5)c(j_6) \\ s(j_4)c(j_5) & c(j_4)c(j_6) - s(j_4)c(j_5)s(j_6) & -c(j_4)s(j_6) - s(j_4)c(j_5)c(j_6) \\ -c(j_4)s(j_5) & s(j_4)c(j_6) + c(j_4)c(j_5)s(j_6) & c(j_4)c(j_5)c(j_6) - s(j_4)s(j_6) \end{pmatrix}$$

Donde $c(x) = \cos(x)$ y $s(x) = \sin(x)$

A partir de esto el valor de j_5 es directo

- $\cos(j_5) = R_{\text{wrist}_{1,1}}$
- $j_5 = \text{atan2}\left(\pm\sqrt{1-\cos(j_5)^2}, \cos(j_5)\right)$

Luego de tener fijo el valor de j_5 podemos obtener los valores de j_4 y j_6

- $j_4 = \text{atan2}\left(\pm R_{\text{wrist}_{2,1}}, \mp R_{\text{wrist}_{3,1}}\right)$
- $j_6 = \text{atan2}\left(\pm R_{\text{wrist}_{1,2}}, \pm R_{\text{wrist}_{1,3}}\right)$

Existe un caso borde donde $j_5 = 0$, esto genera que los ejes de rotación de la articulación 4 y 6 se alinean, por lo que el valor de j_4 y j_6 generando infinitas soluciones, para lidiar con esto podemos fijar un valor para j_4 (por lo general su valor actual) y calcular j_6 como sigue:

- $j_6 = \text{atan2}\left(R_{\text{wrist}_{3,2}}, \pm R_{\text{wrist}_{3,3}}\right) - j_4$

3.4. Modelo Físico

El diseño de brazos robóticos impresos en 3D representa una intersección entre la robótica y la manufactura aditiva. Estos brazos se componen fundamentalmente de tres componentes clave: articulaciones, motores y sensores. Las articulaciones, impresas en 3D, suelen constar de dos partes principales: una fija y otra móvil. Este diseño modular permite la creación de brazos robóticos con distintos grados de libertad y capacidades de movimiento.

La implementación de solución, tanto en la librería como en el firmware, debe tener en cuenta estos componentes básicos. Por ejemplo, la librería de control en Python deberá ser capaz de enviar comandos precisos a los motores para manejar el movimiento de las articulaciones, teniendo en cuenta las características específicas de cada una, como su rango de movimiento y la relación entre los comandos enviados y la respuesta física real.

El firmware programado en el microcontrolador ESP32 debe interpretar estos comandos y actuar sobre los motores de manera eficiente, asegurando que el brazo se mueve según las instrucciones recibidas. Además, deberá gestionar la información proveniente de los sensores, que proporcionan datos cruciales sobre la posición y el estado de las distintas partes del brazo, permitiendo ajustes y correcciones en tiempo real para garantizar la precisión y eficacia del movimiento.

3.4.1. Motores

En la construcción de brazos robóticos, la selección del tipo de motor es crucial y depende en gran medida del tamaño y la aplicación específica del brazo. Dos tipos comunes de motores utilizados son los motores paso a paso y los servomotores.

Motores Paso a Paso: Estos motores son preferidos para brazos robóticos más grandes por su capacidad de dar un control de movimiento muy preciso. Los motores paso a paso se caracterizan por su capacidad para avanzar en pasos discretos, lo que permite un control detallado de la posición y la velocidad. Sin embargo, esta precisión viene con un costo más elevado y una mayor necesidad de hardware complementario. Además, su implementación puede ser más compleja, ya que requieren drivers específicos que manejan dos señales digitales esenciales: una para controlar la dirección y otra para controlar el paso del motor.

Servomotores: Por otro lado, los servomotores son una opción más económica y requieren menos hardware adicional, lo que los hace ideales para brazos robóticos de menor tamaño. Aunque no ofrecen la misma precisión que los motores paso a paso, los servomotores son suficientemente precisos para muchas aplicaciones y son más fáciles de integrar en proyectos con restricciones presupuestarias. Su simplicidad y menor costo los hacen muy populares en el ámbito educativo y en proyectos personales de robótica.

En resumen, la elección entre motores paso a paso y servomotores depende de varios factores, incluyendo el tamaño del brazo robótico, el nivel de precisión requerido y el presupuesto disponible. En el desarrollo de la solución, la librería y el firmware deben interactuar eficientemente con el motor seleccionado, asegurando un control preciso y fiable del brazo robótico en todas sus operaciones.

3.4.2. Sensores

Entre los sensores más comunes están los de fin de carrera, usados en impresoras 3D, y otros tipos de sensores capaces de devolver valores útiles para determinar la posición y estado de las articulaciones del brazo robótico.

Sensores de Fin de Carrera: Estos sensores son utilizados para indicar la posición límite de un movimiento, actuando como un interruptor que se activa cuando una parte móvil del brazo robótico alcanza un punto específico. En impresoras 3D, por ejemplo, se utilizan para calibrar el inicio del eje. En los brazos robóticos, estos sensores pueden ayudar a prevenir movimientos que excedan los límites físicos de la máquina, protegiendo así el hardware de posibles daños.

Sensores en Servomotores: En el caso de los servomotores, un aspecto beneficioso es que generalmente incluyen un sensor de posición integrado. Este sensor permite que el servomotor tenga retroalimentación sobre su posición actual, facilitando el control preciso de la articulación a la que está conectado. La presencia de este sensor integrado elimina la necesidad de agregar sensores adicionales para determinar la posición de las partes del brazo controladas por estos servomotores.

Para implementar la solución, la librería y el firmware deben poder integrar y procesar la información proporcionada por estos sensores. Esto es esencial para garantizar un control preciso y seguro del brazo robótico. En el caso de los servomotores, la librería deberá aprovechar los sensores integrados para realizar ajustes y correcciones en tiempo real. Mientras que para los sensores de fin de carrera y otros sensores externos, será necesario desarrollar una estrategia de integración que permita al firmware interpretar correctamente sus señales y ajustar el comportamiento del brazo en consecuencia.

4. Solución

4.1. Diseño de la Solución

En el diseño de la solución para controlar un brazo robótico, se tomaron en cuenta requisitos clave que alinean la propuesta con los objetivos establecidos. Estos requisitos fueron esenciales para guiar las decisiones de diseño en hardware, software y en la experiencia del usuario con la interfaz. A continuación, se detallan los criterios principales que se tuvieron en cuenta:

1. **Facilidad de Uso:** Se puso un énfasis especial en asegurar que todos los aspectos de la solución, desde el hardware hasta el software, fueran fáciles de utilizar. Esta accesibilidad se extendió a la experiencia del usuario con la interfaz, buscando que fuera intuitiva y no requiriera conocimientos previos en robótica o programación.
2. **Minimización de Dependencias:** Para facilitar la instalación y el uso, se decidió que la arquitectura debería ser totalmente compatible con Docker. Esto permitiría que la solución se ejecutase en un entorno controlado, minimizando la necesidad de configuraciones complejas o la instalación de múltiples dependencias.
3. **Modularidad:** La solución fue diseñada para ser extensible y modificable con relativa facilidad. Esto no solo facilita futuras mejoras y personalizaciones por parte de los usuarios, sino que también permite adaptar la solución a diferentes tipos de hardware o requerimientos específicos.
4. **Tecnologías Web para la Interfaz:** Se optó por utilizar tecnologías web para la interfaz de usuario, con el objetivo de que fuera accesible desde cualquier dispositivo equipado con un navegador web. Esto elimina la necesidad de que los usuarios finales instalen software adicional, mejorando la accesibilidad y la comodidad de uso.
5. **Uso de Python para la Biblioteca de Control:** Se eligió Python para el desarrollo de una biblioteca de control debido a su simplicidad y accesibilidad. La biblioteca fue diseñada para ser fácil de usar, incluso para aquellos sin conocimientos avanzados de programación, lo que alinea la solución con el objetivo de accesibilidad y facilidad de uso.

Estos requisitos fundamentales guiaron la creación de una arquitectura que respondiera a las necesidades tanto de usuarios novatos como de aquellos con experiencia en el campo de la robótica. La estructura general de esta arquitectura se describe en detalle en la siguiente sección del informe, donde se abordan los componentes específicos y su interacción dentro del sistema.

4.2. Arquitectura General

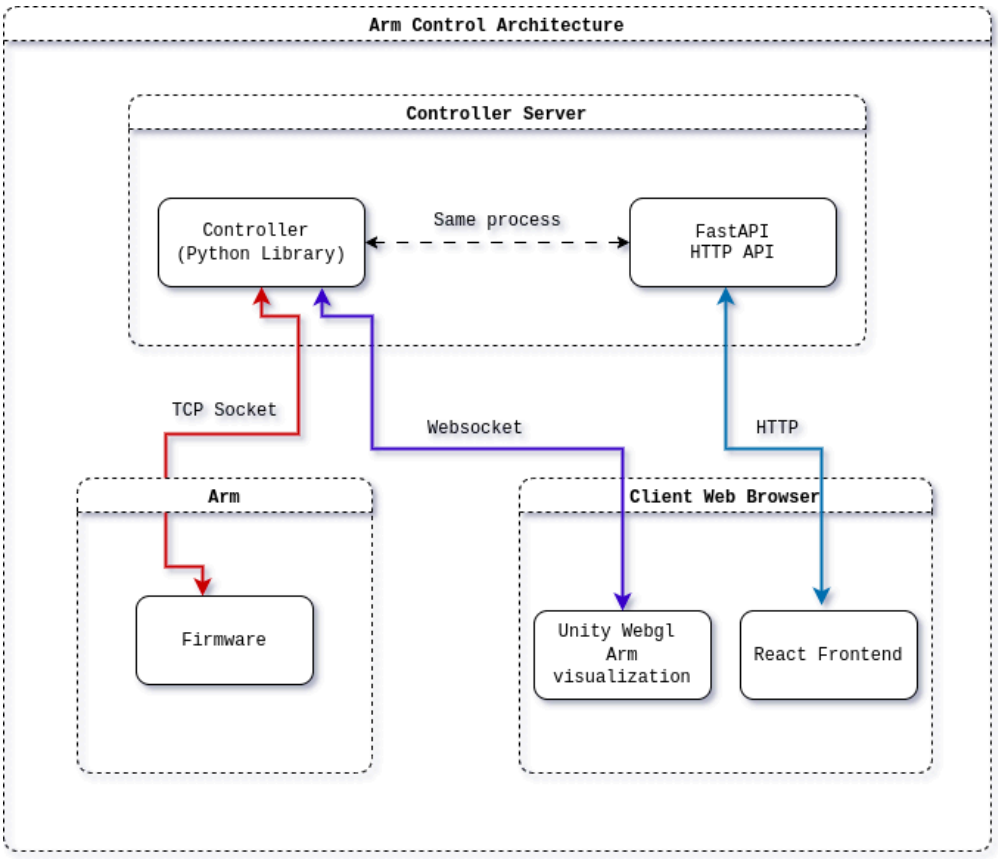

Figura 8:

Diagrama general de la arquitectura de la solución.

La solución propuesta para el control de un brazo robótico se articula alrededor de tres servicios principales, interconectados a través de una infraestructura basada en Internet, lo que facilita una comunicación fluida y eficiente entre los diferentes componentes del sistema.

Interfaz Web: Constituye el cliente que se ejecuta en un navegador web. Esta interfaz es el principal medio de interacción del sistema para el usuario, ofreciendo una plataforma interactiva y accesible para controlar el brazo robótico. La interfaz web tiene la funcionalidad de manipular el brazo robótico, permitiendo al usuario realizar movimientos libres, secuencias y rutinas complejas de manera intuitiva. Se comunica con el backend a través de una API HTTP y utiliza websockets dentro de una visualización creada usando Unity para interactuar directamente con el controlador del brazo robótico.

Librería de Control: La librería de control, escrita en Python, es el núcleo que gestiona la lógica de operación del brazo, traduciendo las instrucciones del usuario en comandos específicos para el hardware.

Firmware del Brazo Robótico: Es el software que se ejecuta directamente en el microcontrolador del brazo robótico, como un ESP32. Este firmware es responsable de interpretar y ejecutar los comandos recibidos desde el controlador en la librería. La comunicación entre la librería de control y el firmware se realiza a través de sockets TCP, asegurando una transmisión de datos fiable y eficiente, fundamental para el control preciso de movimientos y operaciones del brazo.

Estos tres servicios forman una solución integrada que permite controlar un brazo robótico eficazmente. La utilización de tecnologías basadas en Internet para la comunicación entre los diferentes elementos del sistema proporciona una gran flexibilidad y posibilita la integración y escalabilidad futuras del proyecto.

4.3. Metodología de Desarrollo

El desarrollo del proyecto para controlar un brazo robótico se llevó a cabo siguiendo una metodología iterativa, dividida en tres iteraciones principales, cada una centrada en aspectos específicos del sistema y su integración. Este enfoque permitió una evolución gradual y controlada del proyecto, facilitando la adaptación y mejora continua del sistema.

Primera Iteración - Creación de la Librería y el Firmware:

Esta fase inicial se centró en el desarrollo de la librería de control y el firmware correspondiente. La librería, programada en Python, estableció la lógica fundamental para el manejo del brazo robótico, mientras que el firmware, implementado en el microcontrolador ESP32, se encargó de la ejecución directa de los comandos. Esta etapa puso las bases para la posterior implementación de la interfaz de usuario.

Segunda Iteración - Desarrollo de la Interfaz:

Paralelamente a la evolución de la librería, se trabajó en la creación de la interfaz web. Esta interfaz se diseñó para proporcionar una experiencia de usuario intuitiva y accesible, facilitando la interacción con el brazo robótico. La simultaneidad del desarrollo de la librería y la interfaz permitió un ajuste y sincronización eficientes entre ambos componentes.

Tercera Iteración - Integración y Pulido:

En esta etapa final, se integraron todos los servicios - la librería, la interfaz y el firmware y se realizaron ajustes y mejoras para asegurar una operación fluida y eficiente del sistema en conjunto. Se prestó especial atención a la implementación en hardware, asegurando que el sistema funcionara correctamente en un entorno real.

Durante la transición entre la segunda y la tercera iteración, se llevaron a cabo pruebas de usabilidad a pequeña escala, lo que proporcionó información valiosa para el refinamiento continuo del sistema. Tras completar la tercera iteración, se realizó una prueba de usabilidad más amplia, cuyos detalles y resultados se expondrán en el apartado de evaluación de la solución. Los comentarios y datos recopilados durante esta prueba condujeron a pequeñas mejoras y ajustes en el sistema, aunque no se realizaron cambios de gran envergadura.

