UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@
Computación e Informática

Informe Fase 3
“Brazo Robótico de agarre con LEGO”

Alumnos: Carlos Cossio
Bryan Palacios
Franco Churata
Benjamín Aguilera
Joaquín Quezada

Asignatura: Proyecto I

Profesor: Baris Nikolai
Klobertanz Quiroz

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
26/09/2025	1.0	Concepción del documento.	Carlos Cossio
29/09/2025	1.1	Recopilación de datos del proyecto.	Benjamín Aguilera
1/10/2025	1.2	Redacción de ítems iniciales.	Franco Churata
6/10/2025	1.3	Inclusión de planificación de recursos y riesgos.	Carlos Cossio
10/10/2025	1.4	Última revisión y pulimento de detalles.	Bryan Palacios
17/10/2025	1.5	Finalización del informe 1.	Carlos Cossio
11/12/2025	1.6	Avance del informe 2.	Bryan Palacios
12/12/2025	1.7	Revisión de detalles del informe.	Carlos Cossio, Benjamín Aguilera
14/12/2025	1.8	Finalización del informe 2.	Bryan Palacios, Joaquin Quezada, Franco Churata
28/12/2025	1.9	Planificación del Informe 3	Carlos Cossio, Bryan Palacios
30/12/2025	2.0	Finalización del Informe 3	Benjamín Aguilera, Joaquin Quezada, Franco Churata

Tabla de Contenidos

1. Panel General

1.1. Introducción

El vehículo robótico requiere un sistema independiente que permita cargar los bloques de manera óptima y así completar correctamente la cadena de transporte. Durante el desarrollo del semestre, el grupo llevará a cabo la construcción, integración y programación de un brazo robótico de agarre. El objetivo principal es diseñar un sistema automatizado capaz de sujetar, levantar y depositar bloques en un vehículo robótico de transporte, el cual será ensamblado y programado por otro grupo dentro del mismo proyecto colaborativo.

Este proyecto forma parte de una colaboración entre tres equipos, donde cada uno cumple un rol definido: diseño del brazo de agarre, desarrollo del vehículo autónomo y coordinación del sistema conjunto mediante comunicación y control sincronizado.

El trabajo se centra en las etapas de experimentación mecánica, diseño estructural y cinemático, ensamblaje funcional, programación del sistema de control y documentación técnica del proceso. Asimismo, se aplican metodologías de trabajo colaborativo, gestión de tareas mediante herramientas digitales y principios de ingeniería de control y automatización, con el propósito de lograr una integración eficiente entre los distintos subsistemas del proyecto.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar y programar un brazo robótico de agarre que permita transferir bloques hacia un vehículo de carga automatizado, mediante una interfaz manual. Esta herramienta beneficiará al sistema completo del proyecto, ya que asegura una transferencia eficiente y confiable de materiales, mejorando la coordinación y el desempeño global dentro del entorno experimental.

1.2.2. Objetivos Específicos

- Analizar distintos diseños de sistemas de agarre y mecanismos de sujeción, evaluando su eficiencia estructural y capacidad de manipulación mediante piezas LEGO, con el fin de identificar el diseño más eficiente y funcional según los criterios de estabilidad y facilidad de uso.
- Diseñar y ensamblar un modelo funcional del brazo robótico, asegurando que presente estabilidad mecánica y precisión en el movimiento de agarre. Se evaluará la estabilidad mediante pruebas de carga, y el brazo robótico deberá ser capaz de realizar tareas de sujeción de objetos de manera eficiente y sin fallos durante su funcionamiento.
- Programar la secuencia de movimientos del sistema de agarre utilizando la plataforma de programación visual y posteriormente migrar el control a código Python para una mayor flexibilidad.
- Coordinar y colaborar con los demás grupos involucrados en el proyecto para sincronizar las funciones de carga y descarga entre el brazo robótico y el vehículo automatizado.
- Documentar el proceso de desarrollo de software mediante informes técnicos y una bitácora semanal, en la cual un integrante del equipo se encargará de registrar los avances, ajustes y resultados de cada etapa del proyecto. El encargado de hacer la bitácora cambiará semanalmente, rotando entre los miembros del equipo para fomentar el aprendizaje integral de todos los integrantes en la gestión de tareas del proyecto.

1.3. Restricciones

- **Incompatibilidad Nativa de Interfaces Gráficas (GUI):** Debido a las limitaciones de hardware y al uso de Micro Python orientado al control de dispositivos, el Hub SPIKE Prime no soporta librerías para interfaces gráficas complejas. Como consecuencia, cualquier interfaz de control visual debe ejecutarse en un dispositivo externo, delegando el procesamiento gráfico a un computador que se comunica con el robot mediante Bluetooth u otro medio, lo que da lugar a una arquitectura de software distribuida.
- **Plataforma de Hardware:** El sistema debe desarrollarse exclusivamente sobre la arquitectura LEGO SPIKE Prime (núcleo 45678) y su Kit de Expansión (45681). Todos los

sensores y actuadores utilizados deben ser compatibles con el firmware oficial o con Pybricks, lo que restringe la selección de componentes y capacidades del sistema.

- **Latencia de Comunicación:** Debido al uso del protocolo Bluetooth Low Energy (BLE) para la transmisión de comandos en tiempo real, el sistema está sujeto a latencias y posibles interrupciones breves de comunicación. El software debe considerar estos retardos para evitar pérdidas de control, tales como desviaciones de posición del brazo o caída de la carga.
- **Interoperabilidad con Vehículo Autónomo:** El diseño mecánico y la programación del brazo están condicionados por las dimensiones y características del Vehículo de Transporte desarrollado por el Grupo 3. La altura de descarga y el punto de entrega deben coincidir con el receptáculo del vehículo para garantizar la correcta continuidad de la cadena de transporte.
- **Recursos Humanos y Temporales:** El proyecto debe ser desarrollado por un equipo de hasta cinco integrantes, lo que limita la carga de trabajo asumible y exige una distribución clara de roles en las áreas de mecánica, programación y documentación, con el fin de cumplir los plazos establecidos por el calendario académico

1.4. Entregables

- Bitácoras semanales: informe del avance del grupo, rotando el responsable cada semana.
- Carta Gantt: cronograma del proyecto con las actividades principales.
- Informe de Formulación: documento actual que describe la organización, planificación y objetivos.
- Presentación Final: exposición del funcionamiento del brazo robótico y los resultados obtenidos.
- Código base del cliente y servidor, publicados en un repositorio en Github.
- Una página del proyecto en la plataforma Redmine.
- Manual de usuario

2. Organización del Personal

Con el propósito de asegurar el desarrollo eficiente del proyecto, se ha definido una estructura interna de trabajo fundamentada en las habilidades individuales de cada miembro del equipo. Esta organización permite una distribución equitativa de las responsabilidades, garantizando que todas las tareas sean abordadas de forma apropiada.

2.1. Descripción de los Roles

Para organizar de manera eficiente el trabajo en equipo y asegurar que el proyecto avance correctamente, se realizó un análisis de habilidades e intereses de cada integrante. A partir de esto, se definieron los roles que lograron cubrir todas las áreas necesarias para el desarrollo del robot, aparte se optó por un sistema de rotación semanal para que todos los integrantes pudieran experimentar cada función del proyecto.

Jefe de Proyecto / Documentador: coordina el grupo, redacta informes y bitácoras.

Ensamblador: encargado del armado y de probar el funcionamiento del brazo robótico.

Programador: desarrollan los movimientos y controles del brazo en la app Spike Prime y Python.

Diseñador: se encarga de proponer mejoras de diseño, asegurar el funcionamiento y que se vea bien estéticamente el brazo robótico.

2.2. Personal que Cumplirá los Roles

Rol	Responsable
Jefe de Proyecto / Documentador	Carlos Cossio
Ensamblador	Franco Churata, Bryan Palacios
Programador	Joaquín Quezada
Diseñador	Benjamin Aguilera

2.3. Métodos de Comunicación

- WhatsApp: para coordinación rápida y avisos inmediatos.
- Discord: para reuniones de trabajo online
- Reuniones presenciales: durante las clases de taller.
- Redmine: para entrega de documentos y bitácoras semanales.

