UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA
EN COMPUTACIÓN E INFORMÁTICA

Plan de proyecto: Fase 3

Brazo Robótico de Lego

Autores: Eduardo Suaña
Dylan Calderón
Matías Agriano
Benjamín Sucso

Asignatura: Proyecto I
Profesor: Baris Nikolai Klobertanz Quiroz

DICIEMBRE - 2025
ARICA - CHILE

Historial de cambios

Tabla 1

Asignación de roles

Fecha	Versión	Descripción	Autor(es)
09/10/2025	1.0	Creación del documento base del proyecto.	Matías Agriano
09/10/2025	1.1	Agregado de introducción, objetivos y especificación del problema. Inclusión de roles del equipo y métodos de comunicación.	Matías Agriano Eduardo Suaña Benjamín Sucso
10/10/2025	1.2	Desarrollo de la planificación de actividades y Carta Gantt.	Dylan Calderón
11/10/2025	1.3	Incorporación de la gestión de riesgos y planificación de recursos.	Dylan Calderón Benjamín Sucso
13/10/2025	1.4	Redacción del apartado de hardware y software utilizados.	Eduardo Suaña
16/10/2025	1.5	Agregado de costos estimados.	Benjamín Sucso
17/10/2025	1.6	Redacción de la conclusión y revisión general del informe.	Dylan Calderón Matías Agriano Benjamín Sucso
17/10/2025	1.7	Correcciones en referencias, precios y riesgos	Eduardo Suaña Matías Agriano Benjamín Sucso
24/11/2025	1.8	Corrección completa del informe inicial.	Todos
11/12/25 - 16/12/2025	2.0	Redacción del informe 2.	Todos
24/12/25 - 25/12/25	2.1	Corrección del informe 2.	Todos
27/12/25	3.0	Inclusión de pruebas de funcionamiento, trazabilidad de código y cierre del proyecto.	Todos

Índice

Índice de tablas

Índice de figuras

Introducción

El presente informe detalla la fase final del proyecto “Brazo Robótico de Lego", desarrollado por el equipo conformado por Dylan Calderón, Benjamín Sucso, Matías Agriano y Eduardo Suaña.

A continuación, se presenta la estructura del documento: inicialmente se aborda la problemática de seguridad en la minería y los objetivos del proyecto. Posteriormente, se detalla la planificación (Carta Gantt) y la gestión de riesgos. El núcleo del informe describe la ingeniería de la solución, justificando la física de los movimientos y detallando la arquitectura de software Cliente-Servidor implementada. Finalmente, se exponen las pruebas funcionales, los resultados obtenidos y las conclusiones del trabajo.

1. Planteamiento del problema y objetivo

1.1. Problema

El problema identificado se relaciona con la necesidad de reducir la exposición de los trabajadores a condiciones peligrosas durante las jornadas mineras subterráneas.

- En los procesos de carga y manipulación de material, los operarios enfrentan riesgos físicos y ambientales que pueden afectar su seguridad y salud.

- Se requiere una herramienta robótica que incorpore tecnologías como la teleoperación, la automatización y la robótica para mejorar la productividad, reducir costos, optimizar procesos y, principalmente, minimizar los riesgos humanos asociados a las labores mineras.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar, construir y programar un brazo robótico con LEGO Spike Prime, controlado a distancia mediante una interfaz de teleoperación, para manipular de forma segura y eficiente material fragmentado en un entorno minero simulado, reduciendo la exposición de los trabajadores a riesgos operacionales.

1.2.2. Objetivos específicos

1. Seleccionar y construir el diseño del robot con las piezas del Set de Lego Spike Prime, para tener una base resistente y simétrica.

2. Investigar y hacer pruebas para el desarrollo del software del robot, con el fin de crear un software final estable y eficiente para que el robot cumpla sus tareas.

3. Analizar detalladamente el funcionamiento del Set de Lego Spike Prime, para comprender las capacidades y limitaciones de sus componentes.

4. Programar la interfaz del control remoto, para controlar las funciones del robot mediante conexión remota.

5. Diseñar una interfaz que represente un control remoto para el manejo del robot

1.3. Restricciones

Para el desarrollo de este proyecto, se consideraron las siguientes limitaciones técnicas y operativas:

- **Hardware limitado:** Se estableció una restricción técnica autoimpuesta de utilizar exclusivamente las piezas contenidas en los sets LEGO Education SPIKE Prime y Expansión. Esta decisión busca maximizar la eficiencia de recursos y garantizar la total replicabilidad del sistema, descartando el uso de componentes de terceros o manufactura aditiva (impresión 3D).
- **Presupuesto de Software:** Se estableció un costo de licenciamiento de $0 CLP, utilizando exclusivamente tecnologías de Código Abierto (Open Source). Esto se decidió para eliminar costos operativos recurrentes y facilitar la libre distribución del software.
- **Plazo de ejecución:** El prototipo funcional, junto con su documentación y pruebas, debe ser entregado el 30 de diciembre de 2025.
- **Conectividad:** La comunicación entre el puesto de control (PC) y el robot debe ser totalmente inalámbrica vía Bluetooth Low Energy (BLE), asegurando un rango de operación efectivo superior a 2 metros. Esto es imperativo para garantizar la seguridad física del operador al mantenerlo alejado de la zona de riesgo.
- **Trazabilidad y Documentación:** La gestión documental del proyecto (Informes, Bitácoras, Carta Gantt y Presentaciones) se centraliza en la plataforma Redmine para el seguimiento, mientras que el control de versiones del código fuente y los entregables técnicos se administran exclusivamente mediante el repositorio de GitHub, asegurando una separación clara entre gestión y desarrollo.

1.4. Entregables

Al finalizar la Fase 3, el equipo entregará los siguientes productos:

1. **Informe Técnico Final:** Documento PDF detallando la ingeniería, trazabilidad y gestión del proyecto.
2. **Código Fuente y Release:** Repositorio en GitHub con el versionamiento del software (Cliente/Servidor) y la Release v1.0 empaquetada.
3. **Manual de Usuario:** Documentación técnica (archivo README.md) para la instalación y operación del sistema.
4. **Presentación Ejecutiva:** Archivo PPT para la defensa oral del proyecto.
5. **Demostración Funcional:** Video de la operación de carga y descarga.

2. Organización personal

2.1. Descripción de los roles

Para garantizar un trabajo eficiente y colaborativo, el equipo identificó las actividades principales (diseño y construcción, programación, documentación y gestión) y asignó roles específicos aprovechando las fortalezas individuales.