Este proceso iterativo y las pruebas de usabilidad ademas de permitir desarrollar un sistema funcional y eficaz, también permitieron adaptar y mejorar continuamente la solución para satisfacer mejor las necesidades de los usuarios y los requisitos del proyecto.

Durante el desarrollo de la solución, se adoptaron metodologías de integración y testing orientadas a maximizar la eficiencia, la consistencia y la calidad del software. A continuación, se describen los aspectos clave de estas metodologías:

Docker como Herramienta Central:

Docker se utilizó ampliamente en el desarrollo de todos los servicios. Su uso facilita la creación de entornos de desarrollo y pruebas consistentes y reproducibles, lo que garantiza la compatibilidad y el correcto funcionamiento del sistema en plataformas y configuraciones.

Esta decisión también influyó en la elección de basar toda la comunicación en protocolos de internet (TCP, HTTP, Websockets), ya que estos son fácilmente manejables y compatibles con contenedores Docker.

Estándares de Código y Consistencia

Se implementaron linters y formatters para todos los servicios, asegurando así la consistencia y la calidad del código. Esto ayuda a mantener un estándar alto en la codificación y facilita la colaboración y el mantenimiento del código a largo plazo.

Conjunto de Tests:

Tanto para la librería de control como para el firmware, se desarrolló un conjunto de tests que validan el funcionamiento de los algoritmos y la comunicación entre los servicios. La capacidad de ejecutar el firmware en Docker ofrece una ventaja significativa, permitiendo realizar pruebas exhaustivas en un entorno controlado y simulado.

Pipelines CI con GitHub Actions:

Se configuraron pipelines de integración continua (CI) utilizando GitHub Actions. Estos pipelines automatizan la validación de que el código cumple con los estándares de calidad y que todos los tests pasan satisfactoriamente.

Como parte del proceso de CI, se suben las imágenes de los contenedores Docker de cada servicio a Docker Hub. Asimismo, la librería de control se publica como un paquete de Python en PyPI, facilitando su distribución y accesibilidad.

Documentación y Código Fuente:

Junto con el informe, se creó una página de documentación con más detalles sobre el uso del sistema de control [11]. Esta documentación está diseñada para ser una guía

completa y accesible, ofreciendo información detallada y consejos prácticos para usuarios y desarrolladores interesados en el sistema. Asimismo, el código fuente del proyecto está disponible en GitHub, lo que permite a cualquier interesado revisar, descargar y contribuir al desarrollo del sistema [12]. La combinación de esta documentación detallada y el acceso abierto al código fuente asegura que el sistema sea transparente, comprensible y fácilmente adaptable o ampliable por la comunidad.

4.4. Librería y Firmware

La solución desarrollada para el control del brazo robótico se fundamenta en la estrecha colaboración entre dos componentes cruciales: la librería de control y el firmware. Ambos elementos desempeñan roles complementarios y son esenciales para el adecuado funcionamiento del sistema.

Librería de Control: Esta librería, desarrollada en Python, alberga toda la lógica de control del brazo robótico. Incluye algoritmos complejos para calcular la cinemática directa e inversa y determinar los ángulos apropiados a los que deben moverse las articulaciones. Esta librería además de gestionar los movimientos del brazo, también se encarga de interpretar las instrucciones del usuario y transformarlas en comandos específicos que pueden ser ejecutados por el brazo.

Firmware: Por otro lado, el firmware, implementado en el microcontrolador del brazo robótico, se centra en la ejecución de los comandos recibidos de la librería. Su principal tarea es controlar físicamente el brazo, moviendo las articulaciones y ajustando su posición según las directrices recibidas. Este componente es crítico para garantizar que los movimientos del brazo sean precisos y acordes con los cálculos realizados por la librería.

Ambos, la librería de control y el firmware mantienen una relación de dependencia mutua. Mientras que la librería planifica y ordena las acciones, el firmware las lleva a cabo en el mundo real.

La integración de estos dos componentes asegura que todos los aspectos del control del brazo robótico, desde los cálculos matemáticos hasta la ejecución física de los movimientos, sean manejados de manera cohesiva y efectiva. Esta colaboración entre la librería y el firmware es fundamental para lograr un control preciso y fiable del brazo robótico en diversas aplicaciones.

La interacción entre la librería de control es una parte crucial de la solución, y se gestiona a través de una comunicación basada en mensajes TCP. Esta comunicación sigue un protocolo de mensajes bien definido, que será descrita detalladamente en la sección correspondiente del protocolo de comunicación.

1. **Comunicación TCP y Protocolo de Mensajes**: La elección de TCP para la comunicación entre la librería y el firmware asegura una transferencia de datos confiable y ordenada. El protocolo de mensajes definido facilita la interpretación y ejecución precisa de los comandos, lo que es fundamental para el control en tiempo real del brazo robótico.

2. **Uso de Python para la Librería de Control**: La decisión de utilizar Python para desarrollar la librería de control se basa en varias ventajas clave de este lenguaje. Python es conocido por su simplicidad y legibilidad, lo que lo hace accesible para programadores de todos los niveles, incluyendo aquellos en el ámbito educativo. Además, su amplia gama de bibliotecas y su capacidad para manejar cálculos matemáticos complejos lo hacen ideal para el desarrollo de algoritmos de control de brazos robóticos.
3. **Selección de la ESP32 para el Firmware**: La ESP32 fue elegida como el microcontrolador principal debido a su bajo costo, capacidades integradas de conectividad Wi-Fi y facilidad de uso. Estas características hacen que la ESP32 sea una opción excelente para proyectos de robótica que requieren conectividad inalámbrica y un buen equilibrio entre rendimiento y costo.
4. **Desarrollo del Firmware con ESP-IDF y C++**: Para la creación del firmware, se utilizó ESP-IDF, el framework oficial de desarrollo para los microcontroladores ESP32. El lenguaje de programación elegido fue C++, ya que es el principal soportado por ESP-IDF. Esta combinación permite un control detallado y eficiente del microcontrolador, algo esencial cuando se requiere manejar tareas en paralelo, como la comunicación simultánea con la librería de control y el control de los motores del brazo robótico.

4.4.1. Protocolo de Comunicación

El protocolo de comunicación entre la librería y el firmware utiliza sockets TCP y se caracteriza por su estructura simple y eficiente. Este protocolo garantiza un intercambio claro y estructurado de información entre los dos elementos del sistema, facilitando su interpretación y ejecución.

Los mensajes transmitidos entre la librería y el firmware siguen una estructura uniforme compuesta por cuatro partes principales:

- **op (Operación):** Un carácter (1 byte) que especifica la operación a realizar.
- **code (Código):** Un entero (4 bytes) que identifica el código de la operación.
- **num_args (Número de Argumentos):** Un entero (4 bytes) que indica el número de argumentos.
- **args (Argumentos):** Una serie de flotantes, cada uno de 4 bytes, que representan los argumentos de la operación.

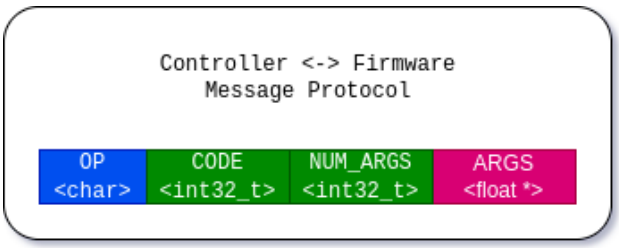

Figura 9:

Protocolo de comunicación entre la librería y el firmware.

Operaciones Básicas:

- **Config (C):** Utilizada para configurar el brazo robótico. Ejemplos incluyen ajustar la velocidad, la aceleración, entre otros parámetros.
- **Move (M):** Empleada para ordenar movimientos del brazo robótico, como mover una articulación específica o todas las articulaciones simultáneamente.
- **Status (S):** Sirve para obtener información sobre el estado actual del brazo robótico, como los ángulos de las articulaciones, el estado de homing, etc.

Ejemplos de Mensajes:

Status

S0 y S1: Mensajes relacionados con el estado del brazo robótico. S0 es enviado por el controlador para solicitar el estado, mientras que S1 es la respuesta del firmware con la información solicitada.

- op = “S”, code = 0, num_args = 0, args = []
- op: “S”, code: 1, num_args: 8, args: [j1, j2, j3, j4, j5, j6,is_homed,tool_value]

Config

C5, C7 y C6: Mensajes vinculados a la configuración de la velocidad del brazo. C5 es una solicitud del controlador para establecer la velocidad, C7 para pedir la velocidad actual y C6 es la respuesta del firmware.

- op: “C”, code: 5, num_args: 1, args: [speed]
- op: “C”, code: 7, num_args: 0, args: []
- op: “C”, code: 6, num_args: 1, args: [speed]

Move

M1: Mensaje para mover todas las articulaciones a ciertos ángulos especificados en los argumentos.

M3: Mensaje para inicializar las articulaciones y encontrar su punto de origen o «home».

- op: “M”, code: 1, num_args: 6, args: [j1, j2, j3, j4, j5, j6]
- op: “M”, code: 3, num_args: 0, args: []

Cada uno de estos mensajes sigue la misma estructura, lo que simplifica el proceso de parseo y envío, y permite una interacción fluida y eficiente entre la librería y el firmware.

Este diseño de protocolo contribuye significativamente a la robustez del sistema de control del brazo robótico.

Se inspira en base a la notación de operaciones en GCODE, un lenguaje de programación para impresoras 3D, un protocolo de comunicación basado en texto plano, muy simple y eficiente, por lo que se decidió usar una notación similar para comunicarse entre librería y firmware.

4.4.2. Elementos Principales

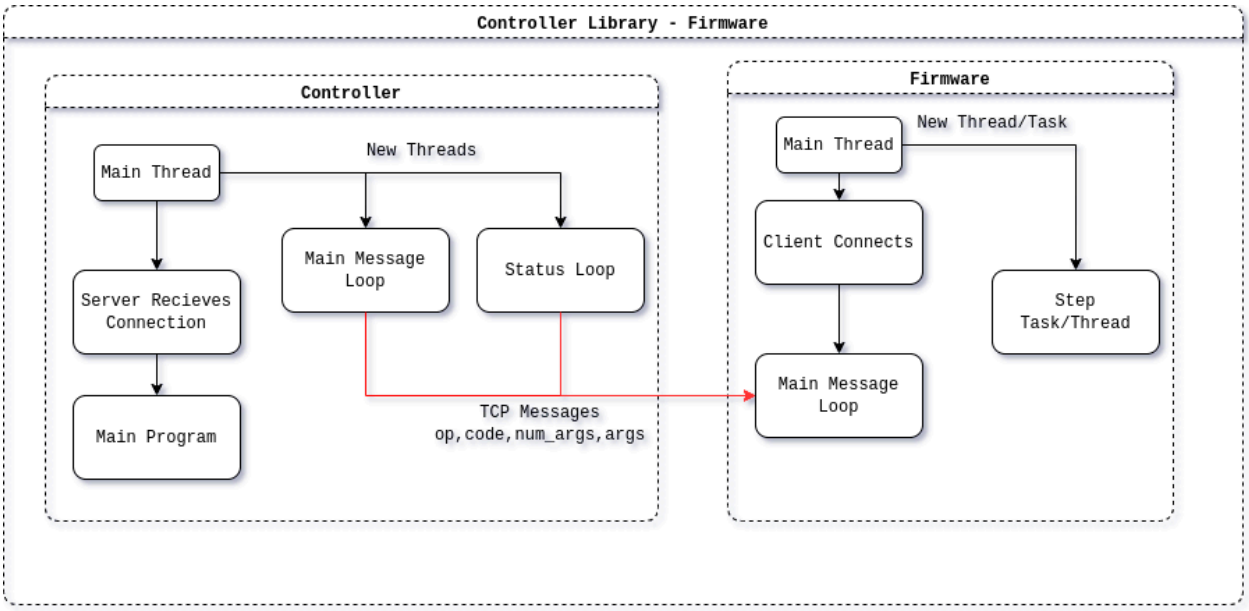

Figura 10:

Diagrama arquitectura de la librería y el firmware.

Los elementos principales de la librería y el firmware en el sistema de control del brazo robótico están diseñados para trabajar de manera eficiente y coordinada, garantizando una comunicación efectiva y un control preciso del brazo. A continuación, se detallan los componentes clave y el flujo de comunicación entre ellos:

Elementos Principales de la Librería:

1. Thread Principal (Main): Este thread se encarga de recibir mensajes del backend y ejecutar las operaciones correspondientes. Es donde se realizan las llamadas a los métodos de la librería, y se envían los mensajes al firmware.
2. Thread de Mensajes (Main Message Loop): Se encarga de la comunicación con el firmware, tanto para enviar como para recibir mensajes. Este thread es crucial para iniciar y mantener el flujo de datos entre la librería y el firmware.

3. Thread de Estado (Status Loop): Este thread se dedica a enviar constantemente mensajes de estado al firmware. La actualización continua del estado permite a la librería tener un conocimiento en tiempo real del estado del brazo robótico. La comunicación "asincrónica" asegura que el thread principal no se bloquee mientras espera una respuesta del firmware.

Elementos Principales del Firmware:

1. Thread Principal (Main): Similar al de la librería, este thread en el firmware se encarga de recibir mensajes y ejecutar operaciones. En caso de recibir un mensaje de movimiento, transfiere la responsabilidad al thread de paso.
2. Thread Step: Este thread se encarga de mover las articulaciones del brazo robótico a los ángulos deseados. Este thread se ejecuta en paralelo con el thread principal, lo que permite que el brazo se mueva mientras el firmware sigue recibiendo mensajes.

Ejemplo: Movimiento de una Articulación

Ahora veamos un ejemplo para entender mejor el flujo de comunicación y manejo de mensajes entre la librería y el firmware. En este caso, se considera un movimiento de la articulación 1 a un ángulo de x grados.

1. **Llamada del Método en el Controlador**:
   - Se invoca el método `move_joint_to(joint_idx, angle)` desde el controlador.
   - Este método genera internamente un mensaje M9 con los argumentos `joint_idx` (índice de la articulación) y `angle` (ángulo deseado).
   - El mensaje M9 se envía al firmware para su procesamiento.
2. **Recepción y Almacenamiento de Mensajes en el Firmware**:
   - Al recibir un mensaje, el firmware lo parsea dentro de su loop principal.
   - El mensaje se almacena en una cola específica para su tipo de operación (M, C, o S). Esta separación permite la ejecución asincrónica de mensajes y evita el bloqueo del loop principal.
3. **Revisión y Gestión de Estados de Mensajes**:
   - Dentro del loop, el firmware revisa la cola de mensajes para cada tipo de operación.
   - Cada mensaje puede encontrarse en uno de tres estados: No llamado, Llamado, o Terminado.
4. **Procesamiento del Mensaje M9**:

- En una iteración, el mensaje M9 se parsea y se prepara para su ejecución.
- En la siguiente iteración, se llama al handler de mensajes de tipo “M”, específicamente al caso del código 9, donde se encuentra la lógica correspondiente.