3. Planificación del Proyecto

3.1. Actividades

A continuación, se detallan las actividades planificadas para el desarrollo del brazo robótico. Esta planificación está estructurada para seguir un flujo lógico y progresivo que se realizó durante el avance del proyecto.

Nombre	Descripción	Responsable	Producto
Investigación	Revisión de ideas, análisis de modelos y revisión de compatibilidad de piezas.	Todo el grupo	Diseño Conceptual

Ensamble	Construcción del modelo base y conexión de piezas	Franco Bryan	Garra parcialmente funcional
Documentación	Registro de bitácoras semanales y elaboración del informe	Todo el grupo	Robot en Movimiento
Codificación	Programación de movimientos básicos con bloques con el software Lego MINDSTORMS	Joaquín Quezada	Código en Python funcional
Presentación Final	Preparación de exposición y demostración	Todo el grupo	Presentación del proyecto

3.2. Carta Gantt

La Carta Gantt representa la visualización cronológica de todas las actividades detalladas, junto a su estado de avance. Esta herramienta es fundamental para la gestión de este proyecto, ya que permite al grupo ver los plazos, la asignación de cada tarea y el siguiente de esta, asegurando el cumplimiento de las actividades de manera organizada y eficiente.

La **Figura 1** detalla la planificación temporal de la fase 1 del proyecto (Investigación, Documentación, Ensamble y Codificación), mostrando el avance que se realizó durante el semestre. Esta fase del proyecto tiene una duración estimada de 5 semanas.

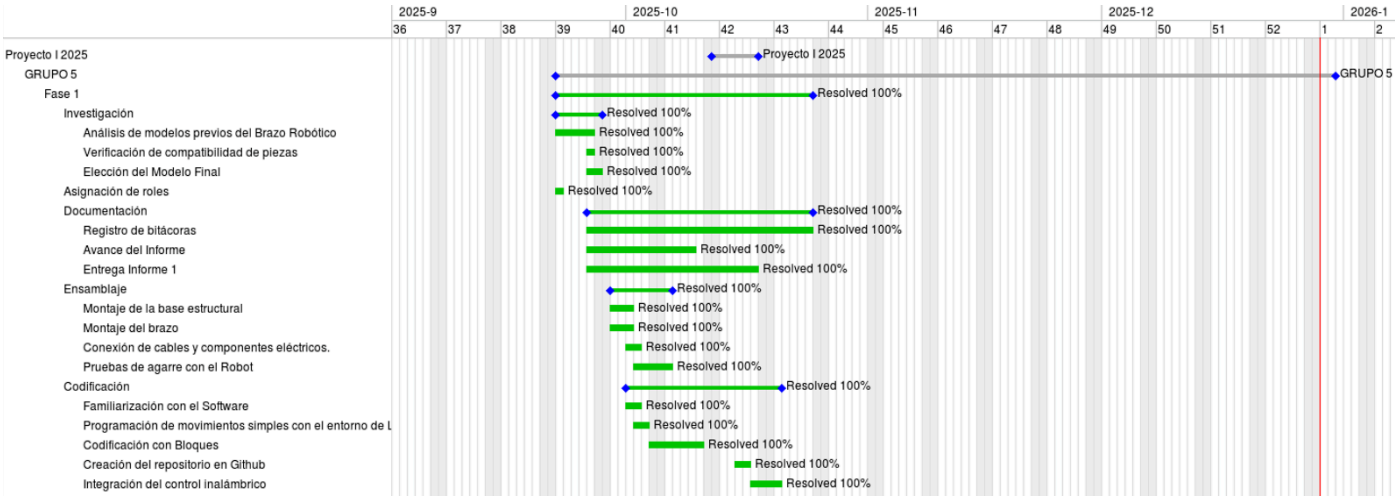

Figura 1: *Fase 1 de la Carta Gantt.*

La **Figura 2** detalla la planificación temporal de la fase 2 del proyecto (Documentación, Actividades del Brazo Robótico, Reensamble y Codificación), mostrando el avance que se realizó durante el semestre. Esta fase del proyecto tiene una duración estimada de 8 semanas.

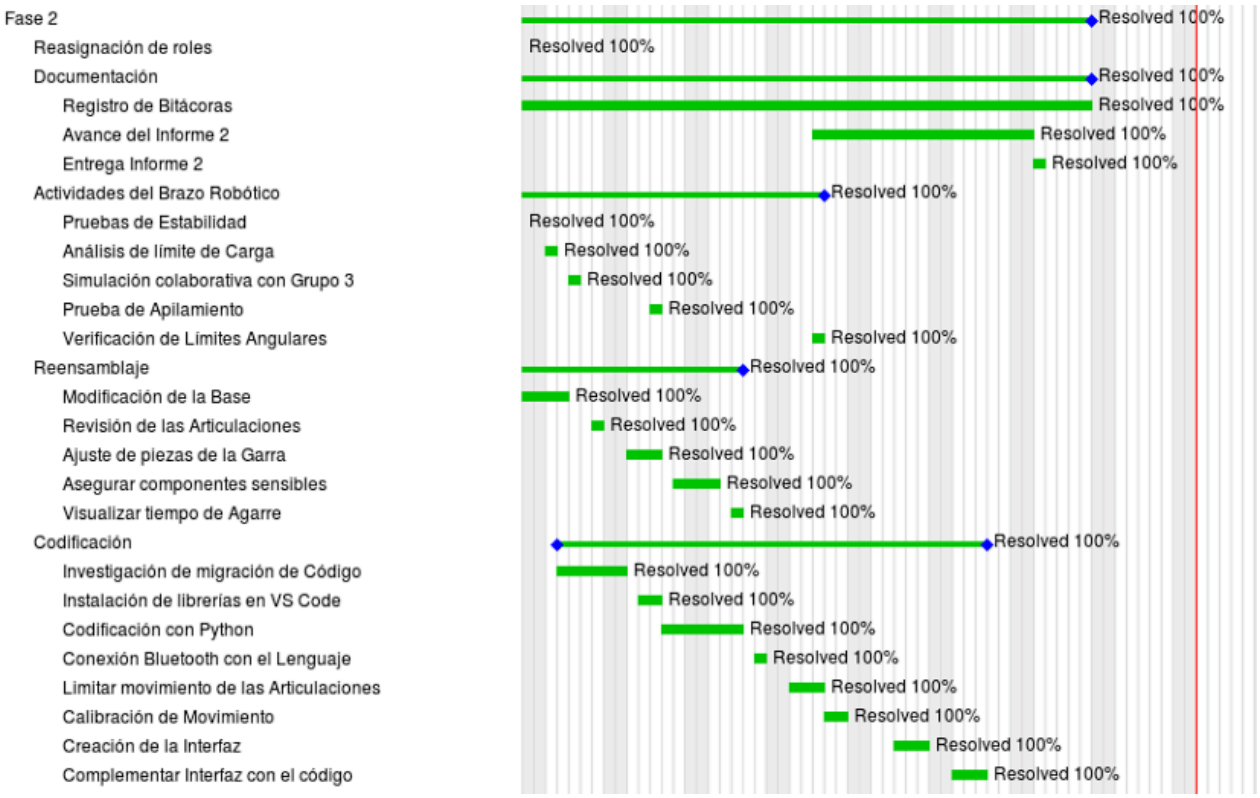

Figura 2: *Fase 2 de la Carta Gantt.*

La **Figura 3** detalla la planificación temporal de la fase 3 del proyecto (Ajustes Mecánicos, Codificación Final del Sistema, Pruebas Finales, Documentación Final, Presentación y Evaluación), mostrando el avance que se realizó durante el semestre. Finalmente, esta fase del proyecto presenta una duración estimada de 3 semanas.

Figura 3: *Fase 3 de la Carta Gantt.*

3.3. Gestión de Riesgos

En esta sección se realiza para identificar, analizar y planificar respuestas a eventos que podrían afectar negativamente los objetivos del proyecto. A continuación, se detallan los riesgos identificados, el nivel de su impacto y las acciones para solucionar esos tipos de eventos.

El Nivel de Impacto se definió utilizando una escala del 1 al 5, donde un valor más alto representa una amenaza más grave para el cumplimiento de los objetivos del proyecto:

- 1 (Bajo): riesgo que tiene una solución sencilla o con un impacto mínimo en el proyecto.
- 3 (Medio): riesgo que requiere tiempo y recursos adicionales, afectando el tiempo de trabajo.
- 5 (Alto): riesgo que amenaza la viabilidad o el alcance del proyecto.

