UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Plan de Proyecto "Maquinaria de clasificación de materiales"

Alumnos: Bastian Hernandez
Saoud Ahmed
Alex Campillay
Marcos Caldas
Enzo Llancabure

Asignatura: Proyecto I

Profesor: Baris Klobertanz Quiroz

22 de Septiembre – 2025

Historial De Cambios

Fecha	Versión	Descripción	Autor(es)
22/09	1.0	Reconocimiento del Problema y Formalización del Proyecto	Todos
26/09	1.1	Finalización de la introducción	Marcos Caldas Enzo Llancabure Bastian Hernandez
08/10	1.2	Finalización de la organización de personal	Marcos Caldas Bastian Hernandez Enzo Llancabure
13/10	1.3	Finalización de la planificación del proyecto	Enzo Llancabure Marcos Caldas Bastian Hernandez
17/10	1.4	Finalización de la planificación de recursos	Enzo Llancabure Marcos Caldas
17/10	1.5	Finalización de la conclusión	Enzo LLancabure

17/10	1.6	Finalización de las referencias	Bastian Hernandez

Tabla N°1 “Historial de Cambios”.

➢ Índice

➢ Índice de tablas

➢ Indice de imagenes

1. Panel General

1.1. Introducción

En la industria minera se evidencia múltiples veces que la clasificación de materiales es una parte importante de la operación; sin embargo, la ejecución ineficiente de esto puede llevar a cuellos de botella operativos. Aún más grave estos procesos pueden llevar a la exposición de los trabajadores a entornos peligrosos si no están debidamente automatizados, evidenciando el problema claramente: La necesidad de implementar un sistema que garantice eficiencia sin comprometer la integralidad de los trabajadores.

Bajo esta premisa, en este informe se demostrará el trabajo colaborativo realizado por el equipo para cumplir el objetivo de mejorar la productividad, optimizar los procesos y elevar los estándares de seguridad de un equipo de minería. Para lograrlo se aplicaron conocimientos de ingeniería mediante una simulación de un clasificador de materiales fabricado con el set de Lego Spike Prime.

A partir de lo anterior, la estrategia seguida para la organización de actividades y asignación de responsabilidades se detalla en los siguientes apartados.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar y programar un robot con el kit LEGO Spike Prime que sea capaz de clasificar bloques en una casilla designada según su color, simulando un proceso de clasificación de materiales de una industria minera.

1.2.2. Objetivos Específicos

- Experimentar para ser capaz de usar correctamente el set Lego Spike Prime hasta el punto de poder aplicar correctamente todas las funciones necesarias para crear el robot clasificador en un tiempo estimado de 2 semanas.

- Asignar roles a cada integrante del grupo para mantener un responsable en cada área de trabajo y poder realizar el proyecto de forma eficiente, en un plazo de 1 semana.

- Planificar y ensamblar un prototipo inicial con el set LEGO Spike Prime que pueda detectar con el sensor los colores de los bloques y mover los motores respectivamente del color, en un plazo de 3 semanas.

- Programar con el lenguaje de programación Python en la aplicacion LEGO Spike un codigo capaz de conectar el sensor de color junto con los motores al mismo tiempo para buscar la rapidez del ordenamiento de bloques, en el tiempo que el ensamblador lo tenga que necesitar.

- Construir el robot que cumpla con una buena coordinación y velocidad con el set Lego Spike Prime, librería Pybricks y Visual Studio Code, además que sea capaz de clasificar correctamente cada bloque en una casilla, antes de la presentación final.

- Documentar y subir los archivos en formato PDF a la plataforma de Redmine para registrar cada avance que se va a realizar hasta la fecha final del proyecto.

- Implementar una interfaz para el robot clasificador que sea capaz de realizar movimientos de este, dependiendo de los botones de la interfaz, utilizando Tkinter y Visual Studio Code, una vez terminado el robot, en un plazo de 3 semanas.

- Definir un repositorio en GitHub para tener un historial de cambios, además de un sitio donde cada integrante será capaz de ver el código, dentro de la web de GitHub, hasta que se finalice el trabajo de codificación.

- Crear un manual de usuario con las instrucciones de cómo usar el robot, en Microsoft Word, al finalizar la construcción y codificación del robot, en un plazo de 1 semana.

- Realizar una presentación que sea capaz de demostrar que el robot clasificador puede realizar una buena clasificación de bloques, utilizando la web Canva, en un tiempo estimado de 1 semana.

1.3. Restricciones

Las restricciones son requerimientos mínimos que deben ser obligatoriamente cumplidos.
Las restricciones que se tiene en este proyecto son:

- Si es necesario, utilizar la extensión de Lego Spike Prime.
- Solo se debe utilizar la plataforma Redmine para los documentos y avance del proyecto.
- Se debe utilizar el Set de Lego Spike Prime.
- Tiempo limitado para la finalización del proyecto, debido al término del semestre.
- Cantidad de integrantes limitada a 5.
- La disponibilidad del robot para su uso sea codificación y construcción, está limitada al horario del departamento de ICCI (Ingenieria civil en computacion e Informatica).
- Robot debe ser capaz de reconocer el bloque y poder clasificarlo dependiendo de su color.
- La conexión entre el pc y el robot debe ser inalámbrica.

1.4. Entregables

Bitácoras: Son informes semanales que describen el avance del equipo en el proyecto, abarcando actividades realizadas, dificultades encontradas, recomendaciones para mejorar y acciones tomadas. Preparadas por los integrantes del grupo, ofrecen un panorama exhaustivo para apoyar decisiones estratégicas, asignan responsabilidades y resaltan asuntos a tratar en grupo.

Carta Gantt: Representación visual de la programación del proyecto, mostrando en una línea de tiempo las tareas, su duración y secuencia, facilitando la gestión del tiempo y los recursos al visualizar la evolución de las actividades a lo largo del proyecto.

Informes de avance: Este documento detalla la organización y estrategia para alcanzar los objetivos de la asignatura. Se abordará la asignación de roles, las metas del equipo y las medidas que implementarán para lograr el propósito académico. Además, se comparten las primeras impresiones durante el proceso de desarrollo y se presenta la documentación relevante recopilada a lo largo del semestre.

Presentaciones: Se harán presentaciones con el formato de PDF, para presentar la información recopilada en los informes entregados, cada presentación tratará los temas de cada informe ya sea avance o posteriores a él.

Manual de Usuario: Se desarrolla un manual de usuario para el funcionamiento correcto del robot enfocado en el aprendizaje para el usuario.

2. Organización del Personal

La organización en un grupo es importante para lograr la finalización de este proyecto, como también un buen ambiente en el grupo.

2.1. Descripción de los Roles

Jefe de proyecto: Representante del equipo, supervisa y organiza el progreso del proyecto.

Ensamblador: Encargado del montaje y el armado de las piezas, monitorea el cumplimiento de las funcionalidades del robot, en conjunto con el programador.

Programador: Encargado del área de la codificación y la lógica de funcionamiento del robot, trabajando en colaboración con el ensamblador para integrar el software con el hardware.

Documentador: Encargado de registrar el avance del proyecto y la redacción de los informes. Para cumplir su función, debe mantener una comunicación constante con todos los miembros del equipo, con el fin de recopilar y unir la información de cada área.

2.2. Personal que Cumplirá los Roles

Rol	**Responsable**
Jefe de proyecto	Bastian Hernandez
Ensamblador	Alex Campillay
Programador	Saoud Ahmed
Documentador	Enzo Llancabure Marcos Caldas Bastian Hernandez

Tabla N°2 “Roles”.

2.3. Métodos de Comunicación

Los principales medios de comunicación que usarán a lo largo del desarrollo del proyecto serán los siguientes:

-WhatsApp: Se utilizará para mensajería haciendo uso de los grupos que ofrece la Aplicación.

-Github: Como repositorio de codificación con avance e historial de versiones de este mismo, es una comunicación enfocada a tareas técnicas.

-Discord: Reuniones a distancia aprovechando los canales de texto y voz que ofrece esta Aplicación.

-Horario del Taller de Clase: Reuniones presenciales y avance de forma presencial para comunicar el avance del proyecto mismo.

3. Planificación del Proyecto

3.1. Actividades

O.E	Nombre	Encargado
O.E.1	Dominar el LEGO Spike Prime para una óptima creación de Robot.	Todos los integrantes.
O.E.2	Asignación de los Roles para cada integrante del grupo.	Todos los integrantes.
O.E.3	Diseño y Ensamblaje del primer prototipo con el set LEGO Spike Prime	Alex Campillay, Bastian Hernandez.
O.E.4	Programación del Robot Lego encargado de clasificar.	Saoud Ahmed Marcos Caldas
O.E.5	Construcción final del robot LEGO clasificador.	Alex Campillay, Bastian Hernandez, Enzo Llancabure.
O.E.6	Documentación y Registro en la plataforma Redmine.	Saoud Ahmed, Bastian Hernandez, Marcos Caldas.
O.E.7	Implementación de la Interfaz de control para el Robot LEGO clasificador.	Marcos Caldas, Saoud Ahmed, Alex Campillay.
O.E.8	Gestión del Repositorio Github con el código de control del Robot LEGO clasificador.	Todos los integrantes.
O.E.9	Elaboración del Manual de Usuario para tener las instrucciones de uso del Robot LEGO clasificador.	Todos los integrantes.