Tabla 2

Definición de roles

Rol	Responsabilidades Clave
Jefe de proyecto	Coordina el equipo, establece metas y plazos. Asegura que los miembros cumplan sus tareas y comunica los avances.
Constructor	Construye el robot o estructura con las piezas de Lego Spike. Asegura que el diseño sea funcional y estable, proponiendo modificaciones para mejorarlo
Diseñador	Crea y desarrolla el diseño visual y estructural del robot, priorizando la funcionalidad y la estética. Planifica la distribución de piezas para optimizar el equilibrio y el rendimiento.
Programador	Programa el comportamiento del robot usando el Software de LEGO Spike (MicroPython). Depura y ajusta el código para lograr la funcionalidad deseada.
Documentador	Documenta todo el proceso (pasos, problemas, soluciones y resultados). Prepara informes, la carta Gantt, la bitácora del proyecto y presentaciones.

2.2. Asignación de roles

La siguiente tabla detalla el integrante asignado a cada rol dentro del equipo:

Tabla 3

Asignación de roles

Rol	Integrante
Jefe de proyecto	Dylan Calderón
Programador	Benjamín Sucso
Diseñador	Matías Agriano
Constructor	Dylan Calderón
Documentador	Eduardo Suaña

2.3. Canales de comunicación

El medio de comunicación principal utilizado por el equipo es Whatsapp.

- Es una herramienta para la comunicación rápida y constante entre los integrantes del equipo.
- A través de este medio se comparten recordatorios, fotos del proceso, dudas, archivos y actualizaciones del trabajo.
- Su uso permite mantenerse conectado en todo momento, incluso fuera del horario de clases.

3. Planificación del proyecto

Las siguientes actividades fueron registradas en el proyecto, detallando los objetivos específicos a los que contribuyen y sus responsables:

3.1. Actividades definidas

Las siguientes actividades fueron registradas para el proyecto, detallando el nombre de la actividad y el o los responsables de llevarla a cabo:

Tabla 4

Definición de actividades y responsables de acuerdo a los roles asignados

Nombre de la actividad	Responsables
Formar el equipo de trabajo	Todos
Analizar la problemática	Todos
Elaborar la bitácoras semanales	Eduardo Suaña
Redactar la Carta Gantt	Eduardo Suaña Dylan Calderón
Elegir diseño del robot	Matías Agriano
Probar los elementos del set	Dylan Calderón
Construir estructura base	Dylan Calderón
Construir brazo del robot	Dylan Calderón
Construir garra del robot	Dylan Calderón
Probar giro de 360°	Benjamín Sucso
Probar movimiento vertical del brazo robótico	Benjamín Sucso
Probar el agarre y liberación de la garra	Benjamín Sucso
Redactar el Informe 1	Eduardo Suaña Dylan Calderón
Preparar presentación oral 1	Todos
Investigar el software y los métodos de programación	Benjamín Sucso
Programar Interfaz gráfica	Benjamín Sucso Matías Agriano
Redactar informe 2	Eduardo Suaña Dylan Calderón
Preparar presentación oral 2	Todos
Demostrar el producto final	Todos
Elaborar Manual de Usuario	Dylan Calderón
Redactar informe 3	Dylan Calderón

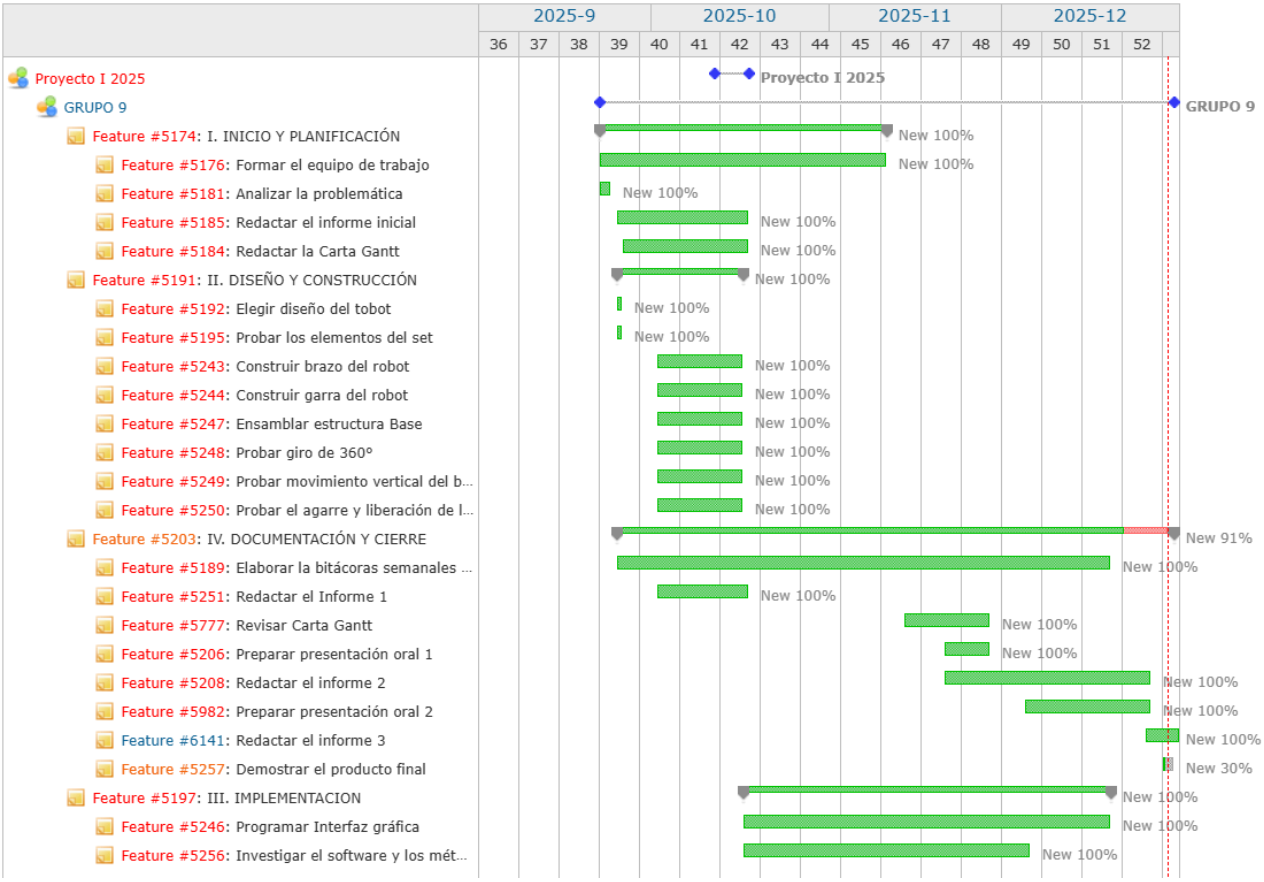
Preparar Defensa Final	Todos

3.2. Carta gantt

Para organizar de manera eficiente las actividades del proyecto, se elaboró una Carta Gantt. Esta herramienta permite visualizar la secuencia de tareas y su distribución en el tiempo. La Carta Gantt facilita la planificación general del trabajo, ya que muestra cuándo debe iniciarse cada actividad, cuánto durará y cómo se relaciona con las demás etapas del proyecto. Gracias a esta representación temporal, el equipo puede observar el progreso de manera gráfica.