Riesgos	Nivel de Impacto	Acción Remedial
Incompatibilidad con mando externo	1	Buscar alternativas dentro del software Spike Prime o mediante Python.
Pérdida del robot o desarme entre clases	2	Guardar el robot en un lugar seguro y registrar su estado al finalizar cada sesión.
Falta de tiempo y carga académica	3	Coordinar mejor los horarios y distribuir tareas entre los miembros.
Falta de piezas	3	Solicitar piezas de reemplazo o usar piezas del kit de expansión.
Errores de codificación	4	Revisar código en equipo y comparación ejemplos de Spike Prime.

4. Planificación de los Recursos

En esta sección presentamos la planificación de los recursos necesarios para llevar a adelante nuestro proyecto. Para organizar adecuadamente el trabajo , identificamos tres tipos de recursos fundamentales. Primero, los recursos de hardware, esto incluye todos los elementos físicos usados para construir y probar el Brazo robótico , después tenemos los recursos de software los cuales son gratuitos los cuales abarcan las herramientas digitales que usamos para programar , documentar y gestionar el proyecto , Por último consideramos los recursos financieros donde tenemos en cuenta los costos asociados tanto a los materiales , el tiempo invertido por el equipo.

Para calcular los costos, usamos un método basado en las horas dedicadas y un valor referencial por hora , todo esto en CLP. Esto nos permitió determinar un costo estimado por integrante, esto nos llevó a ver mejor el costo total del proyecto.

4.1. Hardware

Lego Spike Prime
Kit de Expansión
Portátiles

Software

Spike Prime App
Visual Studio Code
Python(MicroPython)
Redmine

4.2. Estimación de costos

La estimación de los costos totales del proyecto se basó en la suma de los recursos materiales (hardware) y el costo de la dedicación del equipo (mano de obra).

Cálculo de la Mano de Obra

El costo de la mano de obra se calculó considerando una tarifa de $3.000 CLP por hora y una dedicación individual de 20 horas.

Concepto	Valor por Hora	Horas por Integrante	Costo por Integrante
Mano de Obra	$7.000 CLP	20	$140.000 CLP

Recursos de Hardware

A continuación, se detalla el listado del equipo utilizado:

Recurso	Cantidad	Precio Unitario (CLP)
Lego Spike Prime	1	$600.000
Kit de Expansión	1	$150.000
Portátil (Suministrado por la Universidad)	5	$350.000

Resumen de la Estimación Total de Costos

El costo total estimado para el desarrollo del proyecto asciende a $3.200.000 CLP, distribuido de la siguiente manera:B

Concepto	Costo Total (CLP)
Hardware	$2.500.000
Mano de Obra	$700.000
Total Estimado	**$3.200.000**

5. Análisis y Diseño

El análisis y diseño del sistema constituyen una etapa crítica dentro del desarrollo del brazo robótico de agarre, ya que permiten establecer con claridad los requisitos funcionales del sistema, las características de calidad que debe poseer para operar de manera confiable y la arquitectura de software necesaria para articular la interacción entre el usuario y el robot. Esta sección presenta el proceso de especificación de requerimientos, el diseño de la arquitectura cliente-servidor que soporta el control del prototipo y la propuesta preliminar de la interfaz gráfica de usuario, la cual facilitará la interacción entre el operador y el brazo robótico.

El propósito principal del diseño es asegurar que el sistema sea operable, mantenible y extensible, de manera que pueda integrarse adecuadamente dentro del proyecto colaborativo más amplio al cual pertenece, manteniendo coherencia tanto en su funcionamiento mecánico como en su estructura de software.

5.1. Especificación de requerimientos

Problema a resolver

En el contexto de una operación minera que ha experimentado accidentes asociados a labores de transporte, carga y descarga de mineral, surge la necesidad de automatizar estos procesos para reducir la exposición de los trabajadores a zonas de riesgo. La empresa requiere un sistema capaz de manipular materiales de manera segura, precisa y sin intervención humana directa.

Como representación en el ámbito educativo de este escenario industrial, el presente proyecto desarrolla un sistema automatizado compuesto por un organizador de material, un vehículo autónomo y un brazo robótico encargado de la carga y descarga. En particular, el brazo debe ser capaz de recoger y depositar bloques que en el contexto académico representan unidades de mineral de forma confiable. Para cumplir con esta necesidad, el equipo construirá y programará un brazo robótico utilizando LEGO Spike Prime, controlado mediante una interfaz gráfica o un mando externo.

Usuario del sistema

En el contexto minero, el usuario corresponde al operador encargado de supervisar el sistema automatizado desde una zona segura, monitoreando la manipulación del material sin intervenir físicamente en la faena. Este usuario requiere una interfaz clara, simple y confiable para minimizar riesgos y garantizar continuidad operativa.

En el contexto del proyecto universitario, el usuario es el integrante del equipo designado para operar el brazo robótico durante las pruebas y demostraciones. Esta persona interactúa directamente con la interfaz gráfica o con el mando para ejecutar las acciones del sistema,

verificar el funcionamiento del agarre y asegurar que el brazo cumpla correctamente las tareas de carga y descarga simuladas.

Cliente del sistema

En el entorno minero, el cliente corresponde a la empresa operadora que busca automatizar sus procesos de transporte y manipulación de mineral con el fin de mejorar la seguridad, reducir incidentes y optimizar la eficiencia de la faena.

En el marco académico, el cliente es el profesor a cargo del proyecto, quien define los objetivos, supervisa el desarrollo del prototipo y evalúa los resultados finales. El profesor actúa como la entidad que “solicita” la solución técnica que el equipo desarrolla.

5.1.1. Requerimientos Funcionales

Los siguientes requerimientos establecen las funciones esenciales que el sistema deberá cumplir. Estos se definieron considerando las capacidades técnicas del kit LEGO Spike Prime, las restricciones impuestas por el entorno de trabajo y los objetivos específicos del proyecto.

Requerimientos Funcionales (RF):

- **RF1:** El sistema logra abrir la garra mediante un comando enviado desde la interfaz gráfica.
- **RF2:** El sistema permite que la garra pueda cerrarse mediante un comando enviado desde la interfaz gráfica.
- **RF3:** El brazo robótico debe de poder elevarse en una posición segura para el transporte del bloque.
- **RF4:** El brazo robótico debe ser capaz de ajustar su altura (ascendiendo o descendiendo) hasta alcanzar la posición de agarre, la cual se define como la altura variable donde se encuentra situado el bloque objetivo a manipular.
- **RF5:** El mecanismo de apertura del brazo robótico debe permitir soltar el bloque cuando el comando correspondiente sea ejecutado.
- **RF6:** El Cliente (Interfaz Gráfica) debe enviar las instrucciones de control hacia el Servidor (Hub Spike Prime) utilizando el canal de comunicación inalámbrico establecido.

- **RF7:** El Servidor debe accionar los motores correspondientes en función del comando específico recibido desde el Cliente.

- **RF8:** La GUI debe mostrar al usuario botones que le permitan ejecutar las acciones específicas del brazo robótico. Estas acciones son: Abrir la garra, Cerrar la garra, Elevar el brazo, Descender el brazo y Rotar el brazo.

- **RF9:** El brazo robótico debe girar 360 grados sobre su eje principal.

- **RF10:** El brazo debe iniciar su funcionamiento en una posición inicial definida según sus motores.

- **RF11:** La GUI debe mostrar información al usuario acerca del estado de la conexión entre cliente y servidor.

5.1.2. Requerimientos No Funcionales

- **Requerimientos No Funcionales (RNF): Criterios de Calidad del Prototipo**

  Los requerimientos no funcionales definen las características de calidad que el sistema debe poseer para garantizar que los requerimientos funcionales se cumplan de manera adecuada. Estas condiciones son indispensables para asegurar la estabilidad, seguridad y eficiencia del prototipo en operación.