O.E	Nombre	Encargado
O.E.10	Presentación del Robot Clasificador para una demostración sólida del Robot LEGO clasificador.	Todos los integrantes.

Tabla N°3 “Actividades”.

3.2. Carta Gantt

Esta herramienta permite visualizar la secuencia de las actividades, organizadas en fases como la codificación, el ensamblaje y la documentación. A través de este diagrama, se muestra el porcentaje de avance de cada tarea, ayudando a la organización del grupo a la hora de completar las tareas.

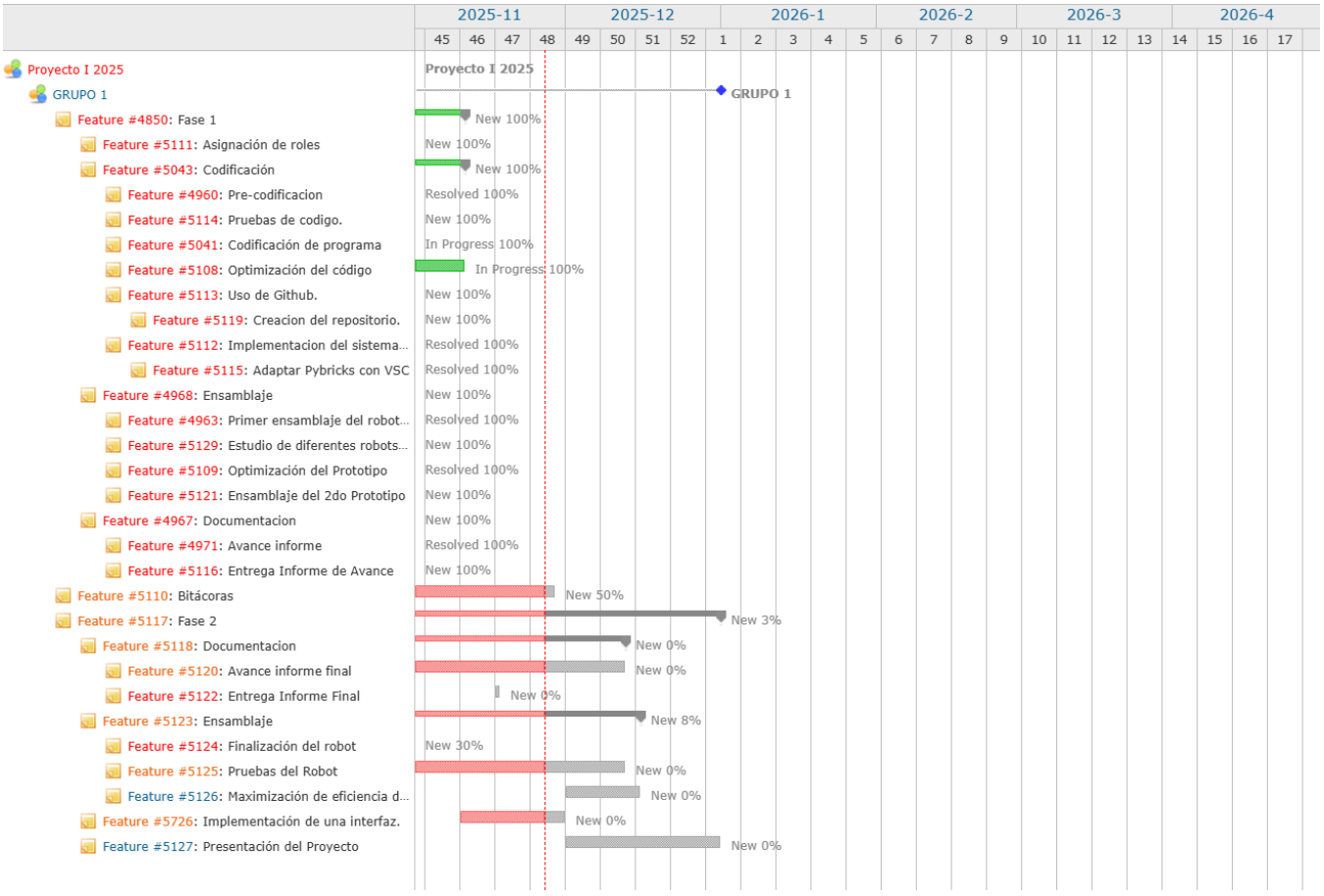


Ilustración N°1 “Carta Gantt”

3.3. Gestión de Riesgos

Para garantizar el cumplimiento de los objetivos del proyecto en los plazos establecidos, se ha elaborado una tabla de gestión de riesgos. Esta herramienta permite identificar, analizar y clasificar los posibles contratiempos que podrían afectar el desarrollo del prototipo robótico y la planificación general.

A continuación, se definen los niveles de severidad establecidos para clasificar cada riesgo según su impacto en el cronograma y en la operatividad del equipo:

1. **Impacto Crítico:** Problema al cual se le debe otorgar la máxima prioridad de resolución por parte del equipo, del caso contrario puede impactar en la entrega puntual del proyecto por retrasos o incluso un reinicio total de este.
2. **Impacto Alto:** Evento de máxima gravedad que compromete la viabilidad del proyecto. Requiere medidas inmediatas, pudiendo implicar el reinicio de etapas completas o la redefinición del alcance.
3. **Impacto Medio:** Riesgo que genera retrasos significativos en una o varias etapas clave. Exige una respuesta prioritaria para evitar que el desfase afecte la fecha de entrega final.
4. **Impacto Bajo:** Riesgo menor o imprevisto cotidiano que no altera la ruta del proyecto y puede ser resuelto con acciones simples sin afectar los entregables principales.

Riesgo	Nivel de Impacto	Acción Remedial
Daño grave de hardware	1	Verificar conexión del puerto, en caso de daños irreparables conseguir repuestos inmediatamente.
Horario insuficiente para el cumplimiento de tareas en conjunto	2	Coordinar los horarios disponibles del grupo.
Desempeño del robot poco eficiente	2	Ensamblar un robot más adecuado siguiendo guías en línea o un nuevo diseño adaptándolo a lo requerido.
Problema a la hora de experimentar con el robot.	2	Buscar la forma de modificar la posición de las piezas para que el robot cumpla su función.
Falla en el registro de redmine	2	Comunicar al profesor para buscar una solución.
Ausencia de piezas	2	Solicitar la extensión de Lego Spike o pedir piezas al ayudante.
Error en la codificación	2	Corregir errores sintácticos y lógicos en lo posible, de no serlo investigar una solución o explorar otro tipo de solución.
Falta de internet en la sala	3	Compartir internet del celular o usar cables ethernet.

Atraso en el cumplimiento de tareas	3	Hablar con el grupo para avanzar el proyecto fuera de clases.
Integrante falta a una clase	4	El Integrante tiene la responsabilidad de ponerse al día con lo que se avanzó en esa clase.

Tabla N°4 “Gestión de riesgos”

4. Planificación de los Recursos

Este punto muestra los recursos requeridos para la ejecución del proyecto, el cual se divide en herramientas físicas y digitales. Para la tabla de costo trabajador, se le asigna un sueldo basado en un tarifa por hora para cada rol, se toma en cuenta la complejidad de cada rol para la asignación de tarifa. El cálculo final es la suma de horas totales del equipo.

4.1. Hardware

- Set Lego Spike Prime.
- Set Lego Spike Prime Extension.
- Computador con el sistema operativo necesario para poder programar las instrucciones para el robot.
- Tablets para poder hacer la documentación necesaria.

4.2. Software

- Sistema operativo Windows para programar las funciones del robot.
- Redmine, página para la organización del proyecto.
- Canva.
- Plataforma Lego Education Spike (Code).
- Flutter para creación de aplicaciones.
- Pybricks en conjunto con VS code para la codificación pero más centralizada para los robots LEGO.