Figura 1

Carta gantt



3.3. Gestión de riesgos

Se identificaron riesgos que podrían afectar el desarrollo del trabajo. Estos fueron categorizados según su nivel de impacto (Bajo, Medio y Alto) para determinar las medidas preventivas o de solución.

- **Bajo:** El riesgo puede generar retrasos o dificultades menores que no afectan significativamente el avance del proyecto; su solución requiere poca intervención.

- **Medio:** El riesgo puede afectar parcialmente el rendimiento del equipo o el cumplimiento del cronograma, requiriendo reestructuración moderada para ser solucionado.

- **Alto:** El riesgo puede comprometer gravemente la continuidad del proyecto, retrasar de forma considerable su desarrollo o impedir el cumplimiento de los objetivos si no se atiende oportunamente.

Tabla 5

Riesgos identificados

Riesgo identificado	Nivel de impacto	Medida preventiva
Abandono de integrante	Alto	Redefinir roles, ofrecer ayuda mutua para tratar de prevenir.
Mal funcionamiento o desempeño del robot	Alto	Volver a la fase de diseño teniendo en cuenta las complicaciones.
Cambios de actividades en horarios destinados al proyecto	Bajo	Acordar reuniones y horas extraordinarias.
Limitación de piezas del Set de LEGO	Bajo	Planificar bien el diseño antes de construir y adaptar el modelo según las piezas disponibles o solicitar piezas adicionales.

Durante la ejecución del proyecto, se materializó el riesgo de **“Mal funcionamiento del robot”** específicamente relacionado con la caída del brazo por gravedad (Motor B). Este riesgo fue mitigado exitosamente mediante la implementación de la función de freno activo (hold) en el código del servidor, tal como se detalla en la sección de problemas y soluciones.

4. Identificación de los recursos y costos asociados

Para asegurar la correcta ejecución del proyecto, se identificaron y organizaron los recursos, clasificados en hardware, software y recursos humanos.

4.1. Hardware

Se utilizó el kit LEGO Spike Prime para la construcción y control del sistema robótico. Los principales elementos de hardware empleados fueron:

- **Hub LEGO Spike Prime:** Es el cerebro del robot, contiene un microcontrolador que ejecuta el código programado y coordina las acciones de los motores y sensores.
- **Motores grandes y medianos del hub:** Se utilizaron para generar el movimiento del brazo y de la garra.
- **Cables de conexión del hub:** Permiten la comunicación entre el Hub y los diferentes motores y sensores.
- **Estructura de piezas LEGO:** Conformada por vigas, ejes, conectores y engranajes.
- **Computador personal (PC):** Utilizado para programar el robot, cargar el código en el Hub y realizar las pruebas de funcionamiento.
- **Expansión del Set de LEGO Spike Prime.**

Tabla 6

Costos del hardware

Hardware	Cantidad	Precio	Fuente de la cotización
Galaxy Tab S9 FE	1	$579.991 CLP [1]	Falabella (s. f.)
Lenovo IdeaPad 5 15ITL05	1	$799.990 CLP [2]	SmartDeal.cl (2023)
Lego Education SPIKE PRIME Set	1	$555.000 CLP [3]	The LEGO Group (s. f.)
Lenovo V14 G2 ALC	2	$1.598.000 CLP [4]	Opcstore (s. f.)
Lego Set de expansión (45680)	1	$174.000 CLP [5]	The LEGO Group (s. f.)
Total Hardware	-	$3.706.981 CLP	-

4.2. Software

Se optó por utilizar software gratuito para la programación, documentación y comunicación.

- **Entorno de programación LEGO Spike App:** Plataforma oficial de LEGO Spike que permite crear, cargar y ejecutar programas en el Hub del robot, usado para las pruebas iniciales de los elementos del set LEGO.
- **MicroPython:** Lenguaje de programación utilizado para controlar los motores, sensores y la lógica de movimiento.
- **Git y github:** Para llevar un control de versiones, facilitar la colaboración y mantener un historial de cambios.
- **Pybricks:** Firmware instalado en el hub.
- **Visual Studio Code:** Editor con varias extensiones para apoyar la programación en MicroPython.
- **WhatsApp:** Aplicación para comunicación no presencial.
- **Costo de Software Total:** $0 CLP.

Tabla 7

Costo de Software

Software	Precio
LEGO Spike App	$0 CLP
MicroPython [9]	$0 CLP
Git y github [8]	$0 CLP
Pybricks [9]	$0 CLP
Visual Studio Code	$0 CLP
WhatsApp	$0 CLP
Total Software	$0 CLP

4.3. Recursos humanos

Los costos de personal son una estimación basada en el valor hora de mercado para perfiles de ingeniería o técnicos especializados. La contabilización de horas de trabajo se registró desde el 26 de septiembre hasta el 30 de diciembre. Las horas extras fueron valoradas al mismo costo que la hora de trabajo regular.

Tabla 8

Sueldo del personal

Rol del personal	Horas regulares	Horas extras	Sueldo / hora
Jefe de proyecto	58	10	$28.000 CLP
Programador [6]	58	10	$24.000 CLP
Diseñador	58	10	$23.000 CLP
Ensamblador	58	10	$24.000 CLP
Documentador	58	10	$23.000 CLP
Total personal	-	-	$8.296.000 CLP

4.4. Costo total de recursos

El costo total del proyecto se obtiene sumando el costo de hardware, software y personal.

Tabla 9

Presupuesto total del proyecto

Costo total	Precio
Hardware	$3.706.981 CLP
Software	$0 CLP
Personal	$8.296.000 CLP
Total general	$12.002.981 CLP

5. Análisis y diseño

5.1 Especificación de requerimientos

Para el desarrollo del sistema, se identificaron los roles de los stakeholders principales:

Cliente: Compañía minera que requiere la automatización para reducir riesgos (quien financia/solicita).

Usuario: Operador de maquinaria pesada que controlará el brazo robótico a distancia (quien usa el sistema).

5.1.1. Requerimientos funcionales

El sistema debe cumplir con las siguientes funciones operativas:

- RF1 Movimiento rotativo: El brazo robótico debe ser capaz de rotar su base en 360 grados.
- RF2 Movimiento flexible: El brazo robótico debe poder articular el codo para alcanzar un objeto.
- RF3 Manipulación de Carga: La garra debe abrirse y cerrarse para sujetar firmemente un bloque de LEGO estándar sin dejarlo caer durante el traslado.
- RF4 Teleoperación: El sistema debe permitir el control manual de todos los motores a través de una interfaz gráfica en el ordenador.
- RF5 Parada de Emergencia: La interfaz debe contar con una función para detener todos los motores inmediatamente en caso de una situación de riesgo.