5. **Primera Llamada del Handler y Actualización de Estado**:

   - En la primera llamada del handler para el mensaje M9, se actualiza el ángulo objetivo de la articulación deseada.
   - El mensaje se marca como «llamado», indicando que la acción requerida se ha iniciado.

6. **Revisión y Finalización del Mensaje**:

   - En llamadas subsiguientes, el handler verifica si la articulación ha alcanzado su ángulo objetivo.
   - Una vez que la articulación llega a su objetivo, el mensaje se marca como «terminado».

7. **Conclusión del Proceso de Movimiento**:

   - El thread Step se encarga de llevar la articulación al ángulo objetivo.
   - Una vez alcanzado el objetivo y el mensaje marcado como terminado, se elimina de la cola y se procede al siguiente mensaje.

Beneficios del Manejo Asincrónico de Mensajes: Esta estrategia permite que las operaciones de movimiento no bloqueen otras acciones, facilita la respuesta a situaciones de emergencia, y permite la existencia de un thread dedicado para movimientos, manteniendo la fluidez y precisión.

En resumen, el diseño de la librería y el firmware, junto con su método de comunicación y manejo de mensajes, proporciona una base sólida y flexible para el control eficaz del brazo robótico. La arquitectura permite un control detallado de las operaciones del brazo, mientras se mantiene la capacidad de responder rápidamente a los cambios y las necesidades en tiempo real.

Codigo de ejemplo

El ejemplo de código presentado ilustra claramente cómo se puede utilizar la librería de control para manejar un brazo robótico. Este ejemplo demuestra la facilidad de uso y la accesibilidad de la librería, algo beneficioso para los usuarios que transitan desde la interfaz visual hacia la programación más directa del brazo robótico.

```python
from ribot.control.arm_kinematics import ArmParameters, ArmPose
from ribot.controller import ArmController, Settings

if __name__ == "__main__":
    # Arm parameters (Physical dimensions of the arm)
    arm_params: ArmParameters = ArmParameters()
    arm_params.a2x = 0
    arm_params.a2z = 172.48
    arm_params.a3z = 173.5
    arm_params.a4z = 0
    arm_params.a4x = 126.2
    arm_params.a5x = 64.1
    arm_params.a6x = 169

    controller = ArmController(arm_parameters=arm_params)
    controller.start(websocket_server=False, wait=True)

    controller.set_setting_joints(Settings.STEPS_PER_REV_MOTOR_AXIS, 400)
    controller.home()

    # Move to a position
    position = ArmPose(x=320, y=0, z=250, pitch=0, roll=0, yaw=0)
    controller.move_to(position)
    controller.wait_done_moving()
```

```
# Move to angles (rads)
angles = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
controller.move_joints_to(angles)


# Move the tool (rads)
controller.set_tool_value(1.2)


print("Homed:", controller.is_homed)
```

Explicación del Código

Instalación: Se instala fácilmente utilizando pip, el gestor de paquetes de Python, lo que facilita su uso y acceso.

```
pip install ribot-controller
```

Flujo del codigo:

1. Se inicia definiendo los parámetros físicos del brazo robótico (ArmParameters), especificando las dimensiones de cada segmento del brazo.

Inicialización del Controlador:

2. Se crea una instancia de ArmController, pasando los parámetros del brazo.

Se inicia el controlador con start(), con la opción de activar o no un servidor websocket para la simulación en el frontend.

Configuración y Homing:

3. Se establecen configuraciones específicas del brazo, como los pasos por revolución del eje del motor.

Se ejecuta el comando home() para llevar el brazo a su posición inicial o «home».

Movimiento a una Posición Específica:

5. Se define una posición (ArmPose) con coordenadas x, y, z y ángulos de orientación.

Se utiliza move_to() para mover el brazo a esa posición y wait_done_moving() para esperar hasta que el movimiento se complete.

Movimiento de Articulaciones a Ángulos Específicos:

6. Se pueden especificar los ángulos de cada articulación y moverlas utilizando move_joints_to().
7. Se ajusta el valor de la herramienta (por ejemplo, la apertura de una pinza) con set_tool_value().
8. Finalmente, se imprime el estado de homing del brazo con controller.is_homed.

El código muestra cómo un usuario con conocimientos básicos de programación en Python puede controlar el brazo robótico de manera eficiente. La disponibilidad de la librería en PyPI y su fácil instalación son aspectos clave que fomentan la experimentación y el aprendizaje en el campo de la robótica. Este enfoque da la posibilidad a los usuarios de la interfaz visual a dar sus primeros pasos en la programación, proporcionándoles una herramienta potente y accesible para controlar el brazo robótico.

4.4.3. Aspectos Relevantes de la Implementación del Firmware:

En esta sección, se exploran detalles clave de la implementación del firmware para el control del brazo robótico, destacando los aspectos que requirieron un esfuerzo y tiempo de desarrollo significativos.

MovementDriver:

Este componente define una interfaz para el movimiento de las articulaciones y permite la creación de subclases específicas según el hardware.

Se implementó soporte para motores paso a paso y servomotores, controlando los motores directamente a través de señales PWM y GPIO.

La implementación desde cero del control de motores busca minimizar las dependencias y las secciones de código dependientes del hardware específico.

Endstop:

Al igual que el `MovementDriver`, `Endstop` define una interfaz para los sensores de fin de carrera y permite la creación de subclases según el hardware.

Flexibilidad y Portabilidad:

Gracias al control detallado del hardware, se puede utilizar el mismo firmware y librería para diferentes tipos de hardware, como motores paso a paso y servomotores.

Aunque no se implementó en otros microcontroladores, el código puede ejecutarse tanto en una ESP32 como en un contenedor de Docker basado en Linux, gracias a la arquitectura modular del firmware y la librería.

Elección de la ESP32:

La comunicación a través de sockets TCP posibilita ejecutar el firmware en un contenedor Docker y comunicarse con la librería de control de manera transparente.

Secciones Dependientes del Hardware:

Se intento minimizar lo más posible las funciones que dependen directamente del hardware, ejemplos de estas son las funciones `hardware_step` y `hardware_setup` en `MovementDriver`, y `hardware_read` y `hardware_setup` en `Endstop`. Las cuales realizan los movimientos físicos del brazo y configuran sus componentes. Como es el uso de las entradas y salidas digitales del microcontrolador.

Por otro lado, es necesario el uso de threads para manejar los motores, siendo un requisito para la implementación del firmware. En ESP-IDF se usan tareas de FreeRTOS y en Linux se emplean pthreads.

4.4.4. Desafíos del Firmware

Manejo de Recursos

El diseño tuvo que tener en cuenta las limitaciones del hardware, especialmente en la ESP32, que tiene dos núcleos. Se requirió un balance entre la respuesta rápida del firmware y el control preciso y fluido de los motores.

Aquí se tuvo que tener en cuenta que el thread/task que se encarga de ejecutar los movimientos físicos de los motores no consumiera completamente un núcleo de la ESP32, esto resulta difícil ya que para lograr la fluidez en motores paso a paso a altas velocidades

se requiere de un control preciso de los tiempos de los pulsos de los motores, lo cual requiere de un control preciso de los tiempos de ejecución del thread/task que se encarga de esto.

Para mitigar esto se tuvo que tener en cuenta que cada cierto intervalo de tiempo se debía ceder el control del núcleo a otros threads/tasks, esto se logró generando pausas y llamando la función "taskYIELD" para liberar el núcleo.

Al mismo tiempo al generar este intervalo de tiempo donde no se estaban ejecutando las actualizaciones para los motores, se tiene que tomar en cuenta que en la siguiente llamada realizada para ejecutar los movimientos se deben de tomar pasos extra para mantener la velocidad esperada.

Control de Motores

En el desarrollo del firmware para el brazo robótico, se tomó la decisión de implementar desde cero el control de los motores, tanto servo como paso a paso. Esta decisión implicó un desafío significativo, ya que se requirió construir la lógica de control de estos motores utilizando únicamente señales PWM y GPIO, sin depender de bibliotecas o módulos pre-existentes.

La implementación del control directo de motores en el firmware del brazo robótico es una parte clave del proyecto que involucró desafíos significativos y ofreció varias ventajas. Este proceso implicó el manejo directo de las señales PWM (Modulación por Ancho de Pulso) para controlar la velocidad y la posición de los motores. El uso de GPIO (General Purpose Input/Output) fue esencial para manejar las señales digitales necesarias en el control de los motores. La programación desde cero del control de motores requirió un conocimiento profundo de cómo funcionan estos dispositivos a nivel de hardware, incluyendo una comprensión detallada de la temporización de las señales PWM y cómo estas interactúan con los motores para producir movimientos precisos y controlados.

Al controlar directamente los motores, se obtuvo una mayor flexibilidad y control sobre el comportamiento del brazo robótico. Esta aproximación permitió una adaptación más fina a las necesidades específicas del proyecto y una optimización más detallada del rendimiento del brazo. Esta metodología es superior porque permite realizar operaciones complejas de comunicación y movimiento de forma paralela, lo que facilita el control y la comunicación entre el firmware y la librería de control. Al ejecutar estos procesos de manera simultánea, se mejora la eficiencia y se reduce la latencia, lo que resulta en una respuesta más rápida y precisa del brazo robótico a las instrucciones dadas. Esto es particularmente beneficioso en aplicaciones que requieren alta precisión y coordinación en tiempo real.

4.4.5. Detalles de la implementación de la librería

La implementación de la librería de control es responsable de la ejecución de la cinemática y del control preciso del hardware. A continuación, se detallan los elementos clave de esta implementación:

Cinemática Inversa y Directa

Basándose en la investigación y cálculos aplicados, se desarrolló la lógica para manejar tanto la cinemática directa como la inversa en un brazo robótico de 6 grados de libertad.

La librería recibe la configuración necesaria para calcular la cinemática, como las dimensiones físicas del brazo y los vectores mencionados en la sección de cinemática. A partir de esto para cualquier movimiento se generan las instrucciones básicas para el funcionamiento del brazo, si se quiere ir a una posición cartesiana (x,y,z,roll,pitch,yaw), esta convierte las coordenadas en ángulos utilizando la cinemática inversa para que el firmware únicamente deba mover las articulaciones a los ángulos esperados.

Configuraciones del Firmware:

La librería permite configurar múltiples aspectos del firmware, incluyendo tipos de motores, pines para cada motor, sensores de fin de carrera, y configuraciones de movimiento como velocidad y dirección.

Las configuraciones se especifican en un archivo de configuración “TOML”, facilitando la librería adaptar a diferentes brazos robóticos.

Este enfoque modular y configurable asegura que la librería sea fácilmente transferible a otro brazo robótico, simplemente cambiando el archivo de configuración.

Ejemplo de Archivo de Configuración

En este archivo se detallan las configuraciones para cada articulación, incluyendo el motor, los pines, el sensor de fin de carrera y sus configuraciones, así como los parámetros generales del brazo y la herramienta.

```
title = "Robot arm configuration"

[joint_configuration]

    speed_rad_per_s = 0.35
```

```
    steps_per_rev_motor_axis = 800 # Microstepping
[arm_parameters]
# J1
    a1x= 0
    a1y= 0
    a1z= 0
#... rest of parameters
[tool]
    name = "tool"
    max_angle_rad = 0.0
    min_angle_rad = 0.0
    [tool.driver]
        type = "servo"
        pin = 27
[[joints]]
    name = "joint_1"
    homing_direction = -1 # 1 or -1
    conversion_rate_axis_joint = 9.7 # Gear ratio
    homing_offset_rad = 0 # Offset from home position
    speed_rad_per_s = 0.4
    [joints.driver]
        type = "stepper"
        dir_pin = 14
        step_pin = 13
    [joints.endstop]
```

```
        type = "hall"

        pin = 26

# ... rest of joints
```

La flexibilidad y modularidad de la librería de control son aspectos fundamentales para su eficacia y adaptabilidad. Al permitir una configuración detallada a través de un archivo "TOML", se facilita la personalización del control del brazo robótico para diferentes hardware y requisitos específicos.

4.5. Interfaz

La interfaz para el control del brazo robótico se diseñó con un enfoque en la usabilidad y la accesibilidad, utilizando tecnologías modernas y principios de diseño intuitivos. Esta interfaz consta de dos partes principales: una aplicación web y una visualización Unity WebGL integrada.

Elección de React:

Se seleccionó React para construir la interfaz web debido a su popularidad y facilidad de uso. React permite la creación de componentes reutilizables y facilita el desarrollo de interfaces dinámicas. La biblioteca `react-dnd` (drag and drop) se utilizó para implementar una interfaz interactiva que es esencial para la creación de rutinas de control del brazo robótico.

Accesibilidad para Usuarios Finales:

La interfaz basada en un framework web ofrece una gran accesibilidad. Un usuario final solo necesita acceder a un enlace desde su navegador para utilizar la interfaz y controlar el sistema completo, lo que elimina la necesidad de configuraciones o instalaciones complicadas.

4.5.1. Proceso de Diseño de la Interfaz

Enfoque en la Simplicidad y Accesibilidad:

La interfaz se diseñó para ser lo más accesible y sencilla posible. Se enfocó en elementos visuales claros, como colores e iconos, limitando la cantidad de texto, botones y menús.

La interfaz consta de tres secciones claras: visualización del brazo robótico, lista de bloques de programación y un panel de control.

Para el diseño de la interfaz del sistema de control del brazo robótico, se aplicaron los diez principios heurísticos de usabilidad de Nielsen [13], enfocándose en optimizar la experiencia del usuario y la eficiencia de la interfaz. Estos principios guiaron el desarrollo de una interfaz intuitiva y accesible:

1. Visibilidad del Estado del Sistema: Se aseguró que los usuarios siempre estuvieran informados de lo sucedido mediante retroalimentación adecuada en un tiempo razonable.