4.3. Estimación de Costos

Costo de Hardware:

Producto	Cantidad	Precio (CLP)
Set Lego Spike	1	$ 622.879
Extension Lego Spike	1	$ 167.202
Notebook LOQ Gen 9	1	$ 769.993
Samsung Galaxy Tab S8 Ultra	1	$ 1.150.000
Lenovo IdeaPad 3 15	1	$ 699.990
Notebook Aspire G A515-58GM-56ZZ-1	1	$ 1.049.990
HP Pavilion Laptop 14-dv2xxx	1	$ 1.500.000
Total:	7	$ 6.005.054

Tabla N°5 “Costo de Hardware”

Costo de Trabajador:

Rol	Horas / Mes	Horas Extra	Precio / Hora (CLP)
Jefe de proyecto	18 horas	5 horas	$ 10.555
Programador	18 horas	5 horas	$ 7.500
Ensamblador	18 horas	5 horas	$ 7.000
Documentador	18 horas	5 horas	$ 3.500
Total :	-	-	$ 2.736.000

Tabla N°6 "Costo Trabajador"

Costo de Software:

Software	Precio (CLP)
Microsoft Office	$ 8.990
Visual Studio Code	$0
Redmine	$0
Pybricks	$0
Flutter	$0
Total	$8.990

Tabla N°7 "Costo Software"

Destacado:

- *La contabilización de las horas trabajadas comienza a partir de la formación del grupo de trabajo, la fecha de inicio fue el día 22 de septiembre del 2025.*
- *Para la contabilización de las horas de trabajo, se tuvo en cuenta el tiempo de trabajo en clases.*
- *Para la contabilización de las horas extras, se tuvo en cuenta el tiempo en las que se trabajó fuera del horario de clase.*
- *El cálculo por hora se basa en el sueldo mensual promedio de cada uno de los cargos del proyecto dividido en una jornada laboral de 180 horas mensuales.*
- *Para el pago de horas extras se aplicó un pago doble sobre el valor de la hora normal.*

Total de Costo:

Costo Hardware	$ 6.005.054
Costo Empleados	$ 2.736.000
Costo Software	$ 8.990
Total :	$ 8.750.044

Tabla N°8 “Costo Total”.

5. Análisis y Diseño

5.1 Especificación de requerimientos

Antes de especificar los requerimientos funcionales y no funcionales, se debe establecer que el cliente es una empresa minera, representada generalmente por el área de ingeniería y automatización de maquinaria, los cuales financian la implementación del sistema robótico encargado para clasificar los materiales, siendo su rol principal asegurar que la solución esperada por la empresa cumpla con los objetivos de mejorar la seguridad de los trabajadores, aumentar la eficiencia y optimizar el clasificación de materiales.
Y tomando en cuenta que el usuario no es necesariamente la misma empresa y que la probabilidad de que interactúe de forma directa con el robot diariamente es escasa, se puede representar al usuario como un operador de maquinaria dentro de la empresa. De esta manera teniendo en cuenta esos dos puntos se pueden definir los requerimientos funcionales y no funcionales.

5.1.1 Requerimientos funcionales

Algunos de los requerimientos funcionales básicos que debe cumplir el robot a la hora de ser implementado y accionado son:
RF1 : El robot debe ser capaz de identificar materiales distintos representados por cuatro colores de bloques de lego (Rojo, Amarillo, Azul, Verde).
RF2 : El robot debe ser capaz de encolar varios bloques de lego para luego atenderlos, una estructura idéntica al funcionamiento de una cola FIFO (First In, First Out) donde sale el primero que entra.
RF3 : El robot debe ser capaz de depositar el bloque de lego identificado por color en su compartimento correspondiente (los 4 compartimentos cada uno asignado a uno de los 4 colores de bloques de lego mencionados en el **RF1**)
RF4 : El robot debe ser capaz de realizar de manera automática los movimientos mencionados en los requerimientos funcionales anteriores (**RF1 , RF2, RF3**).
Además de poder realizarlos de manera manual por el usuario mediante los botones de la interfaz.

5.1.2 Requerimientos no funcionales

Por otro lado algunos de los atributos de calidad del software que a los stakeholders les podrían interesar para cualificar el proyecto serían:

- Disponibilidad : El robot deberá mantener una disponibilidad mínima del 98% durante la jornada operativa de la empresa minera. Ante una falla, el sistema deberá restablecer su operaciones en un tiempo máximo de 10 minutos, garantizando la eficacia del proceso.
  Métrica de Evaluación : Porcentaje de tiempo operativo

- Robustez : El sistema debe ser capaz de manejar errores de sensores, entradas incorrectas o interrupciones momentáneas de comunicación sin detenerse completamente.
  Métrica de Evaluación : Número de fallos no controlados.

- Rendimiento : El robot automatizado debe responder a las órdenes del operador o del sistema de control en un tiempo máximo de 1 segundo, asegurando el flujo de trabajo eficiente, además deberá mantener su desempeño sin interrupciones durante al menos 8 horas de operación continua.
  Métrica de Evaluación : Tiempo promedio de respuesta a comandos y estabilidad del rendimiento durante turno de trabajo.

- Usabilidad : La interfaz del control del sistema debe ser intuitiva, clara y autoexplicativa, permitiendo que un operador capacitado pueda utilizar el robot de forma autónoma tras una breve capacitación donde se le enseñaria:
  - El funcionamiento de los botones de la interfaz.
  - El movimiento del robot asociado a cada botón de la interfaz.
  - Uso del manual de usuario para cualquier problema que pueda ocurrir en la operación del robot.

  Métrica de Evaluación : Tiempo de capacitación requerido y cantidad de errores de operación cometidos.

5.2 Arquitectura de software

La arquitectura de software describe los componentes principales del sistema, la forma en que se organizan y cómo se comunican entre sí para cumplir con los objetivos del proyecto.

en este proyecto se utiliza una arquitectura cliente-servidor, donde la aplicación móvil actúa como cliente y el hub del robot LEGO Spike Prime actúa como servidor, recibe, interpreta y ejecuta las instrucciones enviadas desde la interfaz gráfica de usuario

- Modelo de arquitectura cliente-servidor
  en la arquitectura cliente-servidor, el sistema se divide en dos partes principales:

  Cliente: componente del sistema encargado de la interacción directa con el usuario. Su función principal es capturar las acciones realizadas por este (como presionar botones o manipular el joystick). interpretarlas y generar solicitudes o comandos que son enviados al servidor a través del medio de comunicación definido. El cliente no ejecuta acciones físicas, sino que actúa como intermediario entre el usuario y el sistema de control.

  servidor: componente del sistema responsable de recibir, procesar e interpretar las solicitudes enviadas por el cliente. a partir de dichas solicitudes, el servidor ejecuta la lógica necesaria para controlar los recursos físicos del sistema, como motores y sensores, realizando las acciones correspondiente en este proyecto, el servidor traduce los comandos recibidos en instrucciones comprensibles en este proyecto el servidor traduce los comandos recibidos en instrucciones comprensibles para el microcontrolador del robot y gestiona su ejecución.

Este modelo permite separar la lógica de control del robot de la interfaz gráfica, facilitando el mantenimiento, la depuración y la escalabilidad del sistema.

- componentes del sistema
  el sistema está compuesto por los siguientes elementos:

Cliente lógico:desarrollado en flutter. proporciona la interfaz para el usuario. permite al usuario controlar el robot sorter. envía comandos de texto al robot utilizando bluetooth.

solo se encarga del envío de instrucciones por lo que no controla directamente el hardware.

Interfaz gráfica de usuario (GUI):

forma parte del cliente lógico.

permite la interacción directa del usuario.

interpreta acciones como presionar botones o mover el joystick.

genera comandos digitales que representan dichas acciones.

muestra el estado de conexión y el modo de simulación.

Medio de comunicación (Bluetooth):

Es la línea de comunicación entre el cliente y el servidor.

Se utiliza una implementación de Bluetooth Low Energy(BLE).

Permite el envío de comandos en tiempo real.

No procesa información, solo se encarga de transmitir los datos generados por el cliente.

Servidor lógico:

Código desarrollado en Python, ejecutado en el entorno PyBricks.

Utiliza las bibliotecas de PyBricks para;

Recibir comandos enviados por Bluetooth.

Interpretar los mensajes recibidos.

Traducir los comandos de alto nivel a instrucciones compresibles por el microcontrolador.

Controla motores y sensores del robot según la instrucción recibida.

Puede devolver información de estado si el sistema lo requiere.

- Flujo de comunicación del sistema

  El flujo de trabajo comienza por el usuario el cual interactúa con la interfaz gráfica de la aplicación, está interpreta las acciones del usuario y genera un comando de texto correspondiente el cual se envía mediante Bluetooth

al hub del robot. el servidor lógico (PyBricks) recibe el comando el cual es interpretado y traducido a instrucciones internas para que el microcontrolador del hub ejecute las acciones solicitadas.

El movimiento del robot se controla mediante el joystick virtual, mientras que el motor de empuje se acciona mediante dos botones; L1, que activa el movimiento hacia un lado, y R1, que lo dirige hacia el otro.

Ilustración N°2 "Flujo de comunicación del sistema".

- Ventajas de la arquitectura utilizada

  la arquitectura cliente-Servidor utilizada ofrece múltiples ventajas:

  Separación clara entre interfaz gráfica y control.

  Mayor facilidad para depuración y pruebas.

  Escalabilidad para agregar nuevas funciones.

  Reutilización del código del robot con distintas interfaces de control.

  Esta arquitectura permite que la aplicación móvil se enfoque exclusivamente en la experiencia del usuario, mientras que el robot se encarga del control del hardware, mejorando la organización general del sistema y facilitando futuras modificaciones.