5.1.2. Requerimientos no funcionales

El sistema debe cumplir con los siguientes atributos de calidad, verificables mediante métricas específicas:

- Usabilidad: La interfaz gráfica debe permitir que un usuario sin experiencia previa logre conectar el robot y realizar un movimiento básico en menos de 30 segundos. Los controles deben estar etiquetados claramente en español.
- Rendimiento (Latencia): El tiempo de respuesta entre la pulsación de un botón en la interfaz (Cliente) y la reacción del motor del robot (Servidor) debe ser inferior a 200 milisegundos para garantizar una teleoperación fluida.

- Seguridad (Failsafe): En caso de pérdida de conexión Bluetooth, el robot debe detener todos sus motores automáticamente en un lapso no mayor a 1 segundo para evitar daños.

- Disponibilidad: El sistema debe ser capaz de mantener la conexión activa y operativa durante al menos 10 minutos continuos, tiempo estimado para una demostración estándar.

5.2. Arquitectura de software

El sistema se basa en una arquitectura Cliente-Servidor distribuida físicamente, utilizando comunicación inalámbrica asíncrona mediante el protocolo Bluetooth Low Energy (BLE). Esta arquitectura desacopla la lógica de control de usuario (interfaz) de la lógica de actuación física (motores), tal como se describe en la literatura sobre sistemas distribuidos (Garcia, 2025) [7].

A continuación se detallan los componentes y su interacción:

- **Cliente (PC / Ordenador):** Es el componente de alto nivel ejecutado en un entorno de escritorio (Windows/Linux).

  Hardware: Laptop o PC con adaptador Bluetooth.

  Software: Script en Python que ejecuta la interfaz gráfica (Tkinter) y la librería pybricksdev o bleak.

  Responsabilidad: Captura los eventos del usuario (teclado/ratón), codifica las instrucciones en caracteres simples (bytes) y los transmite vía BLE.

- **Comunicación y Middleware:** Se utiliza la librería **bleak** como middleware en el cliente para gestionar la pila de protocolos y abstraer la complejidad de la conexión. Esta interactúa directamente con el Canal de Comunicación, establecido mediante el protocolo inalámbrico Bluetooth Low Energy (BLE), utilizando el servicio UART estándar para la transmisión bidireccional de datos con baja latencia.
- **Servidor (Hub LEGO Spike):** Es el componente embebido que interactúa con el mundo físico.

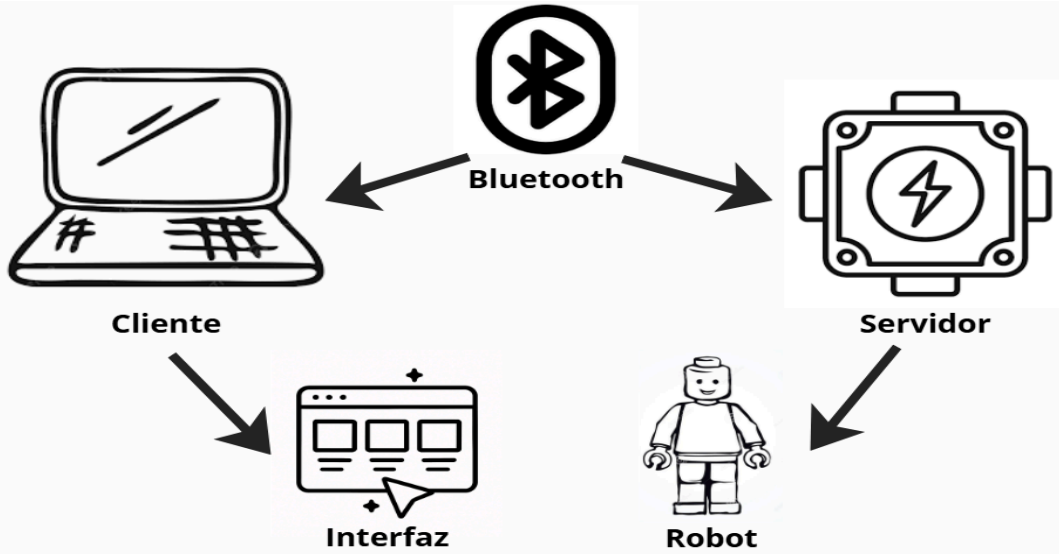
  Hardware: Hub LEGO Spike Prime (Microcontrolador STM32).

  Software: Firmware Pybricks ejecutando un script de MicroPython (main.py).

  Responsabilidad: Ejecuta un bucle infinito (while True) escuchando el puerto serie (stdin). Al recibir un byte, lo decodifica y acciona los puertos de los motores (A, B, C, D) mediante controladores PID internos.

Figura 2

Diagrama arquitectura cliente-servidor del proyecto



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

En la fase de análisis y diseño, se elaboró un boceto o wireframe de baja finalidad de la interfaz gráfica de usuario (GUI). Este diseño preliminar sirve como guía estructural para el componente cliente del sistema, que será utilizado por operador para la teleoperación remota del brazo robótico.

El diseño se centra en la estructura, la disposición lógica y los requerimientos funcionales y no funcionales identificados (especialmente usabilidad y rendimiento), sin incluir detalles de color o estilo visual.

Objetivo del diseño inicial:

El wireframe se diseñó para:

- Asignar un bloque de control dedicado a cada uno de los cuatro motores principales del robot, asegurando un control preciso sobre los ejes de movimiento rotativo(base, brazo, pinza y elevador).

- Incluir botones de conectar y desconectar para gestionar de forma explícita la disponibilidad y la comunicación inicial entre la interfaz y el Hub del robot.

- Proporcionar un control direccional claro (horario / anti horario) para cada motor, permitiendo al operador realizar la teleoperación y manipular el robot con movimiento de avance y retroceso en cada eje.

Componente del wireframe

El boceto de baja finalidad estructura la interfaz en tres componentes lógicos principales, diseñados para cumplir con los requerimientos funcionales del sistema (teleoperación y movimiento).

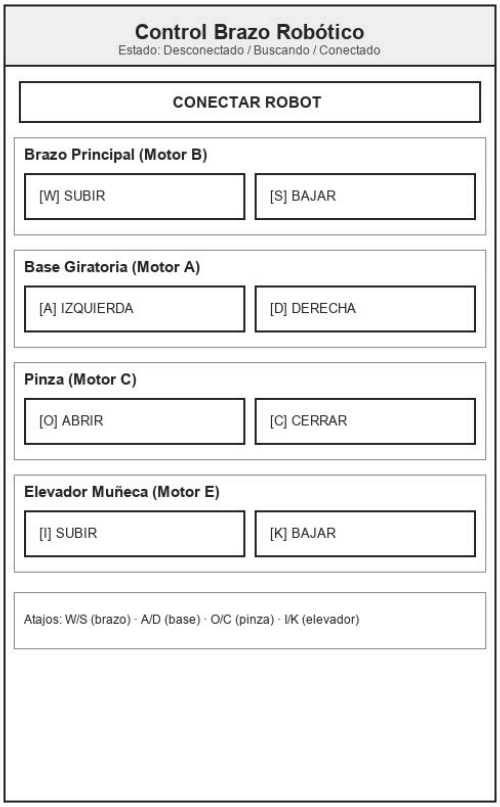
Tabla 10

Desglose de Componentes y función del wireframe de la GUI

Componente	Función del diseño	Requerimiento asociado
Controles de conexión	Gestionan el estado de la comunicación con el Hub del robot, controlando la disponibilidad del sistema.	Disponibilidad
Paneles de control de ejes	Bloques dedicados a cada motor, con botones de horario y antihorario para la teleoperación y el movimiento bidireccional.	Teleoperación y movimiento
Mecanismo de control direccional	La disposición de dos botones por motor permite un control inmediato de la acción en cada eje, lo que es clave para la usabilidad y el rendimiento.	Usabilidad y rendimiento