2. Coincidencia entre el Sistema y el Mundo Real: Se utilizó un lenguaje claro y conciso, con íconos y símbolos familiares para representar acciones, asegurando que la información apareciera en un orden lógico y natural.
3. Control y Libertad del Usuario: Se proporcionaron múltiples caminos para realizar una misma acción, permitiendo a los usuarios elegir la forma que les resulte más cómoda o eficiente. Como es la forma de acceder a los menús de contexto, las múltiples formas de controlar el movimiento físico del brazo o simplemente manipular la lista de acciones.
4. Consistencia y Estándares: Se siguió un diseño estándar, utilizando técnicas y convenciones conocidas para garantizar que los usuarios no se confundieran con términos, acciones o comportamientos inesperados.
5. Prevención de Errores: Se priorizó el diseño minimalista, manteniendo la información relevante a la vista y evitando la sobrecarga de datos innecesarios que pudieran llevar a errores.
6. Reconocimiento en lugar de Recuerdo: Se mantuvo toda la información necesaria a la vista, o accesible con no más de un clic, para minimizar la carga cognitiva del usuario.
7. Flexibilidad y Eficiencia de Uso: Se tienen múltiples formas de realizar una misma acción, como se mencionó, permite que el usuario elija la forma que le resulte más cómoda o eficiente.
8. Estética y Diseño Minimalista: Se adoptó un enfoque minimalista en el diseño, presentando solo la información necesaria para realizar las tareas, evitando elementos irrelevantes o distractores.
9. Ayudar a los Usuarios a Reconocer, Diagnosticar y Recuperarse de Errores: Se implementaron mensajes de error claros y específicos que indicaran el problema, como puede ser el movimiento del brazo fuera de su área de alcance.
10. Ayuda y Documentación: Aunque el objetivo fue hacer que el sistema fuera intuitivo, se proporcionó ayuda adicional en forma de tooltips, iconos de ayuda para campos específicos y un tutorial integrado en la aplicación para guiar a los usuarios a través de su funcionamiento.

La aplicación de estos principios heurísticos de Nielsen aseguró que la interfaz fuera no solo funcional y fácil de usar, sino también agradable estéticamente, mejorando la experiencia general del usuario y facilitando el aprendizaje y la interacción con el brazo robótico.

4.5.2. Mockup de Baja Fidelidad de la Interfaz

Se inició el diseño con la creación de un mockup de baja fidelidad. Para esto, se investigaron interfaces visuales similares o relacionadas en el ámbito de la robótica, buscando inspiración y mejores prácticas.

Se prestó atención a las interfaces de programación visual, tomando como referencia plataformas como Scratch, que es una interfaz de programación visual para niños.

El mockup de baja fidelidad refleja estos principios y proporciona una visión clara del diseño y la disposición de la interfaz:

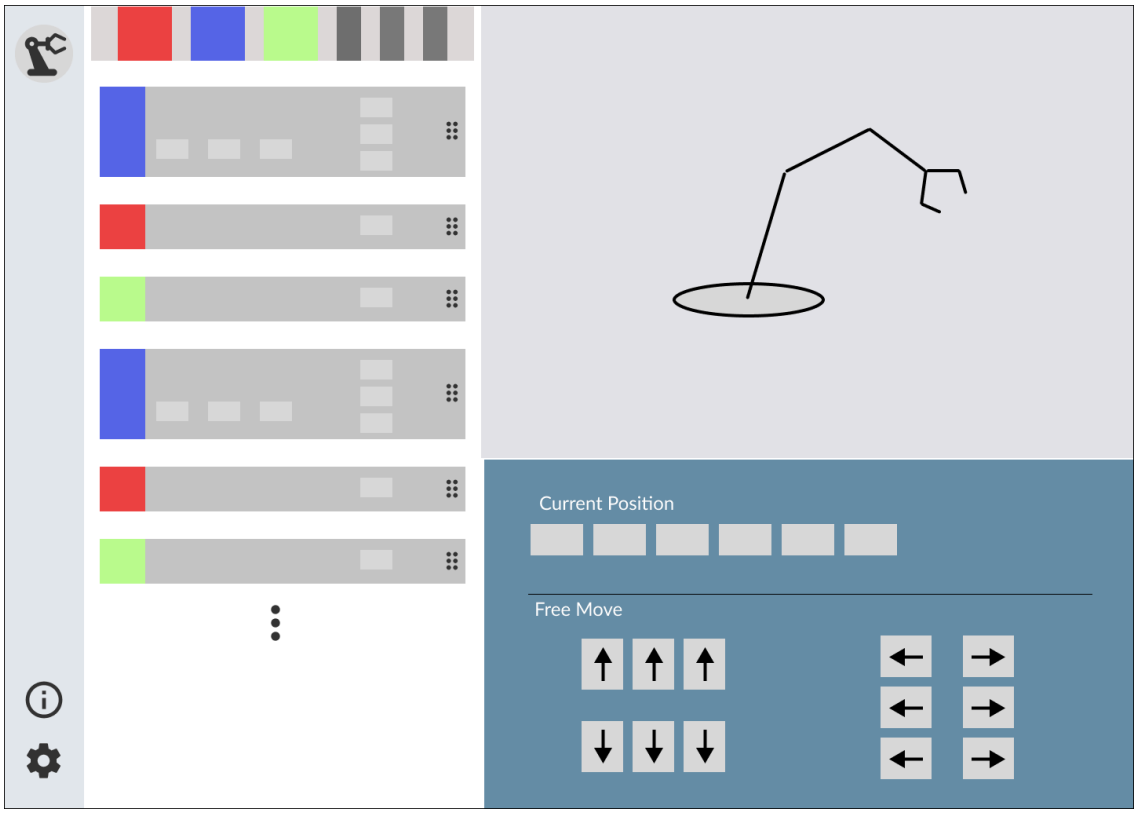

Figura 11:

Mockup de baja fidelidad de la interfaz.

En este mockup se aprecian las tres secciones principales de la interfaz, diseñadas para facilitar la interacción del usuario con el sistema y proporcionar una experiencia de usuario intuitiva y agradable. La integración de la visualización Unity WebGL en la interfaz web es un aspecto innovador que mejora la interactividad y el entendimiento del comportamiento del brazo robótico en tiempo real.

El proceso de desarrollo de la interfaz para el sistema de control del brazo robótico avanzó hacia la creación de un mockup de alta fidelidad, centrándose en los principios de diseño de interfaces generales como la consistencia, la simplicidad y la accesibilidad. Este enfoque garantiza que la interfaz no solo sea funcional, sino también atractiva y fácil de usar para los usuarios finales.

La fase de diseño de la interfaz del sistema de control del brazo robótico incluyó una búsqueda de inspiración adicional más allá de la funcionalidad, centrándose en aspectos visuales y estéticos. Esta investigación se hizo para integrar y respetar los principios de diseño de interfaces ampliamente reconocidos en la industria. Al hacerlo, se garantizó que la interfaz resultante fuera no solo intuitiva y fácil de usar, sino también atractiva desde el punto de vista estético y profesional en su presentación. Este enfoque en el diseño visual y la estética es fundamental para crear una experiencia de usuario que sea agradable y atractiva, así como funcional.

En el proceso de diseño, se pusieron en práctica varios principios clave de diseño de interfaces. En primer lugar, se enfatizó la consistencia en todo el diseño de la interfaz, lo que facilita a los usuarios la familiarización y el uso intuitivo de la plataforma. La simplicidad fue otro principio fundamental, asegurando que la interfaz fuera clara y directa, evitando la sobrecarga de información y elementos visuales que podrían confundir o abrumar a los usuarios. Por último, la accesibilidad se mantuvo como una prioridad en todo momento, asegurando que la interfaz fuera utilizable y comprensible para un amplio espectro de usuarios, incluyendo aquellos sin experiencia previa en programación o robótica. Este enfoque en la accesibilidad es crucial para garantizar que el sistema sea inclusivo y pueda ser utilizado por varios usuarios.

4.5.3. Mockup de Alta Fidelidad

Como resultado de este proceso, se creó un mockup de alta fidelidad que representa el diseño final y detallado de la interfaz:

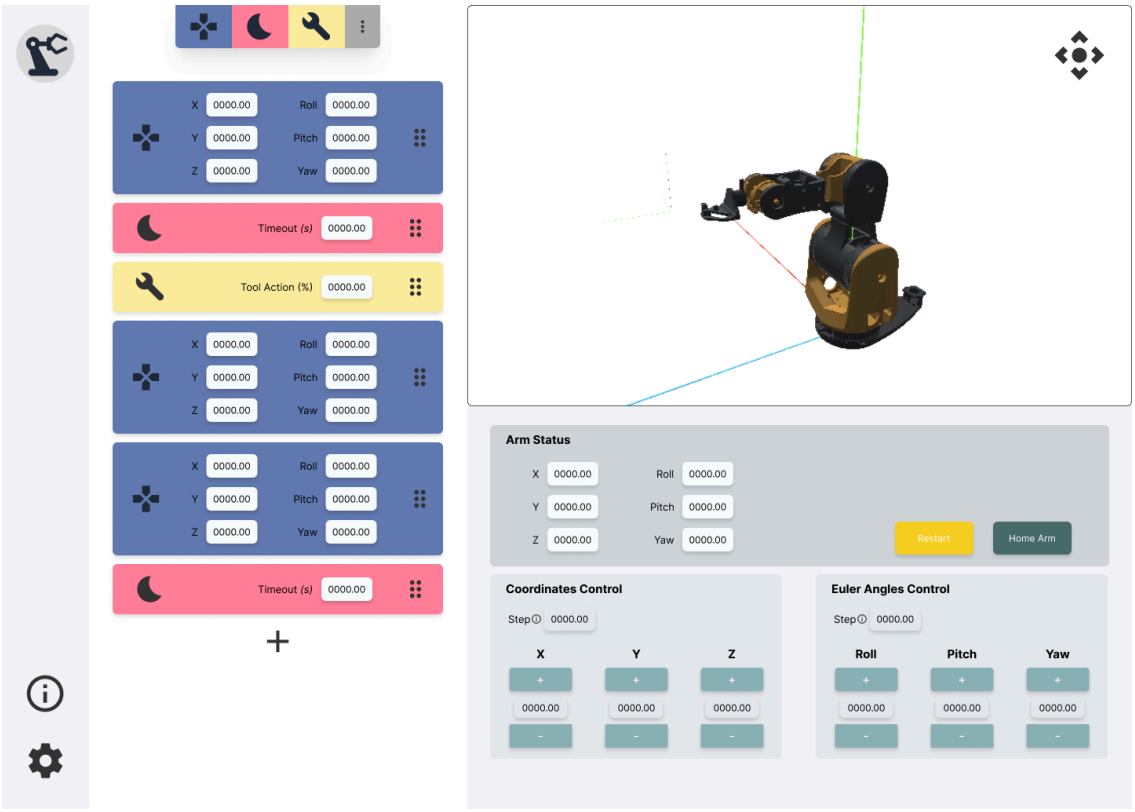

Figura 12:

Mockup de alta fidelidad de la interfaz.

Este mockup de alta fidelidad muestra cómo se integraron los principios de diseño de interfaces en la presentación visual y la disposición de los elementos de la interfaz. Se observa cómo cada componente ha sido cuidadosamente diseñado para proporcionar una experiencia de usuario coherente, atractiva y funcional. Este nivel de detalle y atención en el diseño es fundamental para asegurar que la interfaz no solo cumpla con su función de controlar el brazo robótico de manera eficiente, sino que también ofrezca una experiencia de usuario positiva y enriquecedora.

La interfaz del sistema de control del brazo robótico, desarrollada con un enfoque en la usabilidad y la funcionalidad, se divide en tres partes principales: la lista de acciones, el panel de control y la visualización Unity WebGL del brazo. A continuación, se detallan estos componentes y el proceso de desarrollo de la interfaz.

Lista de Acciones

La lista de acciones es un componente clave de la interfaz, donde los usuarios encuentran bloques para crear rutinas de movimiento del brazo robótico. Estos bloques pueden ser arrastrados y soltados en el panel de rutina. Los tipos de bloques definidos son:

- **Move**: Permite mover el brazo a una posición específica en el espacio, utilizando coordenadas cartesianas y ángulos de orientación.
- **Sleep**: Hace que el brazo espere un tiempo específico, medido en segundos.
- **Tool**: Cambia el valor de la herramienta, como la apertura de una pinza en el caso de un servo.
- **Action Set**: Agrupa varias acciones en un solo bloque para simplificar la creación de rutinas complejas.

Los usuarios pueden guardar, ejecutar individualmente o dentro de una rutina, eliminar o duplicar estas acciones. Además, la rutina completa se puede guardar en un archivo JSON para su posterior carga y ejecución.

Panel de Control

Este panel ofrece controles para mover libremente el brazo robótico en el espacio cartesiano o directamente a través de sus articulaciones. Aquí también se encuentra el control de la herramienta del brazo.

Visualización Unity WebGL

La interfaz incluye una visualización en tiempo real del brazo robótico, realizada con Unity WebGL. Esta visualización se actualiza con los ángulos de las articulaciones y permite a los usuarios ver el estado actual del brazo. La comunicación entre la interfaz y Unity se realiza a través de websockets, empleando la librería socket.io para facilitar esta interacción.

Desarrollo y Pruebas de Usabilidad

La implementación de la interfaz se realizó utilizando React, aprovechando su capacidad para crear componentes reutilizables y dinámicos. El backend se desarrolló con FastAPI, dada su simplicidad y eficiencia en la creación de APIs REST, y se encarga de conectar la interfaz con la librería de control y el firmware.

A lo largo del desarrollo, se realizaron varias pruebas de usabilidad para refinar la interfaz. A pesar de los cambios menores y adiciones de funcionalidades, la estructura general de la interfaz se mantuvo constante, validando el diseño base y la disposición de los elementos.

Finalmente, se alcanzó la versión final de la interfaz, que se encuentra disponible en el repositorio del proyecto. Este desarrollo iterativo y enfocado en la retroalimentación de los usuarios ha resultado en una interfaz intuitiva, funcional y accesible para controlar el brazo robótico.