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

En esta sección se presenta el diseño inicial de la interfaz gráfica de usuario (GUI)

del sistema de control del robot sorter. Este diseño corresponde a un wireframe de baja fidelidad, cuyo objetivo es definir la estructura general de la aplicación, la disposición de sus elementos y la interacción básica del usuario, sin considerar detalles visuales<e, tipografías o estilos gráficos finales.

El wireframe permite planificar la interfaz antes de su implementación, facilitando la comprensión del funcionamiento del sistema y sirviendo como base para el desarrollo posterior en flutter.

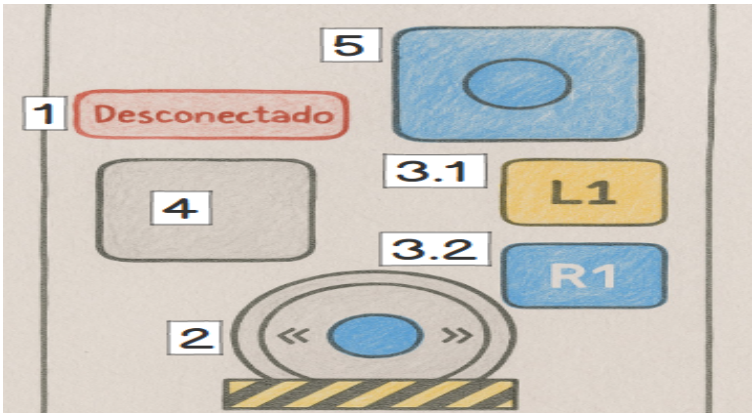


Ilustración N°3 "Wireframe GUI"

Este wireframe se utilizó como referencia directa para la implementación de la interfaz en flutter, donde cada forma del boceto se tradujo en widgets básicos como Container, Row, Column Switch y controles táctiles. El diseño inicial permitió definir claramente la distribución de la pantalla y la lógica de interacción antes de aplicar estilos visuales definitivos.

6. **Implementación**

En esta sección del informe se presentan los resultados obtenidos hasta el momento en el desarrollo del proyecto. Se incluye la justificación de la configuración del robot a partir de principios físicos de movimiento básico estudiados en la asignatura FI 035–Introducción a la Física, una descripción de los componentes más relevantes del sistema implementado y una captura de la interfaz gráfica de usuario ya desarrollada, junto con la explicación de sus elementos y funciones.

6.1 Fundamentos de los movimientos

Se presentan los distintos tipos de movimientos utilizados para el correcto funcionamiento del robot desde la primera aplicación física-matemática hasta la del último uso del robot. Se divide en tres secciones fundamentales que justifican a detalle el funcionamiento matemático del robot, las cuales son:

- Caída del Bloque de Lego:

Se toma en consideración un diagrama de cuerpo libre para utilizar la segunda ley de newton para determinar la aceleración ideal requerida para el correcto funcionamiento; tomando en cuenta diferentes datos, tales como: Peso del Bloque, Ángulo de inclinación de la caída y El roce efectuado del mismo plástico (DINÁMICA).

Datos:

Masa (m): 0.00232 kg.

Coeficiente de roce cinético (uk): 0,654 (estimado para plástico ABS).

Gravedad (g): 9.8 m/s^2.

Ángulo de inclinación(theta) : 60°.

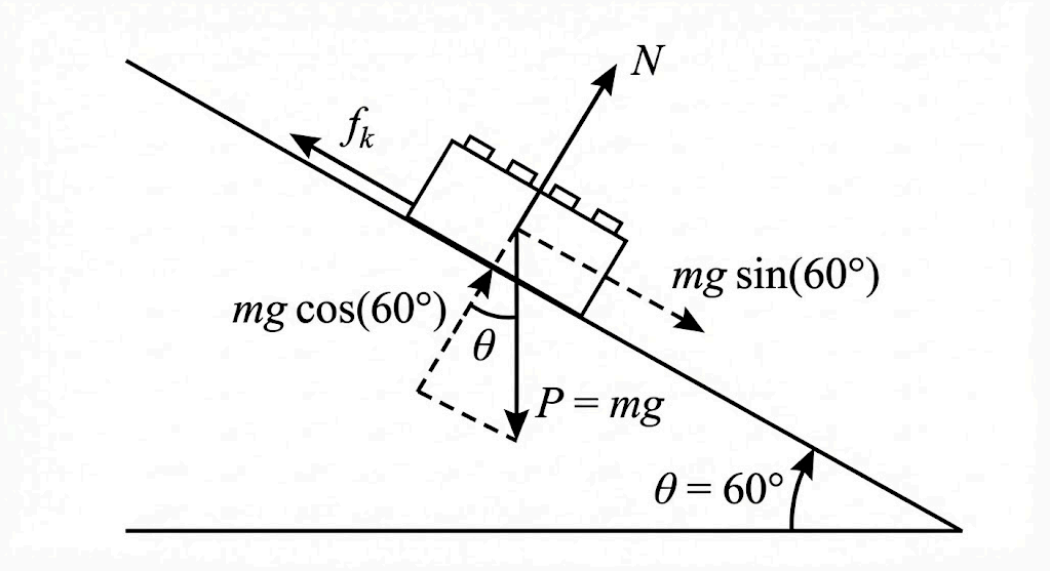


Ilustración N°4 "DCL"

Desarrollo: Analizar las fuerzas al ubicar el bloque sobre la superficie inclinada, que actúan dos fuerzas opuestas en el eje del movimiento: componente horizontal Px que favorece la caída y la fuerza de roce fk que se opone a ella.

Fuerza Normal (N): Es la fuerza de reacción perpendicular a la superficie.
N = m * g * cos(60) N = 0.00232 * 9.8 * 0.5 **N = 0.01137 Newtons.**

Fuerza de Roce Cinético (fk): Es la resistencia opuesta al movimiento, calculada con el coeficiente estimado. fk = uk * N fk = 0.654 * 0.01137 fk = 0.00744 Newtons

Componente del Peso en el eje X (Px): Es la fuerza generada por la gravedad que empuja el bloque hacia abajo por la rampa. Px = m * g * sen(60) Px = 0.00232 * 9.8 * 0.866
Px = 0.01969 Newtons

Teniendo los datos ya resueltos se aplica la segunda ley de Newton (Suma de fuerzas = m * a) la cual dirá la aceleración resultante:

a = (Px - fk) / m a = (0.01969 - 0.00744) / 0.00232 a = 0.01225 / 0.00232 **a = 5.28 m/s**2**

El dato de la aceleración de 5,28 m/s**2 es la ideal, menor a ella obtendrán

resultados poco eficientes en relación al tiempo de ejecución y mayor a ella se obtendrán múltiples errores desde caída con choques y el bloque expulsado hasta el no reconocimiento del sensor de color.

- Cálculo del Tiempo Real de posicionamiento

El giro de clasificación opera en un ángulo simétrico de -35° a +35° donde el ángulo 0° representa el reposo del clasificador. A diferencia del motor de empuje, que realiza un movimiento fijo (“golpe”), el motor de posicionamiento tiene trayectorias que cambian debido al sensor de color para el correcto ordenamiento del bloque.
Determinación de la velocidad real (w) para calcular la eficiencia del giro se calcula el recorrido total del robot de un extremo a otro, de -35° a +35°.
Distancia Máxima = 35 + 35 = 70°
Fórmula = t(max) = Diferencia Angular / velocidad real
La velocidad real está limitada por el mismo Hub del lego la cual es 300°/s .

t(max)= 70°/300°/s = 0.23 s

el tiempo de recorrido es 0,23 segundos, junto con la velocidad real que esta limitada por el Hub del lego spike prime es lo mas eficiente debido a que el recorrido por deteccion de color es minimo para girar y volver del reposo, los datos y resultados son a base del peor caso posible, ir de izquierda a derecha, de extremo a extremo.

- Eficiencia Motor de empuje (radio de empuje = 4 cm)

el motor de empuje se encarga de colocar los bloques en la sección indicada de su color
se selecciona un recorrido de 180° para empujar el bloque, es eficiente ya que al seleccionar no debe dar un giro completo, como si fuera 360°tardaría más en ejecutarse, implementando así efectividad en el ángulo de empuje ya que da un ida y retorno simple y rápido, junto con el tope máximo de velocidad admitida por el Hub (300°/s) para validar que esta configuración ofrece una fuerza óptima en un tiempo mínimo, se aplican las siguientes fórmulas:

Datos:

1. **Velocidad Angular Máxima (omega):** 300°/s (Límite operativo establecido en

el Hub).

2. **Radio de Palanca (r):** 4 cm (Longitud del brazo de empuje).
3. **Desplazamiento (theta):** 180°.

-Cálculo del tiempo de ciclo: determina cuanto tiempo tarda el brazo en completar el empuje al color indicado:
t = desplazamiento/velocidad angular = 180°/(300°/s) = 0.6 segundos

-Cálculo de la Fuerza de empuje (Newtons): determina la fuerza con la que se empuja el bloque (utilizamos el torque del motor = 18 N*CM).