Figura 3

Wireframe



6. Implementación

Esta sección presenta los resultados obtenidos hasta el momento en el desarrollo del proyecto. Incluye la justificación de la configuración del robot basada en principios físicos fundamentales, una descripción de los componentes clave del sistema implementado (cliente y servidor), y una vista de la Interfaz Gráfica de Usuario (GUI) desarrollada, explicando sus elementos y funciones.

6.1. Fundamentos de los movimientos

Los movimientos del Brazo Robótico de Lego se justifican mediante los principios fundamentales de la física para asegurar que el sistema cumpla con su objetivo de manipular material pesado.

Dada la naturaleza del proyecto, que involucra levantar y mover objetos con un brazo robótico, un principio físico clave a aplicar es el cálculo del trabajo mecánico necesario para elevar la carga. Esto es crucial para determinar la potencia y configuración de los motores requeridos en el brazo y la garra.

El trabajo mecánico (W) realizado por una fuerza (F) para levantar un objeto es igual al cambio en su energía potencial gravitatoria, asumiendo una velocidad constante o una aceleración nula.

$$W = F \cdot d \cdot \cos(\theta)$$

- W es el trabajo mecánico (en Joules, J)
- F es la fuerza aplicada, que debe ser al menos igual al peso del objeto (en Newtons, N)
- θ es el ángulo entre la fuerza y el desplazamiento. En este caso, al levantar verticalmente, θ = 0 grados, por lo que cos(θ) = 1.
- d es la distancia vertical a la que se desplaza al objeto (en metros, m)

$$P = m \cdot g$$

- P Es la fuerza con la que la Tierra atrae al bloque de Lego hacia su centro(en Newtons, N).

- m es la masa que contiene el bloque de Lego (en kilogramos, kg).

- g es la aceleración que experimenta cualquier objeto en la superficie de la Tierra, la gravedad (en metros partidos por segundo al cuadrado, m/s²).

Por lo tanto, el trabajo para levantar un objeto de masa (m) a una altura (h) es:

$$W = m \cdot g \cdot h$$

Para el Brazo Robótico de Lego:

1. **Identificación de la masa (m):** Se debe medir la masa (en kg) del bloque de Lego más pesado que el robot debe manipular.

2. **Identificación de la altura (h):** Se debe determinar la altura máxima (en m) a la que el brazo debe levantar el bloque para cumplir con la tarea de manipulación en el entorno minero simulado.

3. **Cálculo:** Sustituyendo la masa (m) y la altura (h) en la fórmula junto con la aceleración de la gravedad ($g \approx 9{,}8 m/s^2$), se obtiene el trabajo mínimo requerido (en J).

Cálculo con Valores Reales del Proyecto

Se tomaron las mediciones del bloque de material más pesado y la altura máxima de levantamiento requerida para la simulación del entorno minero. El cálculo se aplica al motor principal de levantamiento

Tabla 11

Parámetros físicos

Parámetro	Símbolo	Valor medido	Unidad
Masa del objeto	m	0,15	Kilogramos(Kg)
Aceleración de gravedad	g	9,8	Metros partidos en segundos al cuadrado (m/s^2)
Altura de levantamiento	h	0,20	Metros (m)

Cálculo del peso (P)

Esta fuerza es el peso que el motor B debe contrarrestar para iniciar y mantener la elevación.

Pregunta de análisis: ¿Tienen los motores del kit LEGO Spike Prime la potencia suficiente para levantar una carga de 0.15 kg a una altura de 20 cm sin sufrir daños?

$$P = m \cdot g$$
$$P = 0{,}15\ Kg \cdot 9{,}8\ m/s^2$$
$$P = 1{,}47\ N$$

El motor B debe generar un torque equivalente a una fuerza de 1,47 Newtons en el punto de levantamiento para soportar la carga.

Cálculo del trabajo Mecánico (W)

Este es el trabajo total que el motor B debe realizar para elevar la carga a la altura de 20 cm.

$$W = P \cdot h$$
$$W = 1{,}47\ N \cdot 0{,}20\ m$$
$$W = 0.294\ J$$

El Motor B requiere un Trabajo de 0,294 Joules para completar la tarea de elevación.

Este cálculo del trabajo permite justificar la selección y el uso del Hub LEGO Spike Prime y de los motores grandes y medianos, asegurando que estos componentes tengan la potencia necesaria para realizar la acción de levantamiento y movimiento de la carga.

6.2. Descripción del sistema

La arquitectura del sistema implementado es de tipo Cliente-Servidor y utiliza la comunicación inalámbrica Bluetooth Low Energy (BLE) para la teleoperación del robot. La implementación se concentra en un único script de Python que gestiona la Interfaz de Usuario, la Conexión y el Código del Servidor, el cual se inyecta dinámicamente en el Hub LEGO Spike Prime.

El código fuente documentado y el manual de instalación se encuentran disponibles en el repositorio de GitHub. Para esta entrega final, se ha generado la Release v1.0 estable:

- **Repositorio:** https://github.com/M47355/Proyecto-1.git
- **Release Oficial:** https://github.com/M47355/Proyecto-1/releases/tag/v1.0

El repositorio se estructura en 4 componentes principales: Servidor (MicroPython en el Hub), Interfaz Gráfica (GUI en Tkinter), Cliente (Lógica de control en Python) y Controlador (Manejo de eventos).

Además del código fuente, en el repositorio se incluye el **Manual de Usuario** detallado. Este manual guía al operador a través de tres fases críticas:

1. **Pre-requisitos:** Instrucciones para la instalación de Python y las librerías necesarias (asyncio, bleak, tkinter) en el terminal del PC.
2. **Despliegue:** Procedimiento paso a paso para encender el Hub, ejecutar el script Servidor.py en el robot y lanzar la interfaz Cliente.py en el ordenador.
3. **Operación:** Guía de referencia rápida de los controles (teclas **W/S** para el brazo, **A/D** para la base) y protocolos de seguridad a seguir en caso de desconexión imprevista.

6.2.1 Cliente

El componente Cliente es el script principal de Python, ejecuta los demás componentes (utilizando Tkinter para la GUI y la librería pybricksdev para la conexión asíncrona) para formar la aplicación en el PC del operador.