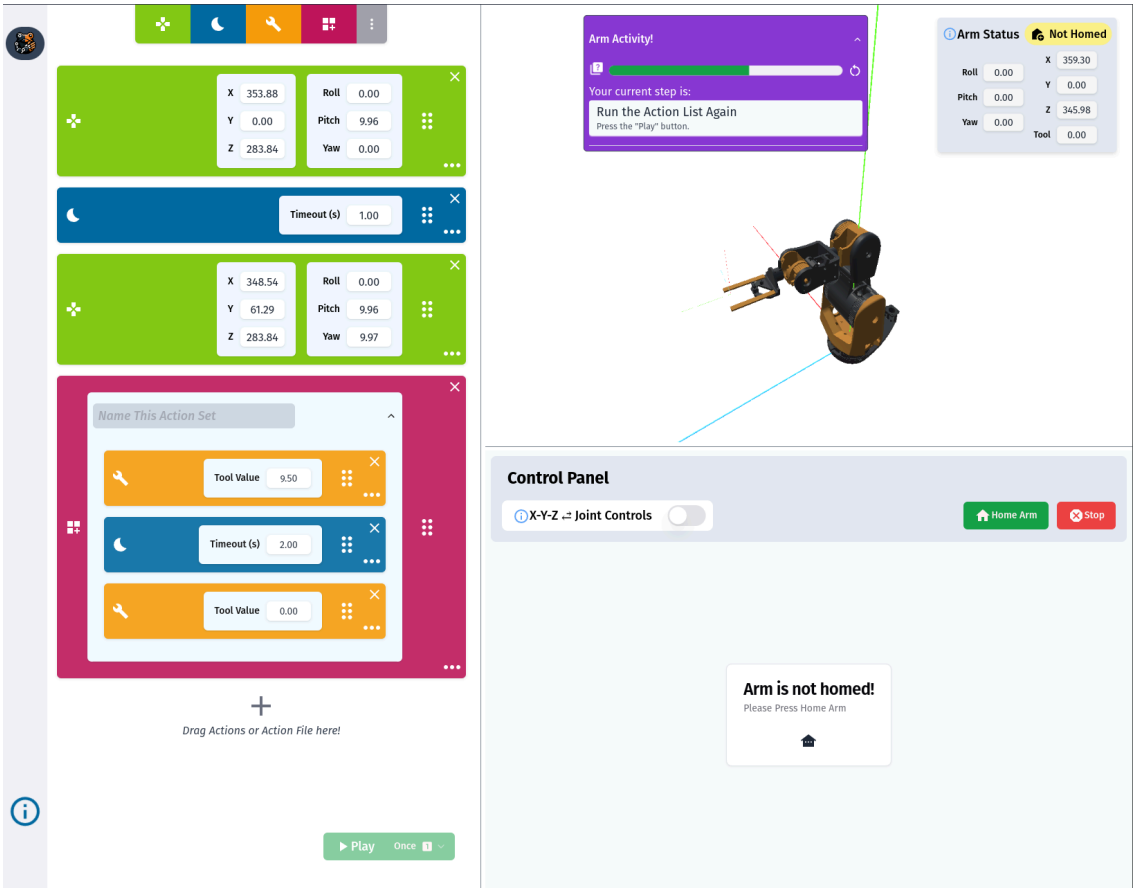

Figura 13:

Versión final e implementada de la interfaz.

4.6. Implementación en Hardware

La implementación del sistema de control en hardware existente representa un paso crucial para demostrar su viabilidad y eficacia práctica. A continuación, se ofrece una descripción resumida y directa de los pasos necesarios para llevar a cabo esta implementación:

4.6.1. Requisitos de Hardware y Compatibilidad

1. **Compatibilidad con el Modelo Matemático**:

   - Cualquier brazo robótico que se ajuste al modelo matemático presentado puede ser un candidato para implementar este sistema de control. Se debe considerar que el sistema está pensado para brazos robóticos de 6 grados de libertad.

2. **Requisitos de Hardware**:

   - **ESP32**: Es el microcontrolador principal utilizado en el sistema.
   - **Motores Paso a Paso**: Se requieren drivers compatibles con pines STEP y DIR, como A4988, DRV8825, TB6600 o DM542.
   - **Motores Servo**: Se necesita un driver tipo PWM, y se han utilizado modelos como mg995 o sg90.
   - **Sensores de Fin de Carrera**: Se deben utilizar sensores tipo hall con salida digital, como el a3144.

4.6.2. Proceso de Implementación

1. **Instalación de Dependencias**: Lo primero es instalar las dependencias necesarias: Docker Desktop y Python 3.8, junto con la librería esptool. Estas herramientas son fundamentales para la configuración y el manejo del firmware.
2. **Flashear el Firmware**: Para cargar el firmware en el microcontrolador ESP32, se utiliza el comando `./manage.py build-esp --flash`. Este paso simplifica el proceso al evitar la necesidad de instalar todo el entorno de desarrollo de ESP32, utilizando en su lugar una imagen de Docker y el protocolo RFC2217 para la comunicación.
3. **Conexiones y Configuración**: Se suministran diagramas detallados de las conexiones necesarias entre el ESP32, los motores y los sensores. La configuración de cada articulación, motor y sensor de fin de carrera se realiza en un archivo específico, donde se definen los pines y las configuraciones particulares.

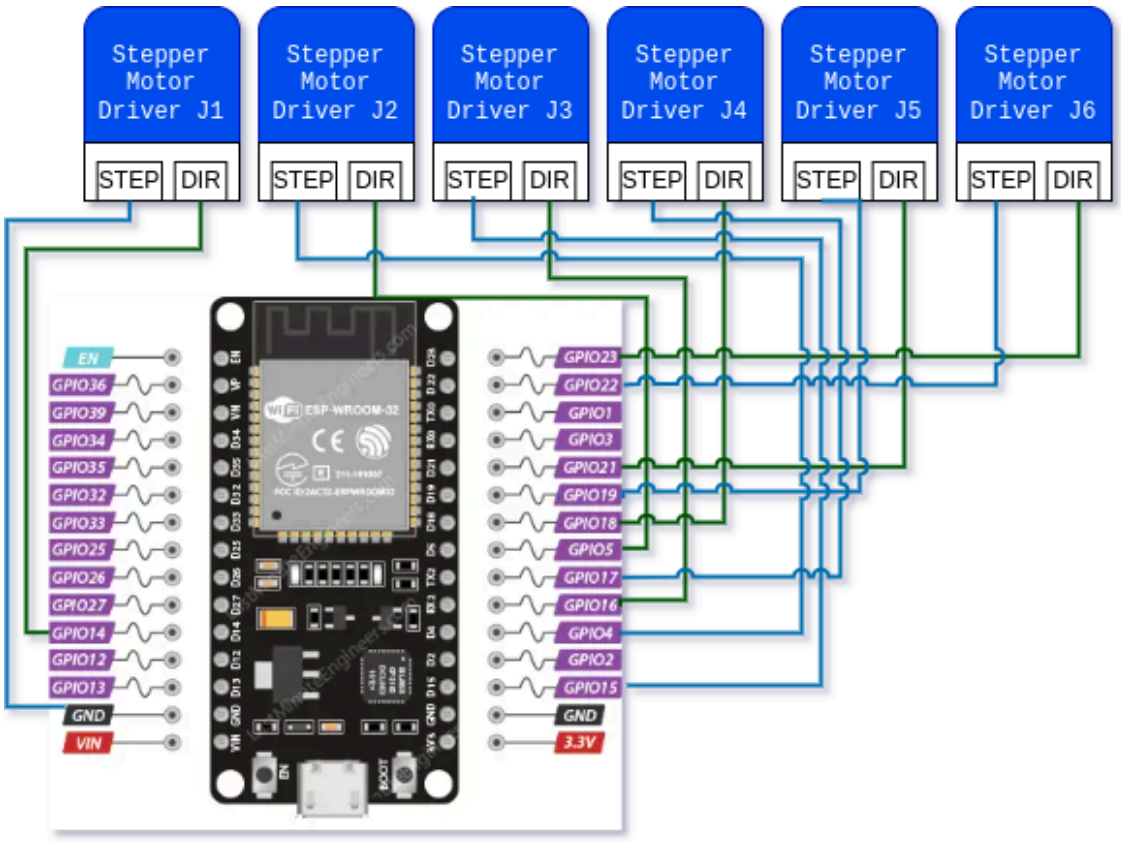

Figura 14:

Diagrama de conexiones: Ejemplo de configuración para un motor paso a paso con driver TB6600.

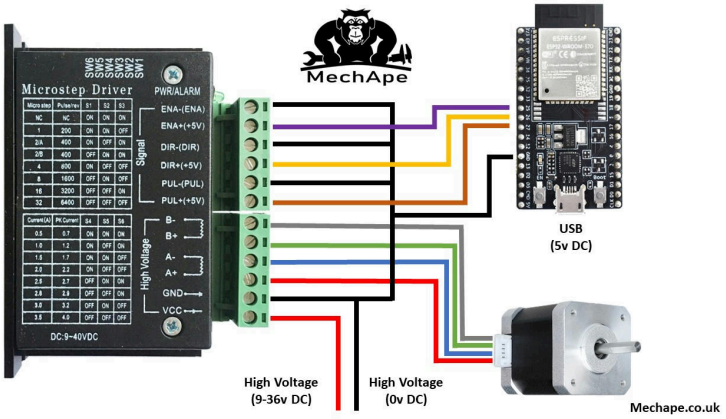

Figura 15:

Diagrama de conexiones: Ejemplo de configuración para un motor paso a paso con driver TB6600.

En estas figuras podemos ver un ejemplo de lo que sería el cableado para 6 motores paso a paso con drivers TB6600.

4. **Configuración del Archivo .env y Modelo Matemático**: Además de la configuración física, es necesario modificar el archivo .env del sistema para incluir detalles como la red WiFi y su contraseña. También se deben ingresar las dimensiones físicas del brazo robótico para los cálculos relacionados con el modelo matemático.
5. **Ejecución del Sistema de Control**: Una vez configurado y flasheado el firmware, el sistema de control se inicia con el comando `./manage.py runserver --esp`. Este comando activa las imágenes de Docker para ejecutar el backend de FastAPI con la librería de control, el frontend de React y la visualización de Unity.
6. **Opcional: Montaje en Raspberry Pi**: Dada la implementación basada en Docker, el sistema puede ser fácilmente montado en un Raspberry Pi. Esto facilita el acceso a la interfaz web desde cualquier dispositivo en la misma red, permitiendo el control del brazo robótico de manera remota y sin instalaciones adicionales.

En resumen, estos pasos delinean un proceso claro y estructurado para implementar el sistema de control en hardware existente. La adaptabilidad y flexibilidad del sistema permiten su uso en una variedad de contextos y configuraciones de hardware, lo que lo hace accesible y práctico para una amplia gama de usuarios.

5. Evaluación de la Solución y Resultados

Para validar la solución propuesta en el proyecto de desarrollo de una plataforma de control para un brazo robótico, se emplearon dos enfoques distintos: uno enfocado en el hardware y firmware, y otro en la interfaz de usuario.

5.1. Validación de Librería y Firmware

La validación de los componentes de la librería y firmware se realizó de manera práctica, utilizando el prototipo de brazo robótico creado por el estudiante. Pese a las limitaciones en la calidad del hardware y su potencial para mejoras, el prototipo bastaba para probar el funcionamiento básico y general del sistema. Los aspectos clave evaluados incluyen:

- **Control de Motores:** Se verificó la capacidad del sistema para controlar efectivamente los movimientos de los motores del brazo robótico.
- **Fluidez de Movimientos:** Se observó la suavidad y precisión en la ejecución de los movimientos del brazo robótico.

5.1.1. Proceso de Validación en Hardware Real:

1. **Implementación del Sistema de Control**: El brazo robótico diseñado por el estudiante fue equipado con el sistema de control desarrollado. Esta implementación incluyó la integración de la librería de control y el firmware adecuado para el microcontrolador ESP32.
2. **Pruebas Continuas Durante el Desarrollo**: A lo largo del desarrollo del proyecto, se realizaron pruebas constantes en el brazo robótico. Estas pruebas no solo verificaron la funcionalidad básica del sistema, sino que también permitieron evaluar y ajustar las diferentes componentes del firmware y la librería, asegurando su correcta interacción y rendimiento.
3. **Presentación de Imágenes y Video Demostrativo**: Para ilustrar la implementación y el funcionamiento del sistema, se presentan imágenes del brazo robótico utilizado en las pruebas. Además, se dispone de un video demostrativo que muestra el brazo en acción, utilizando el sistema de control implementado. Esta evidencia visual sirve para confirmar el éxito de la implementación y proporciona una muestra tangible de cómo el sistema opera en un entorno real.

> ***Video Demostrativo*** [14]

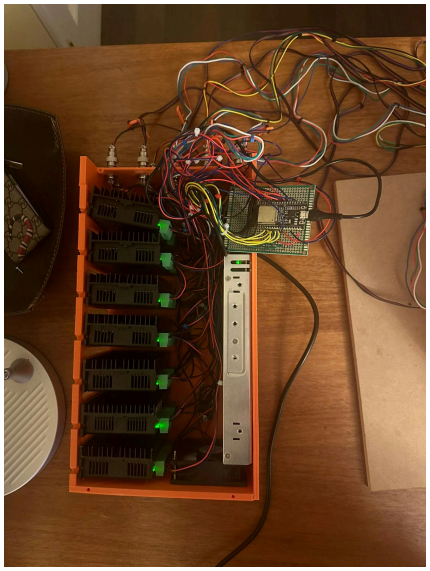

Figura 16:

Caja de Control: Se utilizó una caja de control para alojar el ESP32 y los drivers de los motores.

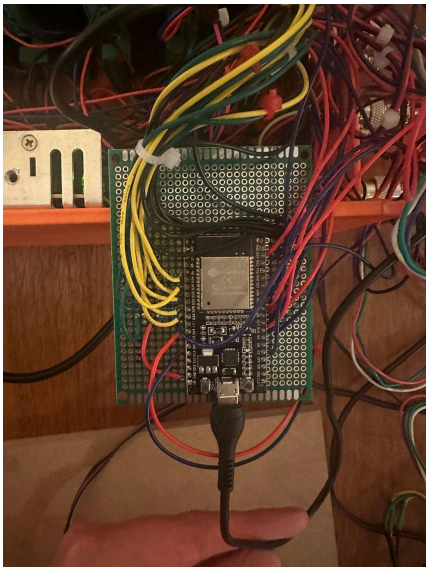

Figura 17:

Microcontrolador ESP32: Se utilizó un ESP32 para controlar los motores paso a paso y servomotores.

Figura 18:

Brazo Robótico: Prototipo utilizado para la validación del sistema de control.

La validación en hardware real es fundamental no solo para probar la efectividad del sistema sino también para identificar áreas de mejora y ajustes necesarios. Al implementar y probar el sistema en un brazo robótico físico, se entiende cómo los elementos teóricos y de software se traducen en el mundo real, indispensable para desarrollar soluciones robóticas prácticas y funcionales.

5.2. Validación de la Interfaz de Usuario

Para la interfaz de usuario, el objetivo era desarrollar una plataforma intuitiva y simple, accesible para usuarios sin experiencia previa en robótica o áreas afines.

La hipótesis planteada para esta validación fue:

> *«Si la interfaz es intuitiva y simple de utilizar, entonces los usuarios podrán crear rutinas de manera rápida y sencilla, sin importar su experiencia previa con el tema ni área de estudio.»*

5.3. Diseño de la Actividad de Prueba

La actividad de prueba para el sistema de control del brazo robótico se diseñó para una evaluación detallada y multifacética de la usabilidad del sistema. Esta prueba combinó enfoques cuantitativos y cualitativos para capturar una visión completa de la experiencia del usuario con la interfaz [15] . A continuación, se detalla la justificación de estas elecciones y los aspectos específicos que se buscaban medir.