FUERZA = TORQUE/RADIO = 18 N*CM/4CM = 4.5 NEWTONS

Los siguientes datos extraídos son necesarios ya que se necesita calcular la fuerza de empuje para asegurar que el motor tiene potencia suficiente para mover el bloque y añadimos velocidad para generar impacto teniendo en cuenta que buscamos la eficiencia respecto al tiempo.

6.2 Descripción del sistema

El sistema utilizado para el control y manejo del robot se basa en Pybricks, una plataforma que permite programar y controlar dispositivos robóticos de manera eficiente. A través de Pybricks, es posible implementar distintas formas de control del robot, lo que otorga flexibilidad al sistema y permite adaptarlo a diferentes métodos de interacción y operación.

Forma 1: Control_Automatico_Archivo_Base.py

Este corresponde al código inicial del sistema y opera de manera automática, es decir, no requiere la intervención directa de una persona para controlar el robot. Además, este código sirve como base de referencia para el desarrollo de los demás programas utilizados en el proyecto, ya que establece la lógica principal de funcionamiento del sistema.

Se puede ver el código en el repositorio github de este proyecto:

“LegoSpikePrimeProyecto1Sorting”.

Version1.0.0.py > ...

```python
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor, ColorSensor
from pybricks.parameters import Port, Color
from pybricks.tools import wait


hub = PrimeHub()

sensor_color = ColorSensor(Port.B)
motor_posicion = Motor(Port.A)
motor_empuje = Motor(Port.D)

while True:
    color_detectado = sensor_color.color()

    if color_detectado == Color.RED or color_detectado == Color.YELLOW:


        motor_posicion.run_target(1000, -35)
        if color_detectado == Color.RED:

            motor_empuje.run_angle(1000, -180)

        elif color_detectado == Color.YELLOW:
            motor_empuje.run_angle(1000, 180)


    elif color_detectado == Color.GREEN or color_detectado == Color.BLUE:

        motor_posicion.run_target(1000, 35)
        if color_detectado == Color.GREEN:
            motor_empuje.run_angle(1000, 180)

        elif color_detectado == Color.BLUE:
            motor_empuje.run_angle(1000, -180)

    wait(10)
```

Ilustración N°5 “Código Control_Automatico_Archivo_Base.py”.

Explicación del código del servidor lógico

Línea 1–4:

Se importan las librerías necesarias de Pybricks. Estas permiten utilizar el hub, los motores, el sensor de color, los puertos físicos del robot y funciones auxiliares como pausas de tiempo.

Línea 6:

Se crea una instancia del PrimeHub, que corresponde al hub físico del robot y actúa como el núcleo de control del sistema.

Línea 9:

El sensor implementado en el robot se guarda como referencia en la variable sensor_color, utilizando el puerto B. Este sensor se encarga de detectar el color del bloque que ingresa al sistema.

Línea 10:

Se inicializa el motor motor_posicion en el puerto A, el cual se utiliza para posicionar el mecanismo en la casilla correspondiente.
Línea 11:
Se inicializa el motor motor_empuje en el puerto D, encargado de empujar o disparar el bloque hacia el lado correspondiente según el color detectado.
Línea 13:
Se implementa un bucle while True, el cual permite que el robot funcione de manera continua. El programa se mantiene en ejecución hasta que se detiene manualmente.
Línea 14:
El color detectado por el sensor se almacena en la variable color_detectado, lo que permite utilizar este valor en la lógica de decisión del sistema.
Líneas 16–36:
En esta sección se encuentra el algoritmo principal de clasificación, el cual permite distribuir los bloques en cuatro casillas distintas. Para ello, se utilizan estructuras condicionales if y elif que evalúan el color detectado:
Si el color es rojo o amarillo, el motor de posición mueve el mecanismo hacia una casilla específica.
Si el color es rojo, el motor de empuje dispara el bloque hacia un lado mediante un giro de −180°.
Si el color es amarillo, el motor de empuje lo hace hacia el lado opuesto con un giro de 180°.
Si el color es verde o azul, el motor de posición se desplaza hacia otra casilla.
Para el color verde, el motor de empuje gira 180°.
Para el color azul, el motor de empuje gira −180°.
De esta forma, el sistema clasifica automáticamente los bloques según su color y los distribuye en la casilla correspondiente.
Línea 38:
Se utiliza la función wait(10) para introducir una pequeña pausa, lo que evita lecturas excesivamente rápidas del sensor y mejora la estabilidad del sistema.

Forma 2, archivo Archivo_Control_Teclado.py: En esta versión se ocupa de base el código anterior pero se implementa una forma de manejar el robot, en este caso sería mediante teclas.

Version1.0.1.py > ...

```python
# este codigo lee la entrada del teclado para controlar el robot
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor, ColorSensor
from pybricks.parameters import Port, Color
from pybricks.tools import wait
from usys import stdin
from uselect import poll


hub = PrimeHub()

sensor_color = ColorSensor(Port.B)
motor_posicion = Motor(Port.A)
motor_empuje = Motor(Port.D)

teclado = poll()
teclado.register(stdin)

while True:
    if teclado.poll(0):
        key = stdin.read(1)
        if key=='w':
            motor_posicion.run_target(1000, -35)
            motor_empuje.run_angle(1000, -180)

        elif key == 'a':
            motor_posicion.run_target(1000, -35)
            motor_empuje.run_angle(1000, 180)
        elif key=='s':
            motor_posicion.run_target(1000, 35)
            motor_empuje.run_angle(1000, 180)

        elif key=='d':
            motor_posicion.run_target(1000, 35)
            motor_empuje.run_angle(1000, -180)

    wait(10)

```

Ilustración N°6 “Codigo Archivo_Control_Teclado.py”.

Puntos claves de este código:

Línea 6–7: Se importan los módulos stdin y poll, los cuales permiten leer la entrada del teclado en tiempo real y detectar cuándo una tecla es presionada por el usuario.
Línea 15: Se crea la variable teclado utilizando poll(), la cual se encarga de monitorear continuamente las entradas del teclado.
Línea 16: Se registra la entrada estándar (stdin) en la variable teclado, habilitando la lectura de las teclas presionadas.
Línea 19: Dentro del bucle principal, se utiliza teclado.poll(0) para verificar si existe alguna entrada del teclado disponible sin detener la ejecución del programa.
Línea 20: La tecla presionada por el usuario se lee y se almacena en la variable key.
Líneas 21–36: Se implementa una estructura condicional que asocia distintas teclas del teclado con movimientos específicos del robot que son respectivamente con las casillas que se tienen, además de los colores que son 4.

Forma 3, carpeta ProyFlutter: En esta carpeta se tiene 2 versiones, app_control y app2_control, estas son 2 aplicaciones creadas para el manejo del robot mediante una conexión bluetooth, además de una interfaz gráfica para el usuario.

```dart
// INICIAR TECLADO BLUETOOTH
// --------------------------
Future<void> startKeyboard() async {
  try {
    await keyboard.start();
    setState(() {
      connected = true;
      status = "Conectado ✓ (Teclado Bluetooth)";
    });
  } catch (e) {
    setState(() {
      status = "Error: $e";
    });
  }
}

// ENVIAR TECLA AL PC
Future<void> sendKey(String key) async {
  if (!connected) return;

  try {
    await keyboard.sendText(key);
  } catch (e) {
    debugPrint("Error enviando: $e");
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.black,
    body: Padding(
      padding: const EdgeInsets.all(12),
      child: Column(
        children: [
          // ---------------- TOP BAR ----------------
          Row(
            children: [
              Expanded(
                child: Container(
                  padding: const EdgeInsets.all(10),
                  decoration: BoxDecoration(
                    color: connected ? Colors.green : Colors.red,
                    borderRadius: BorderRadius.circular(10),
                  ),
                  child: Text(
                    status,
                    textAlign: TextAlign.center,
                    overflow: TextOverflow.ellipsis,
                    style: const TextStyle(
                        fontSize: 18, fontWeight: FontWeight.bold),
                  ),
                ),
              ),
```

```
121                 // -------- LEFT BUTTONS --------
122                 Column(
123                   children: [
124                     ElevatedButton(
125                       onPressed: () => sendKey("w"),
126                       style: ElevatedButton.styleFrom(
127                         minimumSize: const Size(100, 80),
128                       ),
129                       child: const Text("L1 → w",
130                           style: TextStyle(fontSize: 24)),
131                     ),
132                     const SizedBox(height: 20),
133                     ElevatedButton(
134                       onPressed: () => sendKey("s"),
135                       style: ElevatedButton.styleFrom(
136                         minimumSize: const Size(100, 80),
137                       ),
138                       child: const Text("R1 → s",
139                           style: TextStyle(fontSize: 24)),
140                     ),
141                   ],
142                 ),
143
144                 // -------- COLOR BOX (decorativo) --------
145                 Container(
146                   width: 150,
147                   height: 150,
148                   decoration: BoxDecoration(
149                     color: sensorColor,
150                     borderRadius: BorderRadius.circular(20),
151                     border: Border.all(color: Colors.white, width: 3),
152                   ),
153                 ),
154
155                 // -------- JOYSTICK --------
156                 SizedBox(
157                   width: 220,
158                   height: 220,
159                   child: Joystick(
160                     mode: JoystickMode.all,
161                     listener: (details) {
162                       // Usa WASD para movimientos
163                       if (details.y < -0.5) sendKey("w");
164                       else if (details.y > 0.5) sendKey("s");
165
166                       if (details.x < -0.5) sendKey("a");
167                       else if (details.x > 0.5) sendKey("d");
168                     },
```

Ilustración N°7 “Código app2_control”.

línea 74-151, es como está decorado la aplicación, además de que envía texto hacia el computador.

línea 155-168, un joystick que envía texto como “WASD”.