Figura 4

Código del cliente (componente Cliente)

```python
# Punto de entrada de la aplicación

import tkinter as tk
from interfaz import App


if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

El componente controlador recibe las entradas realizadas por el usuario y las comunica hacia el componente Servidor, esto mediante el componente interfaz

Figura 5

Fragmento código del cliente (componente controlador)

```python
class RobotController:
    """Controla la conexión y comunicación con el Hub LEGO."""

    def __init__(self):
        self.hub = None
        self.loop = asyncio.new_event_loop()
        self.thread = threading.Thread(target=self._run_loop, daemon=True)
        self.thread.start()
        self.connected = False
        self._closing = False

    def _run_loop(self):
        """Ejecuta el loop de asyncio en un hilo separado."""
        asyncio.set_event_loop(self.loop)
        self.loop.run_forever()

    def connect(self, on_success, on_error):
        """Inicia la conexión al Hub en segundo plano."""
        if self._closing:
            on_error("El controlador se está cerrando.")
            return
        asyncio.run_coroutine_threadsafe(
            self._connect_task(on_success, on_error),
            self.loop
        )
```

URL del repositorio de GitHub: https://github.com/M47355/Proyecto-1.git

Tabla 12

Componentes importantes del componente controlador

Componente	Explicación	Función dentro del sistema
class RobotController	Implementa la lógica de concurrencia (asyncio y threading) para manejar la conexión BLE en segundo plano.	Asegura que la GUI permanezca responsiva mientras el sistema busca, conecta e inyecta el programa en el Hub.
_connect_task	Contiene la secuencia de conexión: find_device(name="PY-SC"), conexión (self.hub.connect()), y la inyección remota del código con self.hub.run(temp_path, wait=False).	Establece el vínculo de comunicación y carga el código del Servidor en el robot.
send(char_cmd)	Utiliza self.hub.write(data) para enviar comandos de un solo carácter (ej: 'w', 'a') al Hub.	Es el módulo de comunicación encargado de transmitir las órdenes del operador al robot.
setup_keyboard y _key_press/_key_release	Mapean los eventos de presionar y soltar las teclas del teclado (W, S, A, D, etc.) a los comandos de movimiento y parada/freno, respectivamente.	Permite una teleoperación reactiva y fluida, imitando un control de joystick (al presionar mueve, al soltar frena).

La Figura 4 a continuación, demuestra la implementación del módulo de comunicación del cliente, validando los puntos críticos de la Tabla 10: la conexión asíncrona (_connect_task) y el envío de comandos en tiempo real (send).

Figura 6

Función send del cliente (componente controlador).

```
def send(self, char_cmd):
    if self.connected and self.hub:
        data = bytearray(char_cmd, 'utf-8')
        asyncio.run_coroutine_threadsafe(self.hub.write(data), self.loop)
```

Análisis de la Figura:

La función send(char_cmd) es fundamental para la teleoperación. Su implementación garantiza el rendimiento del sistema, ya que el comando de movimiento se codifica a un único byte (data = bytearray(char_cmd, 'utf-8')) antes de ser enviado por Bluetooth. Esta transmisión mínima de datos es crítica para asegurar la baja latencia requerida en la teleoperación reactiva del brazo. Además, el uso de asyncio.run_coroutine_threadsafe asegura que el envío de datos sea asíncrono, evitando que la Interfaz Gráfica se congele.

6.2.2. Servidor

El código del Servidor reside en la variable HUB_PROGRAM dentro del componente Servidor, es un script de MicroPython diseñado para ejecutarse en el Hub LEGO Spike Prime.
URL del repositorio de GitHub: https://github.com/M47355/Proyecto-1.git

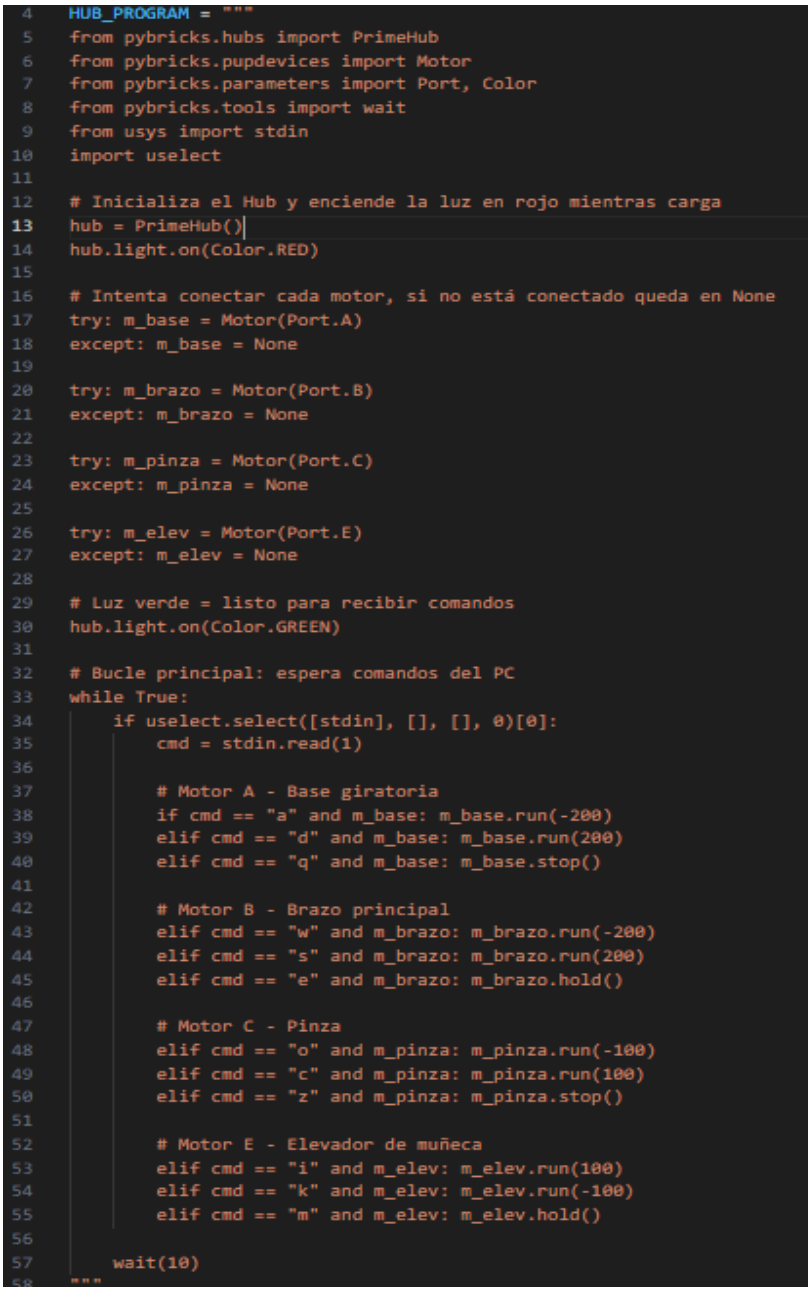
Tabla 13

Componentes importantes del código del Servidor

Componente	Explicación	Función dentro del sistema
Inicialización y Configuración	Utiliza try/except para mapear los motores a los puertos A, B, C y E (m_base, m_brazo, m_pinza, m_elev) y enciende la luz verde para indicar que está listo.	Prepara el hardware del robot y proporciona confirmación visual del estado del programa.
Módulo de Recepción (Bucle Principal)	El bucle while True utiliza uselect.select([stdin], [], [], 0) para esperar comandos sin detener el Hub, y luego lee el carácter con cmd = stdin.read(1).	Escucha activa de comandos remotos desde el Cliente.
Lógica de Actuación y Mapeo	Una serie de condicionales elif que traducen el carácter recibido a una acción motora específica:	Controla los movimientos del robot.
Movimiento Continuo	m_base.run(200) (comandos 'a', 'd', 'w', 's', etc.).	Mantiene el movimiento mientras el operador presiona el botón.
Parada	m_base.stop() (comandos 'q', 'z').	Detiene el motor de la base y la pinza.
Freno Activo	m_brazo.hold() (comandos 'e', 'm').	Aplica un freno activo para mantener la posición del brazo y el elevador, contrarrestando el peso de la carga y la gravedad.

Figura 7

Código del Servidor (MicroPython)