ID	Requisito No Funcional	Descripción y Justificación	RF Relacionados
RNF1	**Estabilidad de la Comunicación**	La conexión inalámbrica entre el cliente y el servidor debe mantenerse ininterrumpida durante toda la operación del brazo, permitiendo un porcentaje de tolerancia de hasta 95% de cortes breves (menores a 100 milisegundos) que no afecten la ejecución de movimientos críticos.	RF7, RF8, RF12

RNF2	**Tiempo de Respuesta (Latencia)**	El sistema debe procesar y ejecutar cada comando de la interfaz en un intervalo de 200–500 ms como máximo, asegurando un control fluido y en tiempo real del brazo robótico.	RF1, RF2, RF3, RF4, RF5, RF6, RF9
RNF3	**Precisión Posicional**	Los motores deben ejecutar los movimientos de elevación, descenso, agarre, apertura, cierre y rotación sin desviaciones que comprometan la estabilidad o la orientación del bloque manipulado. La desviación máxima permitida en cualquier eje es de 0.5 mm.	RF3, RF4, RF5, RF10
RNF4	**Fiabilidad del Agarre**	El mecanismo de sujeción debe mantener el bloque firme durante todo el ciclo de movimiento, incluyendo rotaciones de 360°, eliminando el riesgo de caída. Se establece un criterio de fiabilidad de agarre de 0% de caídas tolerables durante el ciclo de movimiento.	RF5, RF10
RNF5	**Seguridad Operacional**	El sistema debe garantizar la integridad física del brazo robótico y la estabilidad de la carga durante todas las fases de operación. Para evaluar el atributo de Seguridad Operacional, se establece un umbral de velocidad máxima de 45 RPM (revoluciones por minuto) para todos los motores del sistema.	RF3, RF4, RF5, RF10
RNF6	**Usabilidad de la Interfaz (GUI)**	El sistema debe permitir que un operador con conocimientos básicos de robótica pueda controlar el brazo de manera eficiente sin necesidad de capacitación extensa. Para evaluar el	RF1, RF2, RF3, RF4, RF6, RF9

		atributo de Usabilidad, se establece una métrica de "Tiempo de Familiarización" máximo de 60 segundos.	
RNF7	**Robustez y Tolerancia a Fallos**	El sistema debe garantizar la continuidad operativa mediante el manejo de errores, estableciendo una tolerancia de 3 reintentos automáticos en 5 segundos ante fallos de red antes de declarar una desconexión crítica.	RF7, RF8, RF12
RNF8	**Mantenibilidad del Código**	El software debe ser fácil de modificar y escalar, estableciendo como métrica un Índice de Mantenibilidad (MI) mínimo de 75/100 para los scripts de Python y MicroPython.	Todos los RF del sistema
RNF9	**Consistencia de Ejecución**	El sistema debe ejecutar los comandos de control del brazo robótico de forma consistente, asegurando que, ante una misma secuencia de instrucciones, los movimientos y posiciones finales sean equivalentes en cada ejecución.	RF1, RF2, RF3, RF4, RF5, RF10
RNF10	**Disponibilidad Continua**	El sistema debe mantenerse operativo de forma continua durante la ejecución prolongada de comandos, garantizando que el brazo robótico responda correctamente sin interrupciones, bloqueos del software o necesidad de reinicios.	RF7, RF8, RF12

5.2. Arquitectura de software

La arquitectura de *software* cumple un rol fundamental en este proyecto, ya que define la estructura general del sistema, la forma en que sus componentes se organizan y cómo interactúan entre sí. Seguir una arquitectura correctamente definida permite asegurar atributos de calidad del *software*, como la modularidad, mantenibilidad, escalabilidad y confiabilidad, además de aspectos específicos como la comunicación inalámbrica y el control del *hardware*.

En este proyecto, la arquitectura adoptada separa claramente las responsabilidades entre la interfaz de usuario, la gestión de la comunicación de datos y el control físico del brazo robótico. Esta separación facilita la detección y corrección de errores, la incorporación de mejoras futuras y la adaptación del sistema ante posibles cambios de *hardware* o *software*, sin afectar la funcionalidad global.

La estructura del sistema se compone de los siguientes módulos:

- **Cliente:** Se implementa en una computadora personal e incluye una interfaz gráfica de usuario (GUI), cuyo diseño se asemeja al de un *joystick* virtual. A través de esta interfaz, el usuario controla de forma intuitiva los movimientos del sistema, tales como el desplazamiento lateral del brazo, el movimiento vertical y la apertura o cierre de la pinza. La GUI interpreta las acciones del usuario y las traduce en comandos que son transmitidos por la red para su ejecución.

- **Canal de Comunicación:** Este componente es el mecanismo encargado de la transmisión y recepción de datos entre el cliente y el *hub* Spike Prime. En este proyecto, la comunicación se establece de forma inalámbrica mediante *Bluetooth*, ya que es el medio soportado por el dispositivo. Este módulo gestiona el envío de información a bajo nivel, trabajando con *bytes* o paquetes de datos, asegurando que los comandos generados por el cliente lleguen correctamente al servidor y que las respuestas o estados del sistema puedan ser retornados al cliente. La comunicación es bidireccional, permitiendo tanto el envío de órdenes como la recepción de información de estado.

- **Servidor:** El servidor se ejecuta directamente en el *hub* Spike Prime a través de un programa desarrollado en MicroPython o Pybricks. Su función principal es recibir los datos enviados por el cliente, interpretarlos como comandos de control y coordinar su

ejecución sobre el sistema físico. Este componente actúa como intermediario entre la capa de comunicación de datos y el control directo del robot.

- **Módulo de Control del Robot:** Este módulo es esencial. Se encarga de traducir los comandos abstractos ("OPEN", "UP", "SEQUENCE") en movimientos concretos de los motores del brazo robótico. Controla parámetros como la velocidad, posición y fuerza aplicada, asegurando que los movimientos sean precisos y consistentes con las órdenes del usuario.

Este diseño arquitectónico estratégico permite una clara separación entre la lógica de control y la capa de interacción con el usuario. Dicha separación facilita considerablemente la corrección de errores, la adición de mejoras y la sustitución de componentes específicos sin comprometer la estabilidad del sistema general.

La **Figura 4** Ilustra el flujo de interacción del sistema: el usuario selecciona una opción en el Cliente (Interfaz), el cual envía la instrucción mediante paquetes de datos a través de un canal bidireccional (Bluetooth). Finalmente, el Servidor (Hub) interpreta la señal y acciona el Módulo de Control para mover los motores.

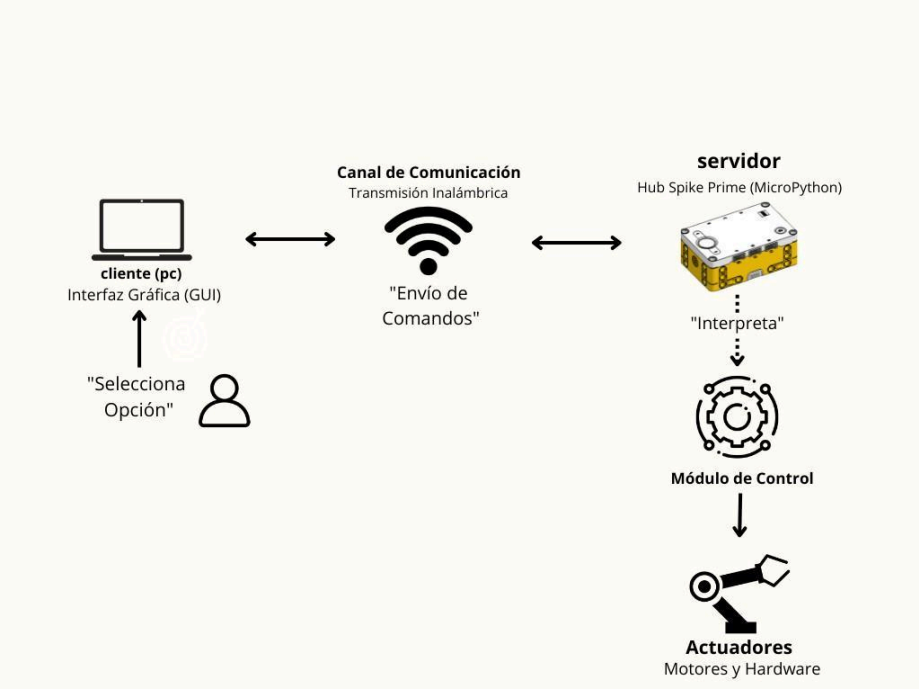

Figura 4: *Diagrama de la Arquitectura de Software Cliente-Servidor.*

5.3. Diseño inicial de la interfaz gráfica de usuario (GUI)

La Interfaz Gráfica de Usuario (GUI) constituye el punto de contacto crítico entre el operador y el sistema robótico, siendo fundamental para la arquitectura Cliente-Servidor (sección 5.2). Para asegurar la operación eficiente del brazo robótico, se requiere que la interfaz cumpla con altos estándares de Usabilidad (RNF 4.1).