6.2.1 Cliente

En el sistema desarrollado se distinguen dos tipos de cliente: el cliente físico y el cliente lógico.

El cliente físico corresponde al dispositivo desde el cual el usuario interactúa con el sistema, y este varía según el código o la aplicación utilizada. En el caso de utilizar la aplicación desarrollada en Flutter, el cliente físico es el teléfono móvil. Si se emplea el código Control_Automatico_Archivo_Base.py, el cliente físico es el computador. Finalmente, cuando se utiliza el código Archivo_Control_Teclado.py, el cliente físico corresponde al control DualShock 4.

Por otro lado, el cliente lógico se refiere al software encargado de gestionar la interacción con el usuario, específicamente la interfaz gráfica. En el estado actual del proyecto, este rol lo cumple únicamente la aplicación desarrollada en Flutter, ya que es la única que incorpora una interfaz gráfica para el usuario.

```
81              // ---------------- TOP BAR ----------------
82              Row(
83                children: [
84                  Expanded(
85                    child: Container(
86                      padding: const EdgeInsets.all(10),
87                      decoration: BoxDecoration(
88                        color: connected ? Colors.green : Colors.red,
89                        borderRadius: BorderRadius.circular(10),
90                      ), // BoxDecoration
91                      child: Text(
92                        status,
93                        textAlign: TextAlign.center,
94                        overflow: TextOverflow.ellipsis,
95                        style: const TextStyle(
96                            fontSize: 18, fontWeight: FontWeight.bold), // TextStyle
97                      ), // Text
98                    ), // Container
99                  ), // Expanded
100
101                 const SizedBox(width: 12),
102
103                 ElevatedButton(
104                   onPressed: connected ? null : startKeyboard,
105                   style: ElevatedButton.styleFrom(
106                     backgroundColor: Colors.blue,
107                     padding:
108                         const EdgeInsets.symmetric(horizontal: 20, vertical: 12),
109                   ),
110                   child: Text(connected ? "CONECTADO" : "CONECTAR"),
111                 ), // ElevatedButton
112               ],
113             ), // Row
```

Ilustración N°8 “Código de la app2_control interfaz gráfica”.

Este fragmento de código corresponde a la barra superior (Top Bar) de la interfaz gráfica desarrollada en Flutter. Su objetivo principal es mostrar el estado de conexión y permitir iniciar la conexión mediante un botón.

Primero, se define un Row, el cual representa una fila. Su principal función es organizar los componentes de forma horizontal dentro de la interfaz.
Luego, se crea un Container donde se muestra un mensaje de estado centrado. Este mensaje cambia de color dependiendo del estado de conexión del sistema: verde cuando el sistema se encuentra conectado y rojo cuando no existe conexión.

Por último, en esta sección de la Top Bar se implementa un ElevatedButton, cuyo texto varía según el estado del sistema. El botón muestra “CONECTAR” cuando el sistema no está conectado y “CONECTADO” cuando la conexión ya ha sido establecida.

Acerca de los mensajes que se envían

Al presionar el botón “CONECTAR”, se ejecuta la función startKeyboard, la cual se encarga de inicializar la comunicación entre el cliente y el servidor.
Otra parte de la interfaz gráfica hecha con Flutter y que sirve como cliente lógico es el joystick creado. Este componente permite al usuario controlar el robot de forma manual mediante una interfaz táctil, simulando el funcionamiento de un control direccional.

El joystick se encuentra contenido dentro de un SizedBox, el cual define un tamaño fijo para asegurar una correcta visualización y uso dentro de la aplicación. El componente se configura para detectar movimientos tanto en el eje horizontal como en el eje vertical, lo que permite interpretar desplazamientos en todas las direcciones.

Cada vez que el usuario mueve el joystick, se ejecuta un listener que analiza la dirección del movimiento. Dependiendo del desplazamiento detectado, el sistema traduce el movimiento del joystick en comandos equivalentes a las teclas W, A, S y

D. Estos comandos representan movimientos hacia adelante, atrás, izquierda y derecha, respectivamente.

Una vez interpretada la dirección, el joystick envía el mensaje correspondiente mediante la función sendKey, la cual transmite la instrucción al servidor. De esta manera, el servidor recibe los comandos y los interpreta para ejecutar los movimientos del robot, permitiendo un control remoto e interactivo a través de la interfaz gráfica.

```
155                  // -------- JOYSTICK --------
156                  SizedBox(
157                    width: 220,
158                    height: 220,
159                    child: Joystick(
160                      mode: JoystickMode.all,
161                      listener: (details) {
162                        // Usa WASD para movimientos
163                        if (details.y < -0.5) sendKey("w");
164                        else if (details.y > 0.5) sendKey("s");
165
166                        if (details.x < -0.5) sendKey("a");
167                        else if (details.x > 0.5) sendKey("d");
168                      },
169                    ),
170                  ),
171                ],
172              ),
```

Ilustración N°9 “Codigo app2_control”.

De la propia app de flutter también se envían mensajes pero no directamente al hub, sino al computador y mediante Archivo_Control_Teclado.py se pueden controlar y usar los métodos anteriormente mencionados.

6.2.2 Servidor

En el sistema desarrollado se distinguen dos tipos de servidores: el servidor físico y el servidor lógico.

El servidor físico corresponde al dispositivo encargado de recibir los mensajes enviados por el cliente. En este proyecto, dicho servidor es el LEGO Technic Large Hub, el cual recibe el código y se comunica con sus distintos puertos para ejecutar las acciones correspondientes. Estas acciones incluyen el control de los motores, así como el uso de sensores y otras funcionalidades integradas en el hub.

Por otro lado, el servidor lógico se refiere al software responsable de gestionar la comunicación con el cliente, específicamente el algoritmo que interpreta los mensajes que el hub debe procesar. En el estado actual del proyecto, este rol lo cumple el código Control_Automatico_Archivo_Base.py, el cual define la lógica de control y operación del sistema.

A continuación, se muestra la lógica base del servidor utilizada en el sistema desarrollado:

```
15
16      if color_detectado == Color.RED or color_detectado == Color.YELLOW:
17
18
19          motor_posicion.run_target(1000, -35)
20          if color_detectado == Color.RED:
21
22              motor_empuje.run_angle(1000, -180)
23
24          elif color_detectado == Color.YELLOW:
25              motor_empuje.run_angle(1000, 180)
26
27
28      elif color_detectado == Color.GREEN or color_detectado == Color.BLUE:
29
30          motor_posicion.run_target(1000, 35)
31          if color_detectado == Color.GREEN:
32              motor_empuje.run_angle(1000, 180)
33
34          elif color_detectado == Color.BLUE:
35              motor_empuje.run_angle(1000, -180)
36
```

Ilustración N°10 “Control_Automatico_Archivo_Base.py”.

Las instrucciones que se envían al hub corresponden principalmente a métodos de la librería Pybricks, específicamente run_target y run_angle.

El método run_target se utiliza cuando se requiere mover un motor hacia una posición específica. Este método es fundamental para posicionar el motor en las casillas designadas dentro del sistema, asegurando un desplazamiento preciso y controlado.

Por otro lado, el método run_angle se emplea para mover el motor en función de un ángulo determinado. En este proyecto, este método se utiliza para accionar el mecanismo que dispara el bloque hacia el lado izquierdo o derecho. Por esta razón, los ángulos se representan como 180° y −180°, indicando la dirección del movimiento.