1. **Medición del Tiempo de Tarea (Cuantitativo)**: La sección cuantitativa se centró en medir el tiempo que los usuarios tardan en completar una tarea específica. En este caso, la tarea asignada fue crear una rutina de movimiento para el brazo robótico. La medición da datos objetivos sobre la eficiencia de la interfaz en cuanto a facilidad de uso y del proceso de creación de rutinas.

1. **Encuesta Post-Actividad (Cualitativo)**: Tras completar la tarea, se realizó una encuesta a los usuarios para evaluar su experiencia con la interfaz. Esta incluyó preguntas sobre la experiencia general del usuario, su edad y su área de estudio o especialización. Este enfoque cualitativo busca comprender aspectos más subjetivos y personales de la experiencia del usuario, como la satisfacción, la facilidad percibida y los posibles desafíos enfrentados.

Objetivos de la Actividad de Prueba

1. **Medir la Experiencia del Usuario**: El objetivo principal de la actividad de prueba es medir de manera integral la experiencia del usuario con la interfaz. A partir de esto se busca entender no solo cuán rápidamente y eficientemente pueden los usuarios completar una tarea, sino también cómo se sienten al usar la interfaz, si encuentran desafíos particulares y cómo su trasfondo personal o profesional podría influir en su experiencia.

2. **Análisis de Resultados**: El análisis se centrará en la sección cuantitativa, especialmente en el tiempo de completación de tareas, para obtener datos concretos y objetivos. Sin embargo, los resultados cualitativos también son esenciales para proporcionar un

contexto más rico y comprensión de las experiencias individuales y las percepciones de los usuarios.

Mediante esta combinación de métodos cuantitativos y cualitativos, la actividad de prueba buscaba obtener una comprensión amplia y detallada de cómo los usuarios interactúan con la interfaz y qué aspectos podrían mejorarse. Este enfoque mixto es crucial para asegurar que la interfaz no

Estructura de la Actividad

1. **Introducción:** Se presentó a los usuarios una breve descripción sobre qué es un brazo robótico, el concepto de coordenadas y los ángulos de aproximación.
2. **Presentación de Instrucciones:** Un recuadro morado en la aplicación mostraba una barra de progreso y una serie de pasos o instrucciones a seguir.
3. **Actividades de Prueba:** Aproximadamente 20 pasos guiaron a los usuarios a través de una actividad de prueba, que incluía mover el brazo libremente, agregar bloques de acción para crear una rutina y luego ejecutarla. Durante esta actividad, se midió la velocidad de completación, es decir, el tiempo transcurrido entre cada paso y el siguiente. Esta medición del tiempo entre el paso anterior y el actual permitió evaluar la eficiencia y la facilidad de uso de la interfaz, proporcionando datos valiosos sobre la experiencia del usuario y la capacidad de respuesta del sistema. Esta información es crucial para entender cómo los usuarios interactúan con la plataforma y para identificar áreas de mejora en la usabilidad y el rendimiento del sistema.

Algunas tareas específicas incluyeron:

- Poner el brazo en posición inicial («Home el brazo»).
- Mover una articulación específica del brazo.
- Agregar un bloque de movimiento.
- Agregar y modificar un bloque de «sleep» en la rutina.
- Mover el brazo a una posición cartesiana específica.
- Interactuar con la lista de acciones para crear un bloque de conjunto de acciones y agregar bloques de movimiento dentro de este.

Recolección de Datos y Feedback

1. **Medición Automática del Tiempo de Realización de Tareas:**

Se capturó la duración que cada usuario tomó para completar las tareas, proporcionando datos objetivos sobre la usabilidad. Esta medición del tiempo fue realizada de forma automática, ya que la actividad fue programada dentro de la interfaz. Por lo tanto, cada paso podía ser validado por el estado actual del software y era recolectado de manera programática. Este enfoque permitió una recopilación de datos precisa y sin sesgos, asegurando que la información recabada reflejara con exactitud el desempeño y la experiencia del usuario en tiempo real. Además, facilitó un análisis más eficiente y detallado de la usabilidad del sistema, contribuyendo a una evaluación completa de su eficacia y accesibilidad.

2. **Encuesta Post-Actividad:**

Al finalizar la actividad, los participantes completaron una encuesta diseñada para recopilar información sobre su experiencia con la interfaz, además de datos demográficos y académicos. Los elementos clave de la encuesta incluyeron:

- **Resultados de la Actividad**: Este apartado se autocompletó con información detallada sobre los tiempos de realización y el grado de completación de la actividad, datos generados automáticamente por la interfaz web.
- **Edad**: Se solicitó a los usuarios seleccionar un rango de edad.

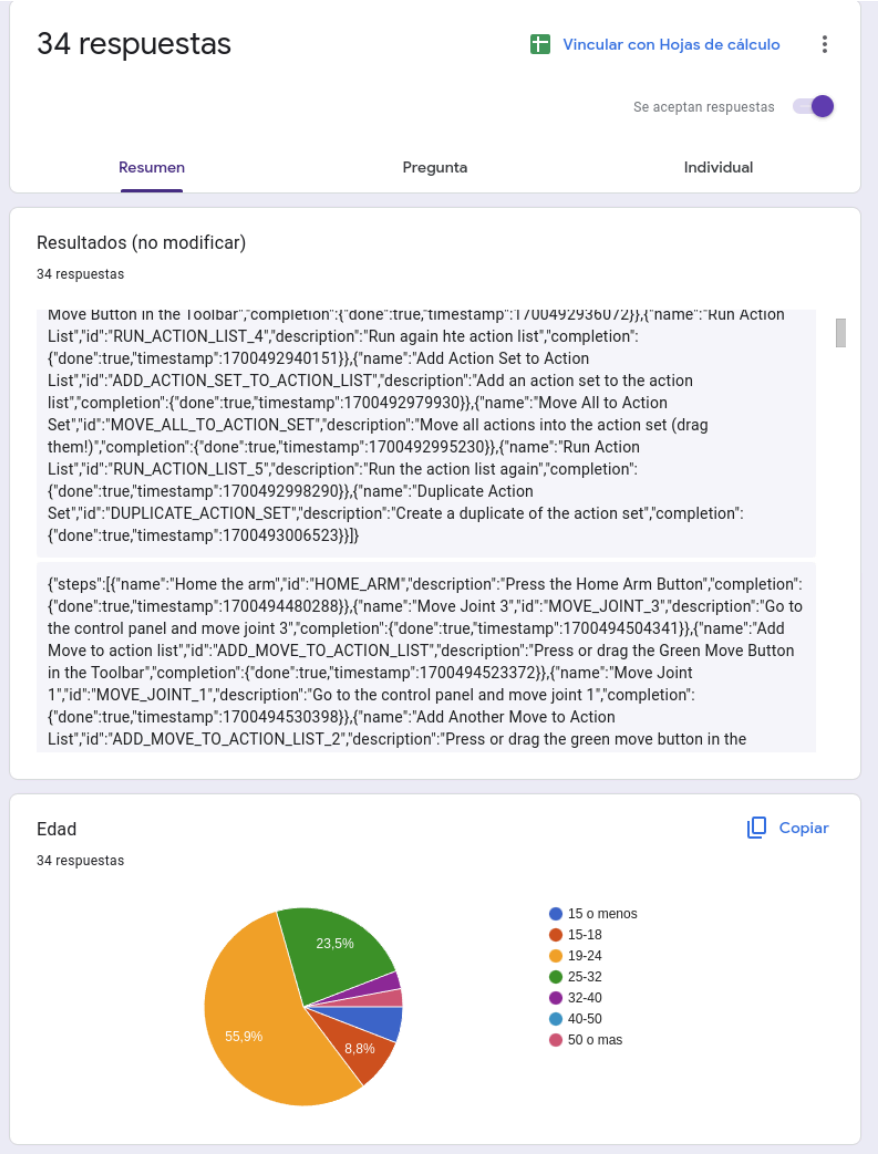

Figura 19: Resultados de la actividad de prueba (Respuesta autorellenada y Edad)

Posteriormente, la encuesta continuó con preguntas relacionadas con el perfil profesional y académico de los usuarios:

- **Ocupación Actual**: Opciones para indicar si el usuario es estudiante escolar, universitario, profesional, entre otros.
- **Área de Estudio**: Selección del campo de estudio, como computación, electrónica, derecho, etc.
- **Facilidad de la Actividad**: Evaluación de la facilidad para completar la actividad propuesta.

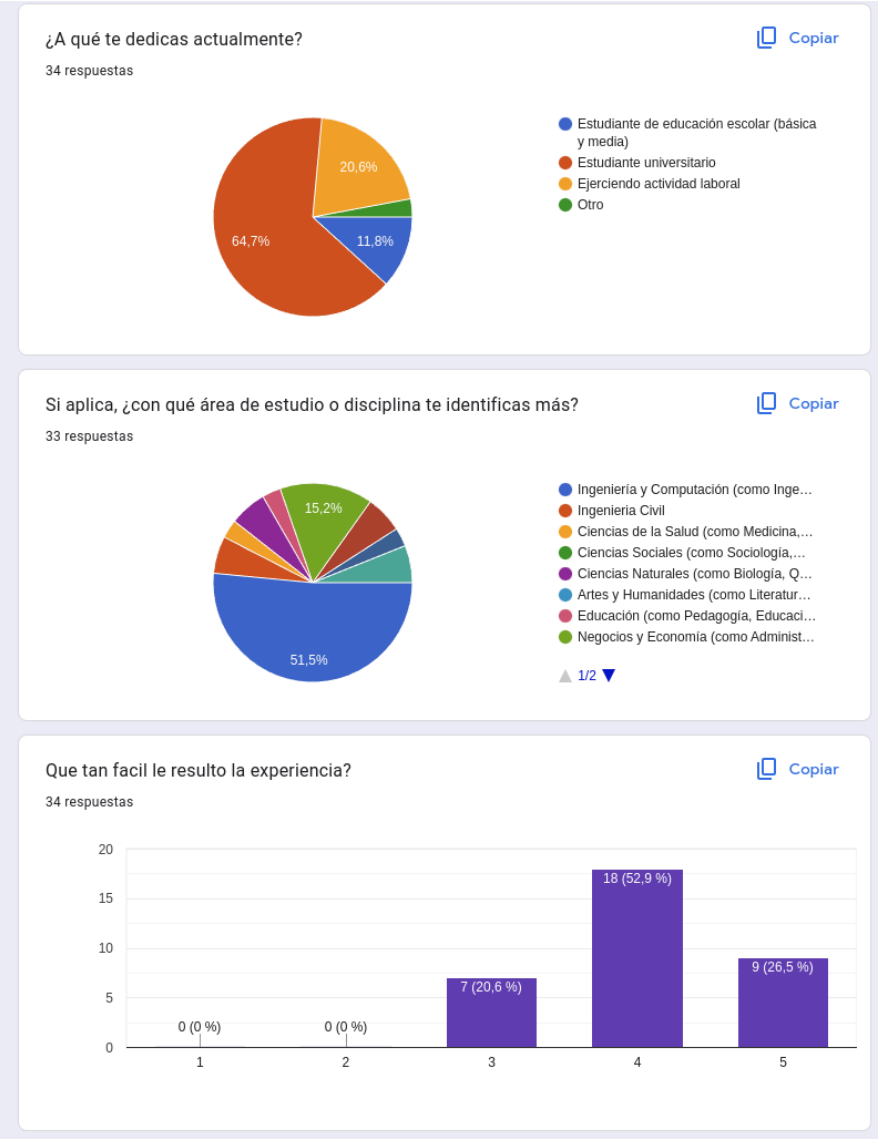

Figura 20: Resultados de la actividad de prueba (Perfil del usuario y facilidad de la actividad)

Esta estructura de la encuesta post-actividad permitió obtener una visión integral de la experiencia de los usuarios con la interfaz y su contexto profesional o académico.

5.3.1. Implementación Técnica de la Actividad

Despliegue en Vercel:

El frontend de la aplicación fue desplegado en Vercel, una plataforma de alojamiento para aplicaciones web frontend. La elección de Vercel como solución de hosting se basó en su facilidad de uso, rendimiento optimizado y capacidad para manejar tráfico web dinámico. La implementación en Vercel permitió un acceso universal a la actividad, haciendo posible que cualquier usuario con acceso a internet y un dispositivo adecuado, como un

laptop o una computadora de escritorio, pudiera participar sin la necesidad de instalaciones adicionales. Esta accesibilidad fue crucial para maximizar el número de participantes y obtener una gama más amplia de datos de usabilidad.

Servicios en Servidor Linode:

Para el backend y otros servicios, se utilizó un servidor Linode. Este servidor en la nube proporcionó la infraestructura necesaria para alojar varios componentes de la aplicación, incluyendo el firmware que se ejecutaba en un entorno Dockerizado, el backend de la aplicación y el servidor web necesario para la visualización de Unity WebGL.

Esta arquitectura híbrida, combinando el despliegue en Vercel para el frontend y Linode para los servicios del backend y otros componentes, resultó en una solución efectiva para la implementación de la actividad. Esta configuración no solo aseguró la disponibilidad y accesibilidad de la aplicación desde cualquier navegador web estándar, sino que también proporcionó la posibilidad de realizar la actividad con un mayor número de personas.

Además, la capacidad de medir automáticamente el tiempo de completación de las tareas dentro de la aplicación fue un componente crítico de la implementación. Permitió recopilar datos precisos y relevantes sobre la experiencia del usuario, facilitando así un análisis detallado y fiable de la usabilidad de la interfaz. En conjunto, la implementación técnica de la actividad fue un factor clave para evaluar y validar la eficacia de la interfaz del sistema de control del brazo robótico.

5.4. Resultados y Análisis de la Actividad de Prueba

Los resultados obtenidos de la actividad con aproximadamente 30 participantes revelaron valiosa información sobre la usabilidad de la interfaz del sistema de control del brazo robótico. La distribución de los participantes incluyó un 50% de estudiantes de ingeniería en campos como computación y electrónica, y el otro 50% compuesto por estudiantes de áreas diversas como negocios y derecho.

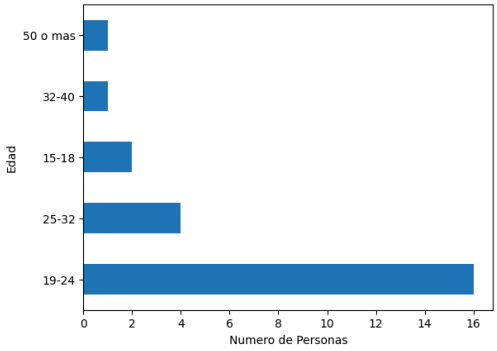

Figura 21:

Distribución de personas por edad.