```
HUB_PROGRAM = """
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor
from pybricks.parameters import Port, Color
from pybricks.tools import wait
from usys import stdin
import uselect

# Inicializa el Hub y enciende la luz en rojo mientras carga
hub = PrimeHub()
hub.light.on(Color.RED)

# Intenta conectar cada motor, si no está conectado queda en None
try: m_base = Motor(Port.A)
except: m_base = None

try: m_brazo = Motor(Port.B)
except: m_brazo = None

try: m_pinza = Motor(Port.C)
except: m_pinza = None

try: m_elev = Motor(Port.E)
except: m_elev = None

# Luz verde = listo para recibir comandos
hub.light.on(Color.GREEN)

# Bucle principal: espera comandos del PC
while True:
    if uselect.select([stdin], [], [], 0)[0]:
        cmd = stdin.read(1)

        # Motor A - Base giratoria
        if cmd == "a" and m_base: m_base.run(-200)
        elif cmd == "d" and m_base: m_base.run(200)
        elif cmd == "q" and m_base: m_base.stop()

        # Motor B - Brazo principal
        elif cmd == "w" and m_brazo: m_brazo.run(-200)
        elif cmd == "s" and m_brazo: m_brazo.run(200)
        elif cmd == "e" and m_brazo: m_brazo.hold()

        # Motor C - Pinza
        elif cmd == "o" and m_pinza: m_pinza.run(-100)
        elif cmd == "c" and m_pinza: m_pinza.run(100)
        elif cmd == "z" and m_pinza: m_pinza.stop()

        # Motor E - Elevador de muñeca
        elif cmd == "i" and m_elev: m_elev.run(100)
        elif cmd == "k" and m_elev: m_elev.run(-100)
        elif cmd == "m" and m_elev: m_elev.hold()

    wait(10)
"""
```

Como se aprecia en la Figura 7, el código del servidor se define como una cadena de texto dentro del cliente para ser inyectada. Se observa la importación de librerías de `pybricks` y la configuración inicial de los motores, lo cual valida la capacidad del sistema para operar de forma autónoma una vez cargado en el Hub.

6.2.3. Interfaz gráfica de usuario (GUI)

La GUI fue implementada en Python utilizando la librería tkinter con un diseño temático oscuro y limpio, reside en el componente interfaz y se comunica con los demás componentes.

Figura 8

Interfaz implementada

Tabla 14

Componentes de la interfaz

Componente	Función
Botón de Conexión	Botón principal de la GUI, que inicia el proceso de conexión al robot.
Botones de Movimiento	Botones que envían el comando de movimiento al presionar y el comando de freno al soltar.
Atajos de Teclado	Indicaciones que señalan los atajos de teclado (W/S, A/D, O/C, I/K).
Botón [W] SUBIR	Botón que permite elevar el brazo principal del robot, permitiendo levantar objetos o posicionar el brazo a mayor altura.
Botón [S] BAJAR	Botón que permite descender el brazo principal, permitiendo bajar el brazo hacia la superficie de trabajo.
Botón [A] HORARIO	Botón que permite girar la base del brazo robótico en sentido horario, cambiando la orientación del robot.
Botón [D] ANTI-HORARIO	Botón que permite girar la base del brazo robótico en sentido antihorario.
Botón [O] ABRIR	Botón que permite abrir la pinza del brazo robótico para soltar objetos.
Botón [C] CERRAR	Botón que permite cerrar la pinza para sujetar firmemente los objetos.

Tabla 15

Continuación de componentes de la interfaz

Botón [I] SUBIR	Botón que permite elevar la muñeca del brazo robótico para ajustar la altura del extremo del brazo.
Botón [K] BAJAR	Botón que permite descender la muñeca del brazo robótico para posicionar correctamente la pinza.

6.2.4. Integración del sistema (Trazabilidad)

Flujo de ejecución Para validar la integración del sistema, se detalla el flujo técnico de la acción "Subir Brazo", desde el clic del usuario hasta el movimiento físico:

1. **Interacción en GUI (Cliente):** El usuario presiona la tecla **W**. La función **start_move** captura el evento:

```
def start_move(self, event):
    if key == 'w': self.controller.send(b'w') # Envía byte
```

2. **Transmisión:** El carácter **'w'** viaja vía Bluetooth (BLE) usando la librería **pybricksdev/bleak**.

3. **Ejecución (Servidor):** El Hub recibe el comando y acciona el motor:

```
# Servidor (MicroPython en Hub)
if cmd == b'w':
    motor_brazo.run(50) # Velocidad positiva
elif cmd == b'e':
    motor_brazo.hold() # Freno activo al soltar
```

7. Resultados

7.1 Estado actual del proyecto

La solución propuesta para la primera fase del proyecto, el brazo robótico de lego teleoperado, se encuentra en una fase de desarrollo avanzado y funcional. Se ha logrado establecer un sistema de control Cliente-Server completamente operativo, que permite la manipulación remota de la carga a través de una interfaz gráfica de usuario(GUI).

Tabla 16

Resultados de las pruebas de cumplimiento de requisitos

Requisito inicial	Estado	Detalle del cumplimiento
Control remoto	Cumplido	El sistema utiliza comunicación BLE gestionada por la clase RobotController para enviar comandos desde el PC al Hub Lego Spike Prime.
Control de los motores	Cumplido	Los cuatro motores responden correctamente a los comandos enviados por la GUI y los atajos de teclado.
Estabilidad de posición	Cumplido	Se implementó la función motor.hold() en los comandos de liberación de los ejes verticales asegurando que el brazo mantenga la carga contra la gravedad.
Interfaz de usuario	Cumplido	Se desarrolló una interfaz gráfica en tkinter que visualiza el estado de la conexión y proporciona un control táctil y por teclado, optimizando la usabilidad.
Capacidad de manipulación	Cumplido	El brazo puede realizar un ciclo completo de trabajo (acercamiento, agarre, elevación, movimiento lateral y liberación) con la carga nominal.

7.2. Problemas encontrados y solucionados

Durante la implementación se enfrentaron dificultades técnicas relevantes:

1. Bloqueo de la Interfaz Gráfica (GUI): Inicialmente, al intentar conectar con el robot, la ventana de la aplicación dejaba de responder.
   Solución: Se implementó programación asíncrona (asyncio) corriendo en un hilo secundario (Daemon Thread). Esto separa la lógica de comunicación del bucle principal de la interfaz visual.

2. Caída del brazo por gravedad: El motor del brazo principal (Motor B) cedía ante el peso de la estructura cuando no recibía órdenes.
   Solución: Se modificó el código del servidor (MicroPython) para cambiar el método de parada. En lugar de usar stop() (que deja el motor libre), se implementó hold() que aplica un torque activo para mantener la posición, resolviendo el problema de estabilidad de la carga.

3. Latencia en la respuesta: Se detectó un retraso al enviar comandos complejos.
   Solución: Se simplificó el protocolo de comunicación a caracteres simples (ej: "w", "s") en lugar de cadenas de texto largas, reduciendo el tiempo de procesamiento y transmisión de datos.

8. Prueba de funcionamiento del sistema

8.1. Descripción de la prueba de funcionamiento

De acuerdo con los requerimientos de la fase final, la prueba validó la capacidad del brazo robótico para completar el ciclo de manipulación de material minero en tres etapas críticas:

1. **Carga (Base a Vehículo):** El brazo debe tomar el material minero (representado por una estructura de piezas de LEGO simulando material "crudo") desde la **base inicial** y cargarlo con precisión sobre el **vehículo transportador**.
2. **Transporte Simulado:** Se verifica que la carga quede estable sobre el vehículo para su traslado.
3. **Descarga y Transferencia (Vehículo a Clasificador):** El brazo debe retirar el material desde el vehículo transportador y transferirlo a la zona de entrada del **robot clasificador** para su posterior procesamiento.

Esta secuencia demuestra el control total de los ejes y la capacidad de la garra para manipular la carga sin desarmar la estructura ni dejarla caer durante las transiciones.

8.2. Resultados observados para la prueba de funcionamiento

Durante la ejecución de la prueba definida y documentada en la demostración (Demo), se obtuvieron los siguientes resultados técnicos y operativos:

- **Ciclo de Carga y Descarga Exitoso:** El brazo robótico logró completar el flujo requerido sin interrupciones. Se extrajo la estructura de LEGO (material crudo) desde la base inicial, se depositó sobre el **vehículo transportador** asegurando su estabilidad y, posteriormente, se retiró del vehículo para transferirlo a la zona del **robot clasificador**.
- **Manipulación de Carga:** La pinza mantuvo una presión constante sobre el objeto de 0.15 kg, evitando caídas durante los movimientos de rotación y elevación. La estructura de bloques no sufrió desmembramiento ni daños durante la transferencia.
- **Precisión de Posicionamiento:** El control a través de la GUI permitió al operador alinear la garra con el vehículo transportador con precisión milimétrica. El sistema de frenado activo (motor.hold()) fue determinante para mantener la carga suspendida mientras se realizaban los ajustes finos de posición antes de la descarga.
- **Rendimiento de la Comunicación:** La latencia entre la orden en la GUI y la reacción del robot se mantuvo por debajo del umbral de **200 ms**, permitiendo una operación fluida en tiempo real durante todo el ciclo.

Videos de demostración: Demostraciones

9. Conclusiones

En este proyecto logramos diseñar, construir e implementar un brazo robótico teleoperado funcional, integrando exitosamente una arquitectura Cliente-Servidor entre Python y MicroPython. Se cumplió el objetivo de controlar los 4 ejes de movimiento de forma inalámbrica, validando la seguridad del sistema mediante pruebas de carga y descarga.

Durante el desarrollo, el aprendizaje técnico más significativo fue el dominio de la programación asíncrona (librería asyncio). Nos enfrentamos a un problema crítico donde la interfaz gráfica se congelaba al esperar respuestas del robot; esto se solucionó implementando la comunicación en hilos paralelos no bloqueantes, lo que fue vital para mantener una latencia baja y una experiencia de usuario fluida.

Aunque el prototipo cumple con los requisitos de seguridad pasiva y control, identificamos oportunidades claras de mejora. Como trabajo futuro, se propone dotar al sistema de movilidad autónoma (chasis con ruedas) e integrar sensores para asistir el agarre. Concluimos que la robótica teleoperada es una herramienta viable y escalable para mitigar riesgos humanos en la minería subterránea, con potencial de expansión a otras industrias peligrosas.

10. Referencias

[1] Falabella. (s.f.). *Samsung Galaxy Tab S9 FE 256 GB*. Recuperado el 10 de diciembre de 2025, de https://www.falabella.com/falabella-cl/product/132630394/Samsung-Galaxy-Tab-S9-FE-256GB/132630395

[2] SmartDeal.cl. (2023). *Notebook Lenovo IdeaPad 5 15ITL05*. Recuperado el 14 de junio de 2025, de https://www.smartdeal.cl/producto/notebook-lenovo-ideapad-5-15itl05-intel-core-i7-512gb-ssd-12gb-ram-15-fhd/

[3] The LEGO Group. (s.f.). *LEGO Education SPIKE Prime Set*. Recuperado el 10 de diciembre de 2025, de https://education.lego.com/en-us/products/lego-education-spike-prime-set/45678

[4] Opcstore. (s.f.). *Notebook Lenovo V14 G2 ALC Ryzen 5*. Recuperado el 10 de diciembre de 2025, de https://opcstore.cl/products/notebook-lenovo-v14-alc-8gb-ssd-512gb-14-hd-w10-pro

[5] The LEGO Group. (s.f.). *LEGO Education SPIKE Prime Expansion Set*. Recuperado el 10 de diciembre de 2025, de https://education.lego.com/en-us/products/lego-education-spike-prime-expansion-set/45681

[6] Computrabajo. (s.f.). *Salarios para el cargo de Programador en Chile*. Recuperado el 14 de diciembre de 2025, de https://cl.computrabajo.com/trabajo-de-programador

[7] Garcia, F. (2025). *Todo sobre la arquitectura cliente-servidor*. Arsys Blog. https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor

[8] Lecher, D. (s.f.). *Pybricksdev: Python Package with Pybricks developer tools* [Repositorio de código]. GitHub. https://github.com/pybricks/pybricksdev

[9] The Pybricks Authors. (s.f.). *Pybricks Documentation v3.6.1*. Recuperado el 12 de diciembre de 2025, de https://docs.pybricks.com/