Inicialmente, el diseño se aborda mediante un bosquejo (wireframe) de baja fidelidad, un diagrama esquemático que representa la estructura y jerarquía de los elementos de control (botones, indicadores, áreas de texto) sin incluir aspectos estéticos. El objetivo de este paso es planificar la disposición lógica de los controles manuales (RF9, RF10), la visibilidad del feedback (RF12) y la coherencia del flujo de operación antes de proceder a la codificación final.

En la **Figura 5** se muestra el diseño en su fase beta. Este modelo será refinado durante la etapa de implementación, incorporando mejoras basadas en pruebas internas y retroalimentación del equipo.

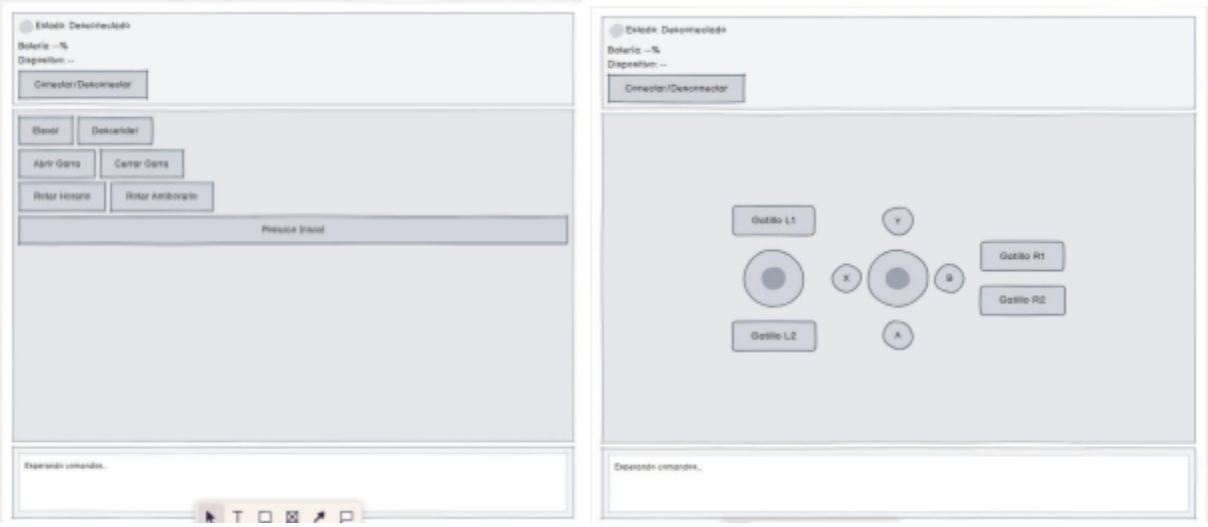

Figura 5: *Fase Beta de la interfaz gráfica de usuario.*

6. Implementación

1. **Justificación Física y Validación del *Hardware***

El diseño mecánico del brazo robótico se fundamenta en principios de movimiento, específicamente el Trabajo Mecánico. Se implementó un sistema de engranajes en la estructura para optimizar la relación velocidad-torque, esencial para la rotación y el

posicionamiento preciso de las articulaciones dentro del área de trabajo.

Los motores están configurados para generar el trabajo mecánico necesario para superar las fuerzas internas (peso del brazo y resistencia de los engranajes) y manipular la carga externa (el bloque). La integración de los engranajes es crucial, ya que facilita la transmisión del movimiento y aumenta el torque disponible en las articulaciones que requieren mayor potencia.

La validación del *hardware* se confirmó mediante pruebas semanales exhaustivas, demostrando que **el brazo fue capaz de realizar** tareas repetitivas y estables de movimiento, levantamiento y agarre. Esto comprueba que el diseño y el trabajo mecánico ejecutado por el Brazo Robótico **fueron** plenamente adecuados para cumplir con los objetivos del proyecto.

6.1. Fundamentos de los movimientos

Esta sección tiene como objetivo justificar la configuración mecánica seleccionada para el robot, aplicando conceptos fundamentales de física estudiados en la asignatura Introducción a la Física. Este análisis se realiza considerando las capacidades reales de torque y velocidad de los motores **LEGO Spike Prime** y las limitaciones estructurales impuestas por el material.

6.1.1. Aplicación del Principio de Velocidad Angular (Justificación del Rendimiento)

Para el motor encargado de la **rotación de 360 grados de la garra** (RF10), se utiliza el concepto de **Velocidad Angular (ω)** para validar el cumplimiento del Requerimiento No Funcional de Rendimiento (RNF 1.2).

La Velocidad Angular se define como la razón de cambio del ángulo (Δθ) con respecto al tiempo (Δt):

$\omega = \Delta\theta/\Delta t$

1. Parámetros de Desempeño Ajustados:

- **Desplazamiento Angular (Δθ):** 360° (2π radianes).
- **Tiempo Objetivo (Δt):** Se establece un tiempo límite realista de Δt = 3 segundos para la rotación (ajustado del RNF 1.2).

2. Cálculo de la Velocidad Angular Requerida:

La velocidad angular mínima que debe alcanzar el motor es:

$$\omega\ requerida = 360°/3\ (s) = 120°/s$$

Convertido a revoluciones por minuto (RPM): ω requerida ≈ 20 RPM.

3. Conclusión de Configuración: El motor LEGO Spike Prime seleccionado tiene una capacidad superior a los 20 RPM. Al operar el motor a una velocidad que alcance este umbral, se garantiza que el requerimiento de 3 segundos para la rotación se cumpla eficientemente, permitiendo además margen para la aceleración y la inercia del brazo.

6.1.2. Aplicación del Principio de Trabajo Mecánico (Justificación de la Capacidad de Carga)

Se utiliza el concepto de **Trabajo Mecánico (W)** para justificar que el motor de elevación puede superar el peso del objeto a manipular (RF3, RF5). El trabajo es el producto de la fuerza aplicada (F) por la distancia recorrida (d):

$$W = F \cdot d$$

1. Parámetros de Carga Ajustados (Centrado en un Bloque):

- **Masa a Levantar (m):** Considerando únicamente un bloque ligero de LEGO (ej. un *brick* de 2×4), la masa estimada es m bloque = 0.003 kg (3 gramos).
- **Altura de Elevación (d):** La altura de elevación segura es d = 0.10 m (10 centímetros).
- **Aceleración de Gravedad (g):** g≈9.8 m/s2.

2. Cálculo de la Fuerza Requerida (Peso del Bloque):

La fuerza de elevación (Peso, P) es:

$$F = m\ bloque \cdot g$$
$$F = [0.003\ (kg)] \cdot [9.8\ (m/s^2)]$$
$$F \approx 0.0294\ (N)$$

3. Cálculo del Trabajo Mecánico Total:

El trabajo (W) que el motor debe realizar para elevar el bloque es:

$$W = F \cdot d$$
$$W = [0.0294\ (N)] \cdot [0.10\ (m)]$$
$$W \approx 0.00294\ Joules\ (J)$$

4. Conclusión de Configuración: El valor del trabajo requerido es extremadamente bajo (0.00294 J). Este resultado confirma que el motor de elevación no tendrá problemas para superar la carga de un solo bloque de LEGO. La principal consideración de diseño no es la

fuerza, sino el **control de la posición y la precisión** para garantizar que el brazo se detenga exactamente en la posición de agarre (RF4) y la posición de depósito (RF3), utilizando los codificadores de los motores Spike Prime para asegurar una precisión angular requerida por el sistema (RNF 3.3).

6.2. Descripción del sistema

El sistema desarrollado constó de dos componentes de *software* principales: el Cliente y el Servidor, en línea con la arquitectura Cliente-Servidor definida en la Sección 5.2. Para facilitar la trazabilidad y la colaboración, el código fuente de ambos programas estuvo centralizado en un repositorio de GitHub.