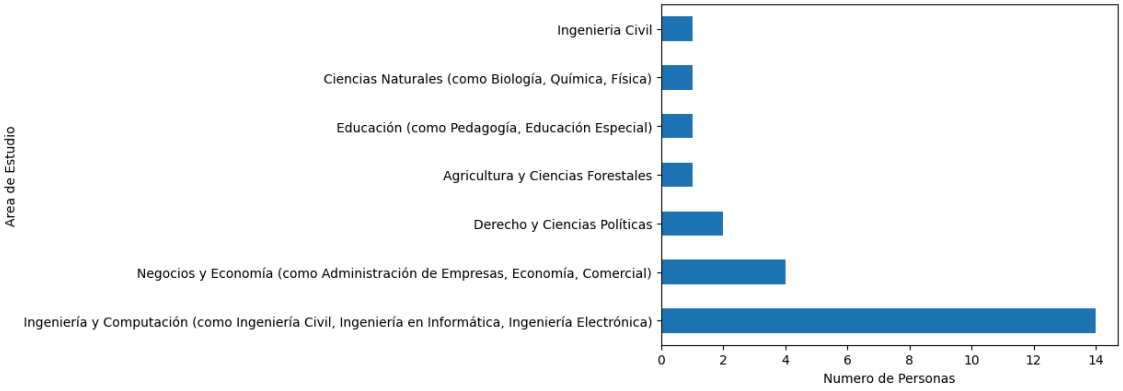

Figura 22:

Distribución de personas por área de estudio.

Para garantizar la precisión en el análisis de datos, se aplicó la técnica de z-score para eliminar outliers, estableciendo un umbral de 4. Este enfoque se eligió con el objetivo de preservar la mayor cantidad de datos viables, manteniendo la integridad y relevancia del análisis. Las figuras presentadas ilustran la distribución de los participantes por edad y área de estudio, proporcionando un panorama claro de la diversidad de los usuarios involucrados en la prueba.

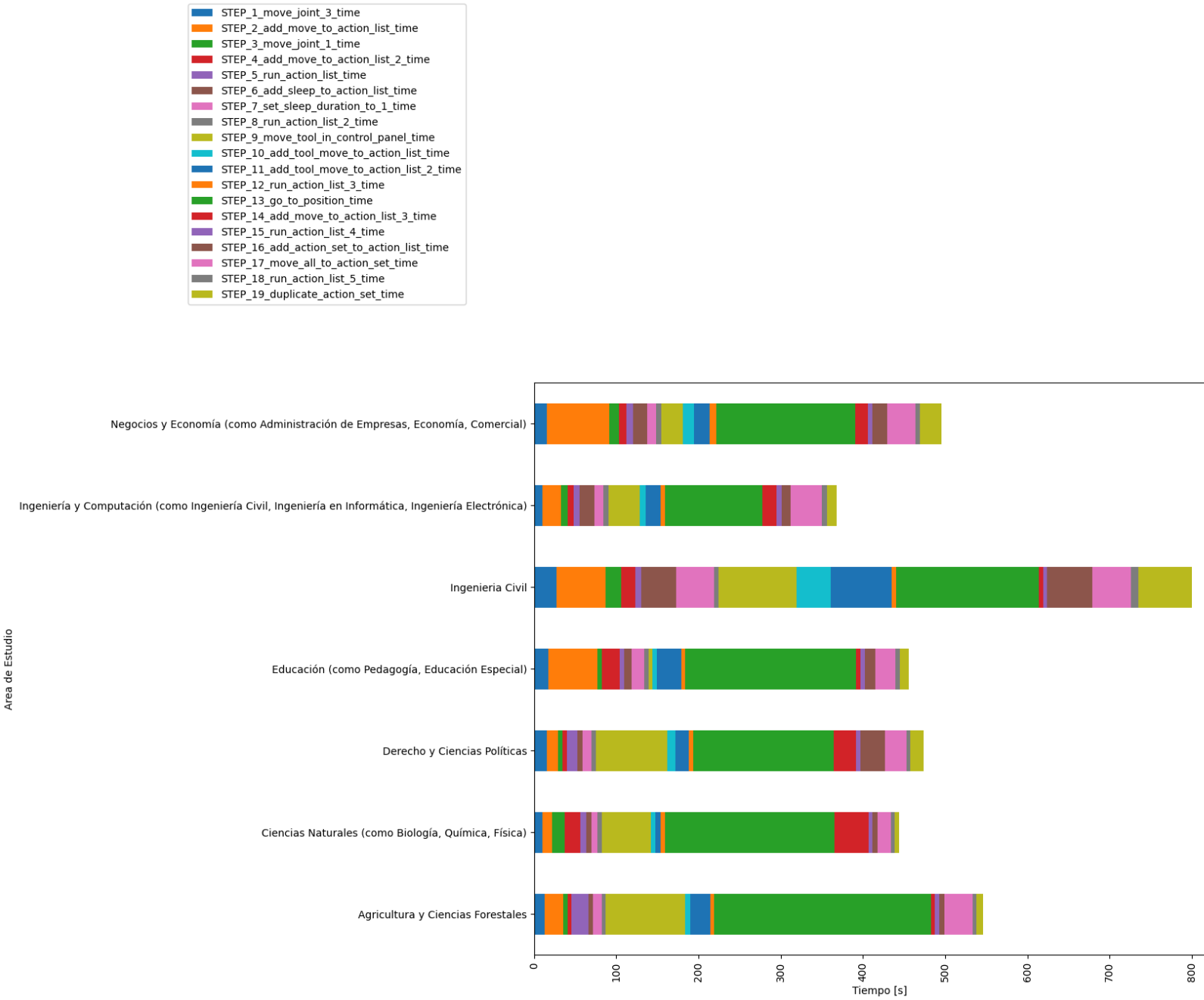

Figura 23:

Tiempo promedio por paso por area de estudio.

Al observar el tiempo promedio que cada grupo tomó para completar cada paso de la actividad, se detectó que, en general, no hay una diferencia significativa entre los tiempos promedio de los distintos grupos. No obstante, se destaca el grupo de ingeniería en computación, el cual mostró el mejor tiempo promedio en todos los pasos. Este resultado podría indicar una mayor familiaridad con la tecnología y los sistemas de control en este grupo específico.

Sin embargo, el análisis más revelador es que el fondo académico o profesional de los participantes no tuvo un impacto significativo en el tiempo requerido para completar la actividad. Esto sugiere que, independientemente del área de estudio y sin experiencia previa en temas relacionados, los usuarios pudieron interactuar con la interfaz de manera eficiente. Tal hallazgo demuestra la accesibilidad y la facilidad de uso de la interfaz, ya que los usuarios no se vieron limitados por la falta de conocimiento previo en el campo.

En base a estos resultados, se puede argumentar que la interfaz desarrollada cumple con el objetivo de ser accesible y utilizable por un amplio espectro de usuarios, independientemente de su experiencia previa en temas de robótica o programación. Por lo tanto, para análisis futuros, es razonable considerar los datos de todos los grupos como un conjunto homogéneo, enfocándose en la experiencia de usuario global en lugar de en diferencias específicas de cada grupo. Esto fortalece la premisa de que la interfaz diseñada es efectivamente inclusiva y adaptable a usuarios con diversos antecedentes y niveles de habilidad.

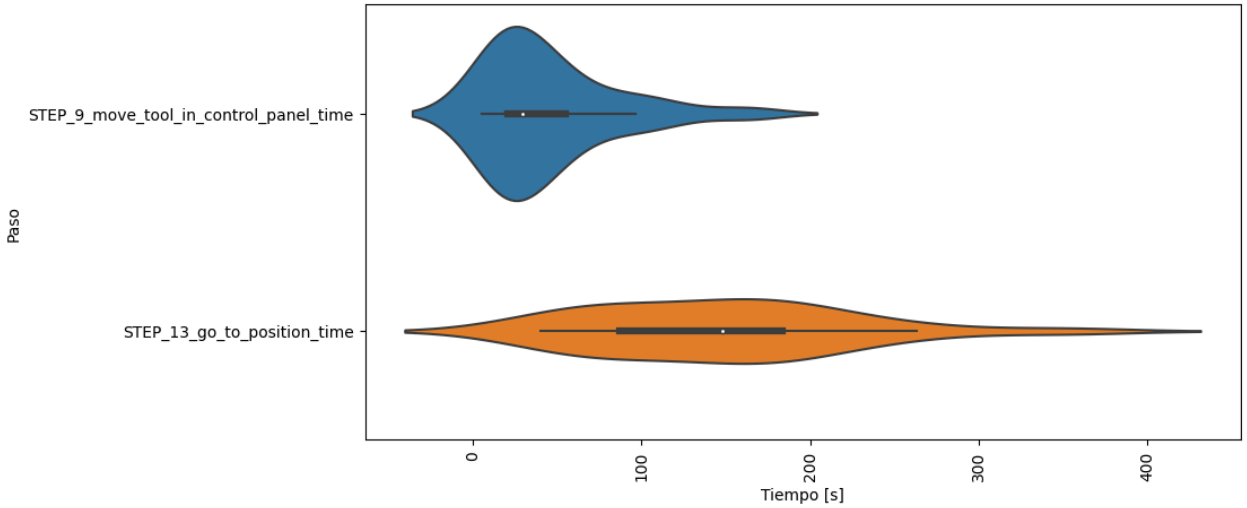

Figura 24:

Gráfico de violín para pasos 9 y 13.

El análisis del tiempo promedio por área de estudio en los pasos de la actividad revela diferencias significativas en dos etapas específicas: el paso 13 y el paso 9. El gráfico de violín para estos pasos proporciona una visión clara de cómo estos dos momentos de la actividad presentaron desafíos distintos para los usuarios.

El **paso 13**, que implica posicionar el brazo robótico en una ubicación específica utilizando coordenadas cartesianas y ángulos de aproximación, es comprensiblemente más desafiante. Este paso requiere una comprensión básica de conceptos espaciales y la habilidad para manipular el brazo utilizando el panel de control o ingresando valores manualmente. La mediana de tiempo para completar este paso fue de aproximadamente 2 minutos y 10 segundos. A pesar de ser el paso más complejo y demandante en términos de comprensión

y ejecución, la mayoría de los usuarios logró completarlo en menos de 3 minutos, lo cual es notable considerando su falta de experiencia previa. La distribución de los tiempos en este paso, como muestra el gráfico de violín, es relativamente uniforme, lo que indica que no hubo un punto claro de dificultad para la mayoría de los usuarios.

En contraste, el **paso 9** presentó un tipo de desafío diferente. Este paso, que consiste en ajustar el valor de la herramienta mediante el panel de control, tuvo una mediana de tiempo de 45 segundos. A pesar de ser un paso conceptualmente más simple, el tiempo promedio relativamente alto sugiere una dificultad en la usabilidad de la interfaz, posiblemente debido a problemas de diseño, especialmente en dispositivos con pantallas más pequeñas. En estas pantallas los usuarios tenían que desplazarse para acceder completamente a los controles de ajuste, lo cual no es evidente. Esto se refleja también en el gráfico de violín, donde se observa que un grupo significativo de usuarios tardó más de lo esperado en completar este paso.

Estos resultados resaltan dos aspectos clave: primero, la capacidad de los usuarios para adaptarse y aprender conceptos nuevos como las coordenadas cartesianas, incluso sin experiencia previa; y segundo, la importancia de un diseño de interfaz que considere todos los tamaños de pantalla para garantizar una experiencia de usuario fluida y sin interrupciones. La identificación de estos puntos de fricción es crucial para futuras mejoras en la interfaz y la experiencia del usuario en general.

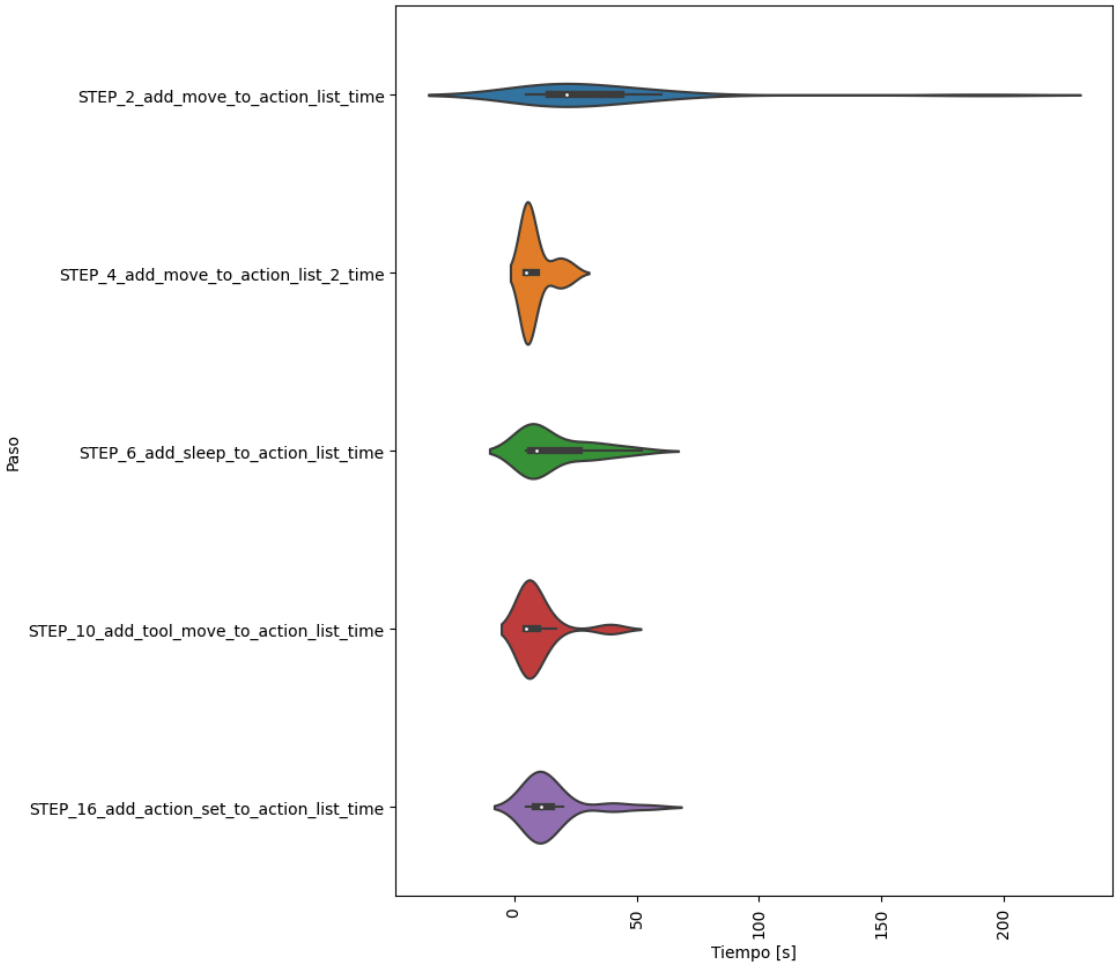

Figura 25:

Violín para tiempos por paso $2, 4, 6, 10$.