6.2.3 Interfaz gráfica de usuario (GUI)

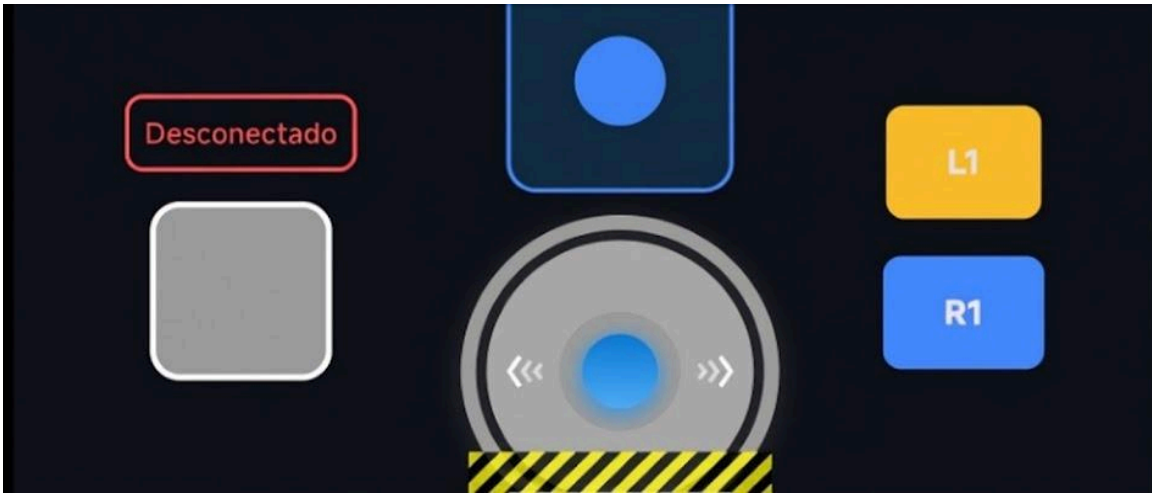


Ilustración N°11 “Interfaz Gráfica de Usuario”.

1. Indicador de estado de conexión

   Representado mediante un rectángulo que actualiza su estado de “desconectado” a la hora de hacer click en el. realiza una búsqueda por bluetooth de los dispositivos cambiando su estado de “desconectado” a “buscando” y en caso de encontrar la señal específica del robot sorter el estado queda en “conectado”

2. Joystick

   Representado como un círculo azul dentro de un círculo mayor como

límites el cual a la hora de ser manipulado por el usuario. Este envía señales al hub del robot sorter para que este mueva el motor de movimiento y poder rotarlo de lado a lado. este solo permite movimiento horizontal

3. Botones (L1) y (R1)

   Representados como dos botones de distintos colores e identificados como “L1” y “R1”, ambos envían instrucciones al hub del robot sorter para que este mueva el motor de empuje, siendo “L1” hacia el lado izquierdo y “R1” el lado derecho.

4. Muestreo de color

   Representado como un cuadro gris cuando no hay conexión o el robot no tiene colores detectados por el sensor, en caso de si tener colores sobre el sensor este se mostrara en el recuadro de muestreo de color para que el usuario efectue el ordenamiento con efectividad.

5. Panel informativo de dirección

   Representado como un cuadrado azul con un círculo dentro. este recuadro es una muestra de movimiento del joystick para otorgar comodidad visual al usuario

7. Resultados

7.1 Estado actual del proyecto

El estado actual de la solución planteada (Maquinaria de clasificación de materiales) se encuentra en una fase de funcionalidad completa respecto a su mecánica y cumpliento de los requerimientos funcionales definidos.

El robot ha logrado exitosamente la automatización total de clasificar bloques respecto a su color tanto de manera manual guiada por el usuario a través de un control remoto. La solución es capaz de operar de manera automática y manual replicando la tarea industrial sin exponer a funcionarios de empresas mineras.

El robot ha logrado alcanzar los siguientes requerimientos funcionales:

-Identificacion de Materiales (RF1): Mediante el sensor de color el robot es capaz de reconocer el color del bloque y separarlo en base a ello.

-Modo de operación Híbrida (RF4): El robot ha sido capaz de operar de manera automática, o sea, un bucle de ordenamiento ejecutando la clasificación de estos mismo bloques hasta no detectar ninguno. Como a su vez un modo manual que implementa un control remoto para que el usuario pueda clasificar los bloques el mismo

-Flujo Continuo (RF2): El robot es capaz de reconocer y ordenar varios bloques de LEGO sucesivamente , manteniendo un constante control de la clasificación.

-Clasificación de Celdas por su color (RF3): El robot es capaz de moverse y empujar el bloque respecto al destino de su color, capaz de sincronizar el movimiento de posición con el empuje del bloque para que el bloque sea colocado en base a su color, enfatizando en su ordenamiento.

Respecto a los requerimientos no funcionales del robot el reporte actual son los siguientes:

-Disponibilidad: La disponibilidad del robot depende netamente de su conexión con el programa, ante una pérdida de conexión con el hub y Pybricks el sistema se detiene por completo. Ante estos casos el robot debe ser reiniciado o desconectado con el hub y reintegrar la misma conexión, siendo su restauración casi de manera inmediata.

-Robustez: El robot no es tan robusto respecto al manejo mecánico debido a que si llega a presentar obstrucción por bloques, necesita intervención inmediata para el correcto flujo, carece de un mecanismo de auto barrido ante estancamiento.

-Rendimiento: El robot en condiciones de flujo normales tanto automáticas como manual cumple con los tiempos de respuesta esperados.

-Usabilidad: Para mejorar la usabilidad y comodidad se implementó un esquema en base a un control DualShock 4 (PS4) ya que es más conocido a nivel del usuario con botones como R1 Y L1, se debe mantener una capacitación antes de usarlo para que el usuario conozca con detenimiento el accionar de cada botón con el fin de cumplir el requerimiento de una interfaz clara e intuitiva. La capacitación para su correcto uso es mínima debido a la familiaridad de los botones implementados en relación al DualShock 4, en un tiempo de intervalo de 5 a 10 minutos como máximo. Los botones al ser grandes y claros minimizan el posible error de pulsación fantasma o más bien error de selección de botón, esto garantiza un accionar claro y conciso.

7.2 Problemas encontrados y solucionados

Durante el desarrollo del proyecto se enfrentaron problemas relevantes que afectaron el avance y funcionamiento de la aplicación. El principal problema estuvo relacionado con la comunicación entre la aplicación y el hub del robot, ya que inicialmente la aplicación no lograba detectar el hub mediante Bluetooth. Esto ocurría debido a las restricciones del sistema operativo Android, el cual requiere permisos explícitos de ubicación y Bluetooth para permitir la búsqueda y conexión con dispositivos externos.

Este problema se resolvió incorporando en la aplicación una solicitud de permisos al momento de su ejecución. Una vez aplicadas estas modificaciones, la interfaz logró detectar el hub correctamente, permitiendo continuar con el desarrollo y las pruebas del sistema.

Además, se identificó un problema organizacional relacionado con el uso del framework Flutter, ya que este solo estaba instalado en el computador del encargado de la programación. Esto generaba retrasos en el proyecto, ya que las pruebas y correcciones no podían realizarse de forma inmediata por todo el grupo. Para solucionar esta situación, se coordinó la instalación de Flutter en todos los equipos del grupo, lo que permitió realizar pruebas continuas, reducir los tiempos de corrección y mejorar el avance del proyecto.

8. Conclusión

En esta etapa del proyecto se lograron avances, abordando el análisis, diseño e implementación del robot. Se definieron los requerimientos funcionales y no funcionales, se diseñó una arquitectura de software basada en el modelo cliente-servidor y se desarrollaron distintas formas de operación del robot, incluyendo un modo automático y control manual.

Durante el desarrollo se enfrentaron dificultades principalmente asociadas a ajustes en el ensamblaje del robot, errores de codificación y limitación de tiempo, las cuales fueron resueltas mediantes pruebas, correcciones en el código y reuniones extras.

Como trabajo pendiente para las siguientes etapas, se considera la optimización del rendimiento del robot, la ejecución de pruebas finales de funcionamiento y la presentación final.

6. Referencias

Repositorio Github Lego Spike Prime.
https://github.com/saoudahmedlanchipa-code/LegoSpikePrimeProyecto1Sorting

Ubuy. (s. f.). *LEGO Education SPIKE Prime Set (45678)*. Recuperado el 15 de octubre de 2025, de https://www.ubuy.cl/sp/product/GMEH1T4-lego-education-spike-prime-set.

Tradeinn. (s. f.). *LEGO Education SPIKE Prime Expansion Set (45681)*. Recuperado el 15 de octubre de 2025, de https://www.tradeinn.com/kidinn/es/lego-juego-de-construccion-education-spike-prime-expansion-set-45681/141562226/p.

Lenovo. (s. f.). *Lenovo LOQ 15IAx9*. Recuperado el 15 de octubre de 2025, de https://www.lenovo.com/cl/es/p/notebooks/loq-laptops/lenovo-loq-15iax9/len101q0006.

RIPLEY. (s. f.). *TABLET SAMSUNG GALAXY TAB S8 PLUS + KEYBOARD COVER AMD RYZEN 9 8 GB RAM 12.4".* Recuperado el 15 de octubre de 2025, de TABLET SAMSUNG GALAXY TAB S8 PLUS + KEYBOARD COVER AMD RYZEN 9 8 GB RAM 12.4"

Paris.cl. (s. f.). *Notebook IdeaPad Slim 3 AMD Ryzen7 5825U 16GB 512GB SSD Windows 11 Home 15.6" FHD Azul Abyss*. Recuperado el 15 de octubre de 2025, de https://www.paris.cl/notebook-ideapad-slim-3-amd-ryzen7-5825u-16gb-512gb-ssd-windows-11-home-156-fhd-azul-abyss-106450999.html.