- https://github.com/joaquinquezadaflores-cell/Proyecto-1/tree/main

6.2.1. Cliente y Servidor

Componente	Plataforma y Lenguaje	Función Principal	Componentes Clave del Código
Cliente	Python (Portátil)	**Control Manual e Interfaz:** Se encarga de recibir las instrucciones del operador y convertirlas en comandos digitales, los cuales son transmitidos de forma inalámbrica.	Los módulos clave son: Interfaz Gráfica de Usuario (GUI), Comunicación y Empaquetamiento de Comandos.
Servidor	MicroPython (Hub Spike Prime)	**Control de Actuadores:** Se encarga de recibir, decodificar y procesar las órdenes para ejecutar movimientos exactos mediante los motores.	Los módulos principales son el de Recepción Inalámbrica, el de Control de Motores (que gestiona las rutinas de rotación y elevación) y el de Reporte de Estado.

En la **Figura 6** se muestra de la línea 11 a la 71, el código **gateway** es el programa que se ejecuta directamente en el hub LEGO SPIKE Prime y actúa como intermediario entre la aplicación de escritorio y los motores del brazo robótico.

Su función principal es recibir comandos por la entrada estándar vía Bluetooth y traducirlos en acciones sobre los motores conectados (base, muñeca, brazo y garra). Cada comando indica el motor a controlar y el sentido de giro, mientras que un comando de parada detiene todos los motores.

Además, el gateway inicializa el hub, configura los motores, resetea los ángulos y aplica límites de movimiento en la muñeca para evitar sobrepasar rangos seguros. Este enfoque permite un control continuo y en tiempo real, manteniendo la lógica de hardware aislada de la interfaz gráfica.

```
CODIGO_GATEWAY_HUB = """
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor
from pybricks.parameters import Port, Direction
from pybricks.tools import wait
import uselect, usys

hub = PrimeHub()

motor_base = Motor(Port.D, Direction.CLOCKWISE)
motor_muneca = Motor(Port.F, Direction.CLOCKWISE)
motor_brazo = Motor(Port.B, Direction.CLOCKWISE)
motor_garra = Motor(Port.C, Direction.CLOCKWISE)

motor_base.reset_angle(0)
motor_muneca.reset_angle(0)
motor_brazo.reset_angle(0)
motor_garra.reset_angle(0)

poll = uselect.poll()
poll.register(usys.stdin, uselect.POLLIN)

hub.display.char('A')

while True:
    if poll.poll(10):
        comando = usys.stdin.read(2)

        # BASE
        if comando == 'D+':
            motor_base.run(100)
        elif comando == 'D-':
            motor_base.run(-100)

        # MUÑECA (límites)
        elif comando == 'F+' and motor_muneca.angle() < 85:
            motor_muneca.run(35)
        elif comando == 'F-' and motor_muneca.angle() > -185:
            motor_muneca.run(-35)

        # BRAZO
        elif comando == 'B+':
            motor_brazo.run(25)
        elif comando == 'B-':
            motor_brazo.run(-25)

        # GARRA
        elif comando == 'C+':
            motor_garra.run(35)
        elif comando == 'C-':
            motor_garra.run(-35)

        # STOP GENERAL
        elif comando == 'ST':
            motor_base.stop()
            motor_muneca.stop()
            motor_brazo.stop()
            motor_garra.stop()

    wait(20)
"""
```

Figura 6: *Código Gateway.*

En la **Figura 7** se muestra de la línea 73 a la 140 la clase **Worker BLE** es el módulo que gestiona la comunicación Bluetooth Low Energy entre la interfaz gráfica y el hub LEGO SPIKE Prime. Se ejecuta en un hilo independiente con *asyncio*, evitando que la comunicación BLE bloquee la interfaz.

Su función es conectarse al hub, instalar un programa gateway y enviar comandos en tiempo real para controlar los motores del brazo robótico. Además, maneja la conexión, desconexión y el envío de estados a la interfaz, garantizando una comunicación estable y fluida.

```
73  class WorkerBLE:
74      def __init__(self, cola_logs: Queue):
75          self.bucle = asyncio.new_event_loop()
76          self.hilo = threading.Thread(target=self._hilo_principal, daemon=True)
77          self.cola_comandos = None
78          self.hub = None
79          self.en_ejecucion = threading.Event()
80          self.cola_logs = cola_logs
81          self.dispositivo_objetivo = None
82          self.solicitud_conexion = asyncio.Event()
83
84      def log(self, mensaje: str):
85          self.cola_logs.put(mensaje)
86
87      def iniciar(self):
88          if not self.hilo.is_alive():
89              self.hilo.start()
90
91      def _hilo_principal(self):
92          asyncio.set_event_loop(self.bucle)
93          self.bucle.create_task(self._ejecutor())
94          self.bucle.run_forever()
95
96      async def _ejecutor(self):
97          ruta_temp = None
98          while True:
99              await self.solicitud_conexion.wait()
100             try:
101                 self.log("Conectando al hub...")
102                 self.hub = PybricksHubBLE(self.dispositivo_objetivo)
103                 await self.hub.connect()
104
105                 with tempfile.NamedTemporaryFile(
106                     mode="w", suffix=".py", delete=False, encoding="utf-8"
107                 ) as archivo:
108                     archivo.write(CODIGO_GATEWAY_HUB)
109                     ruta_temp = archivo.name
110
111                 self.log("Instalando controlador en el hub...")
112                 self.cola_comandos = asyncio.Queue()
113                 asyncio.create_task(self.hub.run(ruta_temp))
114
115                 self.en_ejecucion.set()
116                 self.log("LISTO - Control habilitado")
117
118                 while self.en_ejecucion.is_set():
119                     comando = await self.cola_comandos.get()
120                     await self.hub.write(comando.encode())
121
122             except Exception as error:
123                 self.log(f"Error BLE: {error}")
124
125             finally:
126                 if ruta_temp and os.path.exists(ruta_temp):
127                     os.unlink(ruta_temp)
128                 if self.hub:
129                     await self.hub.disconnect()
130                 self.en_ejecucion.clear()
131                 self.solicitud_conexion.clear()
132                 self.log("Hub desconectado")
133
134     def conectar(self, dispositivo):
135         self.dispositivo_objetivo = dispositivo
136         self.bucle.call_soon_threadsafe(self.solicitud_conexion.set)
137
138     def enviar(self, comando: str):
139         if self.en_ejecucion.is_set() and self.cola_comandos:
140             self.bucle.call_soon_threadsafe(self.cola_comandos.put_nowait, comando)
```

Figura 7: *Clase WorkerBLE.*

En la **Figura 8** se muestran las líneas 142 a la 186, La **clase VentanaSeleccionDispositivo** se encarga de buscar y mostrar los dispositivos Bluetooth disponibles para que el usuario seleccione el hub LEGO a conectar. Esta ventana se presenta como un cuadro modal independiente de la interfaz principal.

Internamente, realiza el escaneo BLE en un hilo separado, utilizando Bleak Scanner, para no bloquear la interfaz gráfica. Los dispositivos encontrados se listan dinámicamente como botones, y al seleccionar uno, se envía el dispositivo elegido al sistema de conexión y la ventana se cierra.

```
142  class VentanaSeleccionDispositivo(ctk.CTkToplevel):
143      def __init__(self, padre, callback_seleccion):
144          super().__init__(padre)
145          self.callback_seleccion = callback_seleccion
146
147          self.title("Buscar Hub LEGO")
148          self.geometry("400x450")
149          self.attributes("-topmost", True)
150          self.grab_set()
151
152          ctk.CTkLabel(
153              self,
154              text="Dispositivos encontrados",
155              font=("Arial", 14, "bold")
156          ).pack(pady=10)
157
158          self.frame_scroll = ctk.CTkScrollableFrame(self, width=350, height=300)
159          self.frame_scroll.pack(expand=True, fill="both", padx=10)
160
161          ctk.CTkButton(self, text="Escanear", command=self.escanear).pack(pady=10)
162          self.escanear()
163
164      def escanear(self):
165          for w in self.frame_scroll.winfo_children():
166              w.destroy()
167          threading.Thread(target=self._hilo_escaner, daemon=True).start()
168
169      def _hilo_escaner(self):
170          bucle = asyncio.new_event_loop()
171          dispositivos = bucle.run_until_complete(BleakScanner.discover(timeout=3))
172          bucle.close()
173          self.after(0, lambda: self._actualizar(dispositivos))
174
175      def _actualizar(self, dispositivos):
176          for dispositivo in dispositivos:
177              if dispositivo.name:
178                  ctk.CTkButton(
179                      self.frame_scroll,
180                      text=dispositivo.name,
181                      command=lambda d=dispositivo: self._seleccionar(d),
182                  ).pack(fill="x", padx=10, pady=5)
183
184      def _seleccionar(self, dispositivo):
185          self.callback_seleccion(dispositivo)
186          self.destroy()
```

Figura 8: *Clase VentanaSeleccionDispositivo.*

En la **Figura 9** se muestran las líneas 188 a la 274, La clase **InterfazBrazo** implementa la interfaz gráfica principal del sistema de control del brazo robótico LEGO. Su función es permitir al usuario conectarse al hub, enviar comandos de movimiento y visualizar el estado del sistema.