El análisis de los tiempos promedio en la realización de acciones generales, especialmente en la adición de bloques de acciones a la lista, ofrece una perspectiva valiosa sobre la curva de aprendizaje y adaptabilidad de los usuarios frente a la interfaz. La figura del gráfico de violín para los pasos 2, 4, 6, 10 y 16 muestra variaciones interesantes en los tiempos de ejecución que merecen una discusión detallada.

Inicialmente, en el **paso 2**, donde se introduce por primera vez la acción de agregar un bloque de movimiento, observamos una media ligeramente más alta y una distribución más amplia. Esto sugiere una fase inicial de adaptación y aprendizaje, donde los usuarios se familiarizan con la nueva tarea de agregar bloques. Sin embargo, en los pasos subsiguientes (4, 6, 10, 16), que implican la introducción de nuevas acciones como el bloque de «sleep» y el movimiento de la herramienta, se aprecia una reducción en los tiempos medios y una distribución más estrecha. Este patrón indica que los usuarios se adaptaron rápidamente al uso de la interfaz y a la incorporación de nuevas acciones, una vez que superaron la barrera inicial del aprendizaje.

Esta observación es consistente con los principios de usabilidad establecidos por Nielsen Norman Group [16], que enfatizan la importancia de interfaces intuitivas y el aprendizaje progresivo para los usuarios. El hecho de que los usuarios pudieran manejar de forma efectiva la adición de nuevas acciones a la lista, y que la curva de aprendizaje disminuyera con cada nueva acción introducida, resalta la eficacia del diseño de la interfaz en facilitar la comprensión y ejecución de tareas complejas.

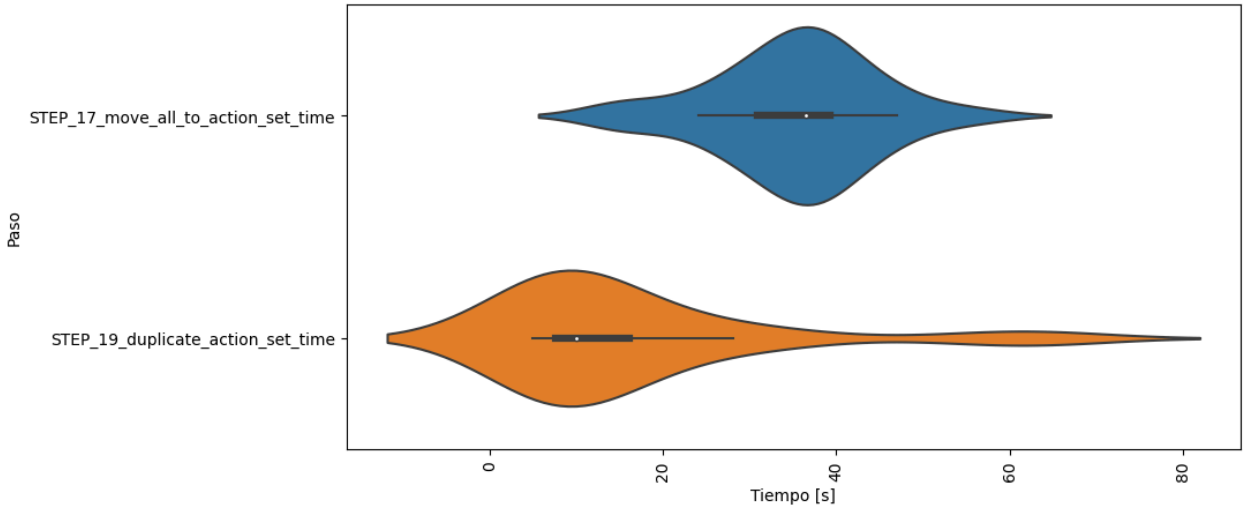

Figura 26:

Violín para tiempos por paso 17, 19.

En cuanto al **paso 17**, donde se implementó la lógica de arrastrar y soltar (drag and drop) para mover bloques de acciones, la distribución de los tiempos se asemeja a una normal, con una media de 40 segundos. Esto sugiere que la funcionalidad de arrastrar y soltar fue rápidamente comprendida y utilizada de manera eficiente por la mayoría de los usuarios, lo cual es un indicativo de una buena implementación de esta característica en la interfaz.

Finalmente, el **paso 19** presentó un desafío distinto. Aunque la media de tiempo para duplicar un bloque de acciones fue relativamente baja (15 segundos), se observaron algunos resultados que se desviaron significativamente de esta media. Esto podría deberse a la utilización de un icono no estándar para el menú contextual durante las pruebas, lo que pudo haber causado confusión entre los usuarios menos experimentados. Este hallazgo refuerza la necesidad de adherirse a iconografías y patrones de diseño familiarizados por los usuarios, como se recomienda en las heurísticas de Nielsen para la usabilidad, para garantizar una experiencia de usuario coherente y eficiente.

En resumen, los resultados del análisis sugieren que, aunque hubo desafíos iniciales en la comprensión y uso de la interfaz, los usuarios lograron adaptarse rápidamente y llevar a cabo tareas más complejas de manera eficiente. Estos hallazgos apoyan la eficacia del diseño de la interfaz en términos de facilitar el aprendizaje progresivo y la realización de

tareas, alineándose con los principios de usabilidad reconocidos en el campo del diseño de interacción.

5.5. Conclusiones de la Evaluación

Las conclusiones derivadas de la evaluación realizada sobre la interfaz del sistema de control del brazo robótico son fundamentales para comprender la eficacia y accesibilidad de la solución propuesta. A partir de los análisis de los tiempos promedio por paso y área de estudio, se pueden extraer varias conclusiones clave:

1. **Cumplimiento de la Hipótesis**: Los resultados confirman la hipótesis inicial de que la interfaz es accesible y fácil de usar para una amplia gama de usuarios, independientemente de su área de estudio. La consistencia en los tiempos promedio por paso entre los distintos grupos de usuarios sugiere que la interfaz logra una buena usabilidad transversal.
2. **Tiempo Promedio Aceptable en Pasos Relevantes**: La mayoría de los pasos relevantes, como agregar acciones a la lista y utilizar nuevas funcionalidades, mostraron tiempos promedios razonables. Esto indica que los usuarios lograron comprender y utilizar la interfaz de manera efectiva, lo que refleja una curva de aprendizaje positiva y una adaptación rápida a las tareas propuestas por la interfaz.
3. **Identificación de Problemas de Diseño:** A pesar de los resultados positivos, se identificaron áreas de mejora, especialmente en los pasos 9 y 19, donde se observaron tiempos promedios más largos o una gran variación en los datos. Estas observaciones apuntan a posibles fallas de diseño o falta de claridad en la interfaz, lo que puede afectar negativamente la experiencia del usuario. En particular, los problemas relacionados con la accesibilidad de los controles en pantallas más pequeñas y la confusión generada por el uso de iconos no estándar en el menú contextual son aspectos críticos que requieren atención y rectificación.

En conclusión, la evaluación muestra que, si bien la interfaz cumple en gran medida con sus objetivos de usabilidad e intuitividad, hay oportunidades claras para mejorar y refinar el diseño. Abordar estas áreas no solo mejorará la experiencia del usuario, sino que también reforzará la accesibilidad y efectividad de la interfaz para un espectro aún más amplio de usuarios. Estos hallazgos proporcionan una base sólida para futuras iteraciones y mejoras en el diseño y la funcionalidad de la interfaz.

6. Trabajo Futuro

Las posibilidades de trabajo futuro en este proyecto abarcan diversas áreas, ofreciendo oportunidades significativas para mejorar y expandir tanto la librería de control, como la interfaz y el aspecto del hardware. Estas mejoras no solo aumentarán la funcionalidad del sistema sino también su accesibilidad y eficacia en una variedad más amplia de aplicaciones y contextos.

Interfaz:

- **Adaptación Dinámica del Modelo Visual**: Una mejora clave sería la implementación de un modelo visual dinámico en la interfaz. Esto permitiría que la representación visual del brazo robótico se ajuste automáticamente según las dimensiones específicas de diferentes modelos. Tal adaptabilidad haría que la interfaz sea más versátil y representativa de una variedad de brazos robóticos.
- **Mejoras de Diseño UI/UX**: Otra área de mejora es el diseño de la interfaz, especialmente en términos de responsividad para diferentes tamaños de pantalla y optimización del uso del espacio. Trabajar en estos aspectos mejorará la experiencia del usuario, haciendo que la interfaz sea más intuitiva y accesible en una gama más amplia de dispositivos.

Librería de Control y Firmware:

- **Mejora de Movimientos y Funcionalidades**: Se puede trabajar en la mejora de los movimientos del brazo robótico, incluyendo la capacidad de seguir trayectorias más complejas. La integración de sensores de fuerza y otros tipos de sensores expandiría las capacidades de control y percepción del brazo, aumentando su aplicabilidad en tareas más sofisticadas y en entornos que requieren mayor precisión y adaptabilidad.
- **Implementación de Aceleración en Motores**: Actualmente, el sistema no maneja la aceleración en los motores, por lo que una futura mejora podría ser la incorporación de esta característica. Implementar un control de aceleración no solo mejoraría el rendimiento del brazo robótico, sino que también podría prolongar la vida útil de los motores y del conjunto del brazo en sí, al reducir el estrés mecánico durante los movimientos.
- **Aspectos de Seguridad y Capa de Protección:** Fundamental en el desarrollo futuro es la integración de robustos mecanismos de seguridad en la librería de control y el firmware. Esto incluiría la implementación de protocolos de seguridad para prevenir malfuncionamientos y la adición de una capa de protección que supervise el funcionamiento del brazo robótico, detectando y respondiendo a condiciones anormales o potencialmente peligrosas. Mejorar la seguridad es crucial, especialmente en aplicaciones donde el brazo interactúa con humanos o realiza tareas en entornos sensibles.

Estas mejoras y expansiones propuestas no solo aumentarán la eficacia del sistema actual, sino que también abrirán nuevas vías para su uso en aplicaciones más avanzadas y en entornos educativos y de investigación, ampliando el alcance y el impacto del proyecto.

7. Conclusiones

El proyecto presentado logró cumplir con éxito todos los objetivos planteados inicialmente. Se creó una librería de control en Python que facilita la manipulación de un brazo robótico de seis grados de libertad, caracterizándose por su facilidad de adaptación a otros brazos robóticos con modelos matemáticos similares.

Uno de los logros más destacados fue el desarrollo de una interfaz intuitiva y simple, que permite el control eficiente del brazo robótico. Esta interfaz resultó ser accesible para usuarios sin experiencia previa en robótica, abriendo así el campo de la robótica a un público más amplio.

Además, se desarrolló un firmware versátil que permite el control de motores paso a paso, servomotores y sensores de fin de carrera. Este firmware, al igual que la librería de control, se diseñó para ser fácilmente adaptable a una variedad de brazos robóticos con estructuras matemáticas equivalentes.

Otro aspecto clave del proyecto fue la validación exitosa en hardware existente, así como la comprobación de la usabilidad de la interfaz con usuarios de diversas áreas de estudio y sin conocimientos previos en el tema. Estas pruebas confirmaron la eficacia y accesibilidad de la solución propuesta.

La implementación del sistema se ha simplificado gracias al uso de Docker, eliminando la dependencia de múltiples configuraciones de software y facilitando la implementación en diversos sistemas operativos. Esto representa una ventaja significativa para usuarios interesados en implementar el sistema desde cero en un brazo robótico, especialmente aquellos sin conocimientos avanzados en programación o sistemas complejos como ROS. En resumen, el proyecto ofrece una forma completa, fácil e interactiva de operar un brazo robótico impreso en 3D de código abierto, democratizando el acceso a la tecnología robótica avanzada.

En conclusión, el proyecto tuvo un resultado positivo, ofreciendo un sistema de control viable para brazos robóticos de seis grados de libertad. Esta solución es ideal para entornos donde no se requieren conocimientos previos de computación o robótica. Asimismo, la librería de control provee una herramienta sencilla y fácil de usar, aprovechando la popularidad y accesibilidad de Python como lenguaje de programación para principiantes.

BIBLIOGRAFÍA

[1] RobotLAB, «Dobot Magician Robotic Arm - Advanced Educational Plan». [En línea]. Disponible en: https://www.robotlab.com/store/dobot-robotic-arm

[2] Dobot, «Cobot in Education». [En línea]. Disponible en: https://www.dobot-robots.com/industries/education.html

[3] BCN3D, «GitHub - BCN3D/BCN3D-Moveo: Open Source 3D Printed Robotic Arm for educational purposes». [En línea]. Disponible en: https://github.com/BCN3D/BCN3D-Moveo

[4] «Thor - Open Source 3D Printed Robotic Arm». [En línea]. Disponible en: https://hackaday.io/project/12989-thor

[5] Arctos Robotics, «Arctos Robotics - Arctos 3D Printed Robot Arm». [En línea]. Disponible en: https://arctosrobotics.com/

[6] ROS, «Why ROS?». [En línea]. Disponible en: https://www.ros.org/blog/why-ros/

[7] Wiktionary, «Abr_control repository». [En línea]. Disponible en: https://github.com/abr/abr_control/tree/main

[8] Jesse Weisberg, «Moveo ROS - ROS package for the BCN3D Moveo robotic arm». [En línea]. Disponible en: https://github.com/jesseweisberg/moveo_ros

[9] «Udemy - Industrial Robotics». [En línea]. Disponible en: https://www.udemy.com/course/industrial-robotics/

[10] «Euler angles - Wikipedia». [En línea]. Disponible en: https://en.wikipedia.org/wiki/Euler_angles

[11] «Documentacion del Proyecto». [En línea]. Disponible en: https://ribot.dev/

[12] «Codigo Fuente del Proyecto». [En línea]. Disponible en: https://github.com/alberto-abarzua/ribot

[13] «10 Usability Heuristics for User Interface Design». [En línea]. Disponible en: https://www.nngroup.com/articles/ten-usability-heuristics/

[14] «Video Demostrativo del Proyecto». [En línea]. Disponible en: https://www.youtube.com/watch?v=HM4zTVW9VRM&ab_channel=AlbertoAbarzua

[15] «Quantitative Studies: Quantitative vs. Qualitative Usability Testing». [En línea]. Disponible en: https://www.nngroup.com/articles/quant-vs-qual/

[16] «Usability 101: Introduction to Usability». [En línea]. Disponible en: https://www.nngroup.com/articles/usability-101-introduction-to-usability/