Esta clase coordina la interacción entre la interfaz y el Worker BLE, generando comandos a partir de botones de control, mostrando mensajes de estado y manteniendo una experiencia de uso clara y responsiva.

```
188  class InterfazBrazo:
189      def __init__(self, raiz):
190          self.raiz = raiz
191          self.raiz.title("Control Brazo LEGO")
192          self.raiz.geometry("720x720")
193
194          self.cola_logs = Queue()
195          self.trabajador = WorkerBLE(self.cola_logs)
196          self.trabajador.iniciar()
197
198          self._construir_ui()
199          self._procesar_logs()
200
201      def _construir_ui(self):
202          barra_superior = ctk.CTkFrame(self.raiz)
203          barra_superior.pack(fill="x", padx=20, pady=15)
204
205          ctk.CTkButton(
206              barra_superior,
207              text="BUSCAR HUB",
208              command=self.abrir_selector
209          ).pack(side="left", padx=10)
210
211          self.estado = ctk.CTkLabel(
212              barra_superior,
213              text="● DESCONECTADO",
214              text_color="red"
215          )
216          self.estado.pack(side="right", padx=15)
217
218          # ====== FRAME CENTRAL ======
219          frame_control = ctk.CTkFrame(self.raiz)
220          frame_control.pack(expand=True)
221
222          for i in range(3):
223              frame_control.columnconfigure(i, weight=1)
224              frame_control.rowconfigure(i, weight=1)
225
226          self._boton(frame_control, "BASE ◀", "D-", 0, 0)
227          self._boton(frame_control, "BASE ▶", "D+", 0, 2)
228
229          self._boton(frame_control, "GARRA +", "C+", 1, 0)
230          self._boton(frame_control, "MUÑECA ▲", "F+", 1, 1)
231          self._boton(frame_control, "BRAZO ▲", "B+", 1, 2)
232
233          self._boton(frame_control, "GARRA -", "C-", 2, 0)
234          self._boton(frame_control, "MUÑECA ▼", "F-", 2, 1)
235          self._boton(frame_control, "BRAZO ▼", "B-", 2, 2)
236
237          self.texto_logs = ctk.CTkTextbox(self.raiz, height=160)
238          self.texto_logs.pack(fill="both", padx=20, pady=15)
239          self.texto_logs.configure(state="disabled")
240
241      def _boton(self, padre, texto, comando, fila, columna):
242          boton = ctk.CTkButton(
243              padre,
244              text=texto,
245              width=160,
246              height=60
247          )
248          boton.grid(row=fila, column=columna, padx=30, pady=20)
249
250          boton.bind("<ButtonPress-1>", lambda e: self.trabajador.enviar(comando))
251          boton.bind("<ButtonRelease-1>", lambda e: self.trabajador.enviar("ST"))
252
253
254      def abrir_selector(self):
255          VentanaSeleccionDispositivo(self.raiz, self.trabajador.conectar)
256
257      def _procesar_logs(self):
258          try:
259              while True:
260                  mensaje = self.cola_logs.get_nowait()
261                  self.texto_logs.configure(state="normal")
262                  self.texto_logs.insert("end", f"> {mensaje}\n")
263                  self.texto_logs.see("end")
264                  self.texto_logs.configure(state="disabled")
265
266                  if "LISTO" in mensaje:
267                      self.estado.configure(text="● CONECTADO", text_color="green")
268                  if "desconectado" in mensaje.lower():
269                      self.estado.configure(text="● DESCONECTADO", text_color="red")
270
271          except Empty:
272              pass
273
274          self.raiz.after(100, self._procesar_logs)
```

Figura 9: *Clase InterfazBrazo.*

En la **Figura 10**, el **main** es el punto de entrada de la aplicación. Su función es inicializar la configuración visual, crear la ventana principal y lanzar la interfaz gráfica del sistema.
En este caso, se establece el modo oscuro de CustomTkinter, se instancia la clase **InterfazBrazo** y se inicia el bucle principal de la aplicación, dejando el control total en manos de la interfaz y los módulos asociados.

```
276     if __name__ == "__main__":
277         ctk.set_appearance_mode("dark")
278         raiz = ctk.CTk()
279         InterfazBrazo(raiz)
280         raiz.mainloop()
```

Figura 10: *Main.*

6.2.2. Interfaz gráfica de usuario (GUI)

En la **Figura 11**, se muestra la Interfaz Gráfica de Usuario (GUI) del sistema que fue desarrollada como un módulo independiente que se ejecuta en un computador. Su función es establecer una comunicación inalámbrica, utilizando Bluetooth Low Energy (BLE), con el Hub LEGO SPIKE Prime. Esta interfaz es esencial para la interacción, ya que permite al usuario enviar comandos de control al brazo robótico sin requerir manipulación física directa del robot.

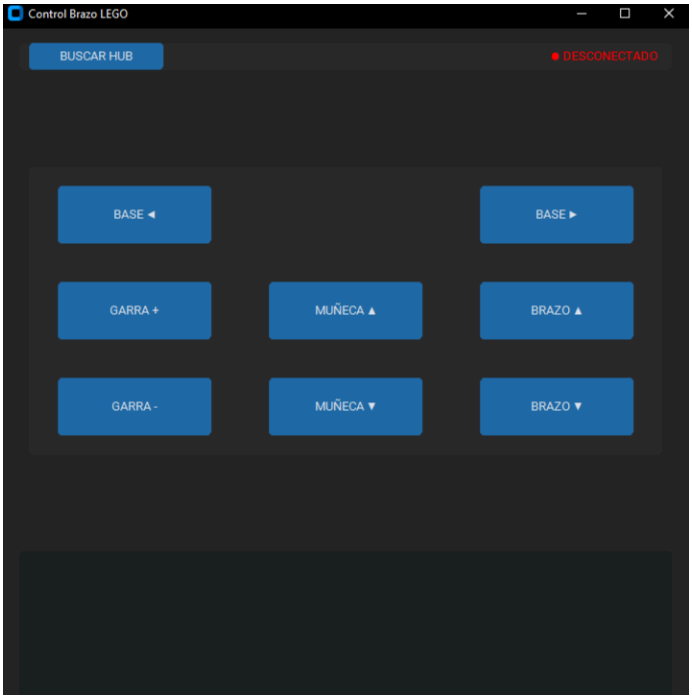

Figura 11: *Imagen de la Interfaz Gráfica de Usuario.*

Se ha completado la implementación de la interfaz gráfica de usuario (GUI), superando exitosamente todas las pruebas de validación funcional. Los resultados confirman una comunicación estable y precisa, garantizando que el robot ejecute correctamente cada comando enviado desde la interfaz.

7. Resultados

7.1. Estado actual del proyecto

El estado actual del proyecto corresponde a un sistema completamente funcional tanto en sus componentes mecánicos como en la lógica de control y la interfaz de usuario del brazo robótico. Durante las pruebas realizadas, el sistema presentó un comportamiento estable y consistente, permitiendo ejecutar correctamente todas las acciones para las cuales fue diseñado.

En relación con los requisitos funcionales, se verifica el cumplimiento de los RF1, RF2, RF3, RF4, RF5, RF6, RF7, RF8, RF9, RF10 y RF11. En particular, el brazo robótico es capaz de abrir y cerrar la garra, elevar y descender el brazo hasta la posición de agarre, rotar 360° sobre su eje principal y comenzar su funcionamiento desde una posición inicial definida. Asimismo, el sistema logra manipular bloques de forma segura, sosteniendo y trasladándose sin presentar inconvenientes mecánicos. La lógica de control permite actuar correctamente sobre los motores en función de los comandos definidos.

Adicionalmente, se ha desarrollado una interfaz gráfica de usuario que permite el envío de comandos al sistema y que actualmente se encuentra en fase de pruebas y validación funcional. La interfaz permite interactuar con el robot mediante botones de control y visualizar el estado de la conexión cliente–servidor. Estas funcionalidades se integran a la arquitectura Cliente–Servidor propuesta, facilitando la interacción entre el usuario y el sistema y contribuyendo a mejorar la experiencia de uso. De este modo, el proyecto alcanza un nivel de funcionamiento coherente con los objetivos planteados, manteniendo aspectos aún en proceso de validación.

7.2 Problemas encontrados y solucionados

Durante el desarrollo del proyecto se presentaron diversos inconvenientes técnicos y organizativos propios del proceso de construcción y programación del brazo robótico. Estas dificultades permitieron al equipo identificar oportunidades de mejora y realizar los ajustes necesarios para asegurar el correcto funcionamiento del sistema.

Uno de los principales problemas estuvo relacionado con la disponibilidad limitada de piezas LEGO, lo que obligó a modificar el diseño original de algunos mecanismos. Esta situación se soluciona mediante el uso de un kit de extensión, lo que permitió construir una estructura estable, funcional y acorde a los requerimientos del proyecto.

Asimismo, se presentaron dificultades en la comunicación inalámbrica entre la interfaz gráfica y el hub de SPIKE Prime, provocando retrasos en la ejecución de algunos comandos. Para resolver este inconveniente, se optimizó el proceso de envío y ejecución de comandos, logrando una comunicación más eficiente y una mejor respuesta del sistema.

Finalmente, debido al requerimiento de implementar un sistema de control mediante una interfaz gráfica, fue necesario diseñar y desarrollar una interfaz con funcionamiento tipo joystick virtual. Este proceso implicó un período de adaptación y pruebas, pero permitió obtener un control más intuitivo y alineado con los objetivos del proyecto.

8. Prueba de funcionamiento del sistema

8.1 Descripción de la prueba de funcionamiento

La prueba de funcionamiento del sistema corresponde a una validación funcional mínima del brazo robótico de agarre, basada en la configuración seleccionada para el proyecto. El objetivo de esta prueba es verificar que el sistema completo incluyendo la comunicación Cliente–Servidor, la interfaz gráfica de usuario (GUI) y el control físico del robot opere de manera correcta y coordinada.

La prueba se realiza con el brazo robótico ensamblado, conectado al Hub LEGO SPIKE Prime y enlazado de forma inalámbrica con un computador que ejecuta la interfaz gráfica. El sistema inicia desde una posición inicial definida y con un bloque ubicado en la zona de agarre.

Durante la prueba, el robot debe ejecutar la siguiente secuencia de acciones:

1. Establecer conexión entre la GUI (cliente) y el Hub SPIKE Prime (servidor).
2. Recibir comandos desde la interfaz gráfica.
3. Abrir la garra y posicionarse sobre el bloque.
4. Cerrar la garra para sujetar el bloque de forma segura.
5. Elevar el brazo hasta una posición de transporte.
6. Rotar el brazo hacia el punto de descarga.
7. Descender el brazo y liberar el bloque.
8. Retornar el brazo a la posición inicial.

El cumplimiento de esta secuencia valida los requerimientos funcionales asociados al control manual del sistema y a la correcta manipulación de la carga.

8.2 Resultados observados para la prueba de funcionamiento

Al momento de la elaboración de este informe, el sistema se encuentra en condiciones operativas para la ejecución de la prueba de funcionamiento definida en la sección anterior. Las pruebas internas realizadas de manera parcial han permitido verificar el correcto funcionamiento de los movimientos básicos del brazo robótico, tales como la apertura y cierre de la garra, la elevación y descenso del brazo, la rotación sobre su eje y la manipulación segura de bloques.

Asimismo, se ha comprobado la correcta transmisión de comandos entre el cliente y el servidor

mediante comunicación inalámbrica, permitiendo el control del sistema desde una interfaz externa.

La demostración de la prueba de funcionamiento del sistema será presentada durante la instancia final del proyecto mediante material audiovisual, el cual documenta la ejecución del brazo robótico bajo las condiciones definidas para la prueba. En dicho material se evidencia el funcionamiento continuo del robot durante la secuencia completa, el uso de la interfaz gráfica de usuario para su operación y el envío de comandos desde la GUI hacia el sistema.

A través de esta demostración registrada, se confirma la correcta ejecución de la secuencia definida y el cumplimiento de los principales objetivos funcionales del proyecto, permitiendo validar de forma práctica el desempeño del sistema en condiciones controladas.

9.Conclusión

El proyecto de construcción y programación de un brazo robótico de agarre con LEGO SPIKE Prime ha cumplido con sus objetivos principales, demostrando la viabilidad técnica del sistema y su lógica de control. Se lograron avances significativos en el diseño mecánico y la implementación de funcionalidades clave, incluyendo la apertura y cierre de la garra, el control de elevación y rotación, y la manipulación segura de bloques. Un hito importante fue la implementación exitosa de un sistema de control externo basado en código Python, validando la comunicación inalámbrica y la capacidad de operación remota.

A lo largo del desarrollo, el equipo enfrentó desafíos técnicos y organizacionales. En el ámbito técnico, los problemas más relevantes incluyeron la dificultad inicial para establecer la comunicación Bluetooth con una interfaz gráfica de usuario, la imprecisión en los movimientos del brazo y problemas persistentes de conexión al migrar a la programación en Visual Studio Code. A nivel organizacional, se presentaron dificultades de coordinación y sincronización grupal.

Para superar estos obstáculos, se implementaron soluciones efectivas. Se ajustó el entorno de desarrollo a una versión compatible de Python (3.8.10) con la librería *Bleak* para habilitar la comunicación Bluetooth bajo Pybricks. La imprecisión de los movimientos se mitigó ajustando los parámetros de velocidad de los motores para priorizar la estabilidad. Los problemas de conexión en Visual Studio Code se resolvieron mediante la correcta instalación y configuración de extensiones. Finalmente, la coordinación del equipo mejoró a través de sesiones de trabajo grupales sincronizadas en plataformas en línea.

A pesar de los logros, el proyecto aún presenta oportunidades de mejora. Las tareas pendientes más relevantes para el futuro próximo incluyen la validación final y completa de la interfaz gráfica

de usuario (GUI), la optimización de la precisión y suavidad de los movimientos del brazo, la incorporación de mecanismos de manejo de errores y tolerancia a fallos en la comunicación, y la optimización general del código. Asimismo, la integración con el vehículo autónomo desarrollado por otros grupos es una etapa clave para completar el proyecto colaborativo de transporte. En conclusión, el proyecto establece una base sólida y funcional, con un claro camino a seguir para aumentar su robustez, usabilidad y alcance.

10.Referencias

1. Lego Brick & Tech. "Building a 360-Degree Rotating LEGO Robotic Arm | SpikePrime 【Tutorial】." YouTube. [Online]. Disponible en: https://www.youtube.com/watch?v=fAM7FBOOy74 (3 de enero de 2025).

2. LEGO Education. "LEGO Education SPIKE Prime." [Online]. Disponible en: https://education.lego.com/es-es/product/spike-prime.

3. Pybricks. "Pybricks Code." [Online]. Disponible en: https://code.pybricks.com.

4. Universidad de Tarapacá (UTA). "Plataforma Redmine UTA." [Online]. Disponible en: https://redmine.uta.cl.

5. IREA 국제로봇교육협의회. "How to connect Remote controller with Spike prime." YouTube. [Online]. Disponible en: https://www.youtube.com/watch?v=jdy6h0xA6lg (15 de junio de 2023).

6. *Balsamiq: Fast, focused wireframing tools*. (s. f.). https://balsamiq.com/