Paris.cl. (s. f.). *Notebook Gamer Acer Aspire G A515-58GM-56XX-1: Intel Core i5, NVIDIA RTX 2050, 16GB RAM, 512GB SSD, 15.6" FHD*. Recuperado el 15 de octubre de 2025, de https://www.paris.cl/notebook-gamer-aspire-g-a515-58gm-56xx-1-intel-core-i5-8-nucleos-nvidia-rtx-2050-16gb-ram-512gb-ssd-156-700915999.html.

HP. (2025). *Notebook HP Pavilion Plus 14-ew1002la*. Paris.cl. Recuperado el 27 de noviembre de 2025, de Notebook HP Pavilion Plus 14-EW1002LA Intel Ultra 7 32GB RAM 1TB SSD 14" 3K OLED Windows 11 Home HP | Paris.cl.

Glassdoor. (2025). *Sueldos para Jefe de Proyecto en Chile (Basado en datos de Claro Chile).* Recuperado el 26 de noviembre de 2025, de KEY_NOT_FOUND: ei-salaries.seo-metadata.el-details.title.singular | Glassdoor.

Glassdoor. (2025). *Sueldos para el puesto de Programador en Chile.* Recuperado el 26 de noviembre de 2025, de Sueldo: Programador (Noviembre, 2025) | Glassdoor.

Glassdoor. (2025). *Salario Mensual para Computer Hardware and Software en Latam (Chile).* Recuperado el 26 de noviembre de 2025, de KEY_NOT_FOUND: ei-salaries.seo-metadata.el-details.title.singular | Glassdoor.

Glassdoor. (2025). *Salario mensual para Analista y Documentador Técnico en TrackTec (Chile).* Recuperado el 26 de noviembre de 2025, de KEY_NOT_FOUND: ei-salaries.seo-metadata.el-details.title.singular | Glassdoor.

Bodega Digital. (2025). *Licencia Microsoft Office 2024 Professional Plus (Código Digital)*. Recuperado el 27 de noviembre de 2025, de Office 2024 Professional Plus • Bodega Digital.

BrickLink. (s.f.). *Part 3001: Brick 2 x 4*. Recuperado el 14 de diciembre de 2025, de https://www.bricklink.com/v2/catalog/catalogitem.page?P=3001

Wikipedia. (2024). *Acrilonitrilo butadieno estireno*. En Wikipedia, la enciclopedia libre. Recuperado el 15 de diciembre de 2025, de https://es.wikipedia.org/wiki/Acrilonitrilo_butadieno_estireno.

Hurbain, P. (2024). *LEGO Motor Characteristics: Comparative Test Data*. PhiloHome. Recuperado de https://www.philohome.com/motors/motorcomp.htm

Anexos

9.Anexo 1

Se adjunta fotografía de los precios reales de los productos (Hardware) mencionados en la tabla de la sección de costo de hardware.

1. SET LEGO SPIKE PRIME.

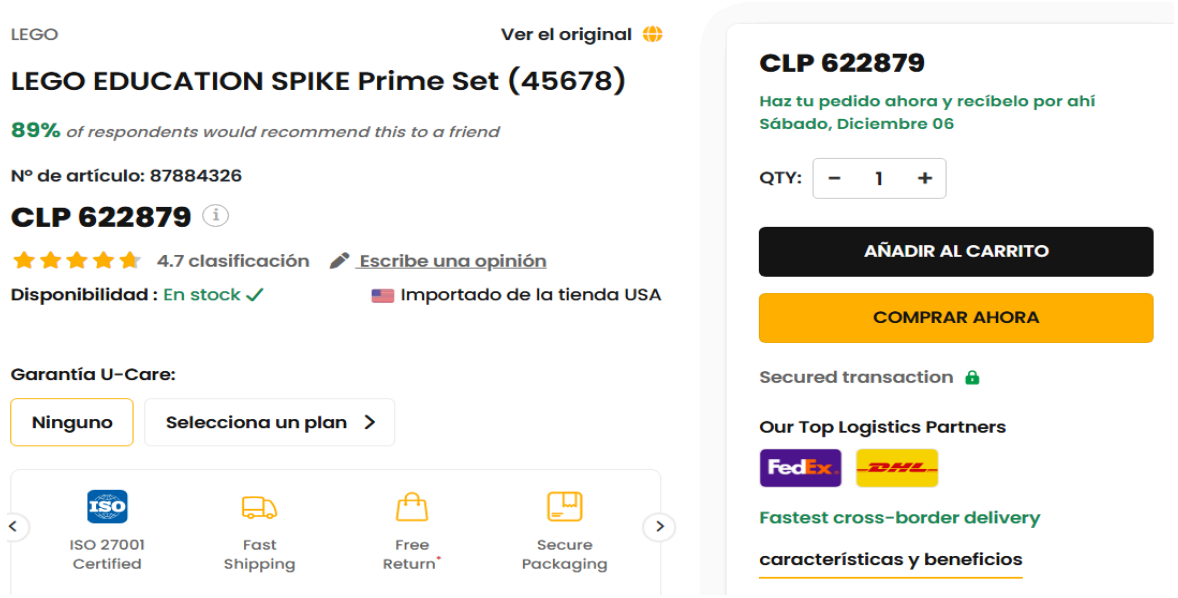


Ilustración N°12 "LEGO SPIKE PRIME".

2. EXTENSION LEGO SPIKE PRIME.

Ilustración N°13 "EXTENSION LEGO SPIKE PRIME"

3. NOTEBOOK LOQ GEN 9.

Ilustración N°14 "Notebook loq gen 9".

4. SAMSUNG GALAXY TAB S8 ULTRA.

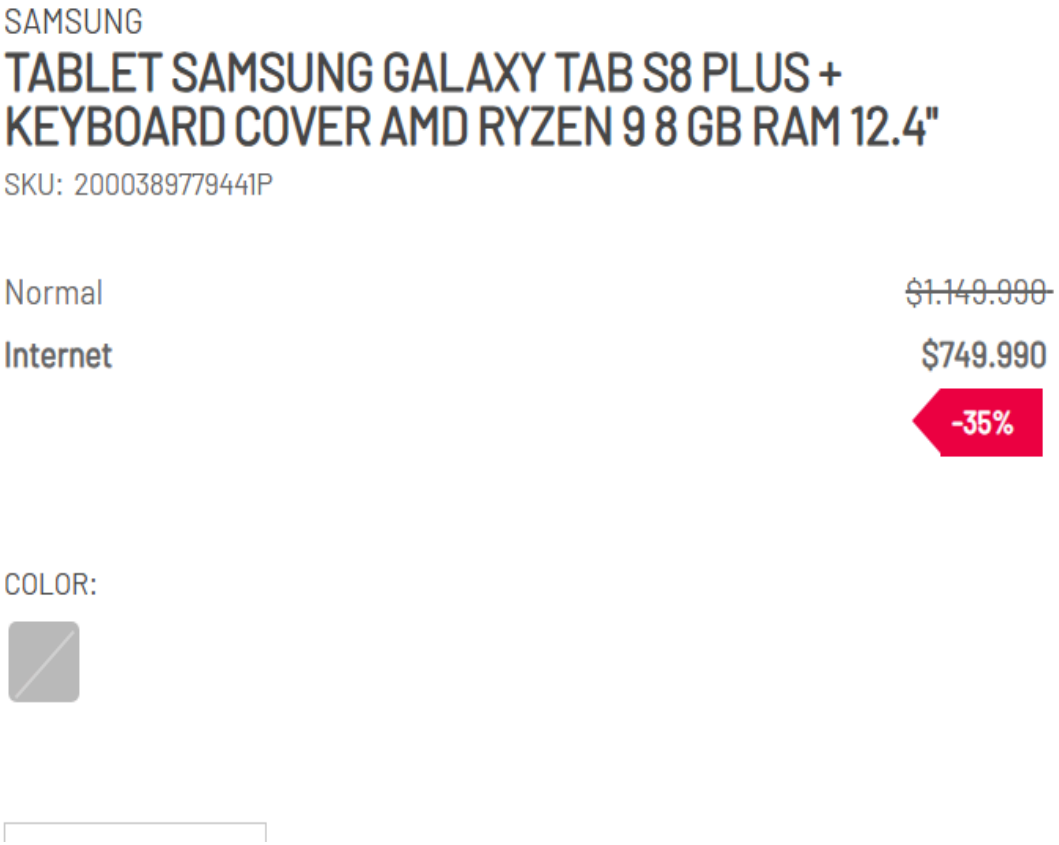


Ilustración N°15 "GALAXY TAB S8 PLUS".

5. LENOVO IDEAPAD 3 15.

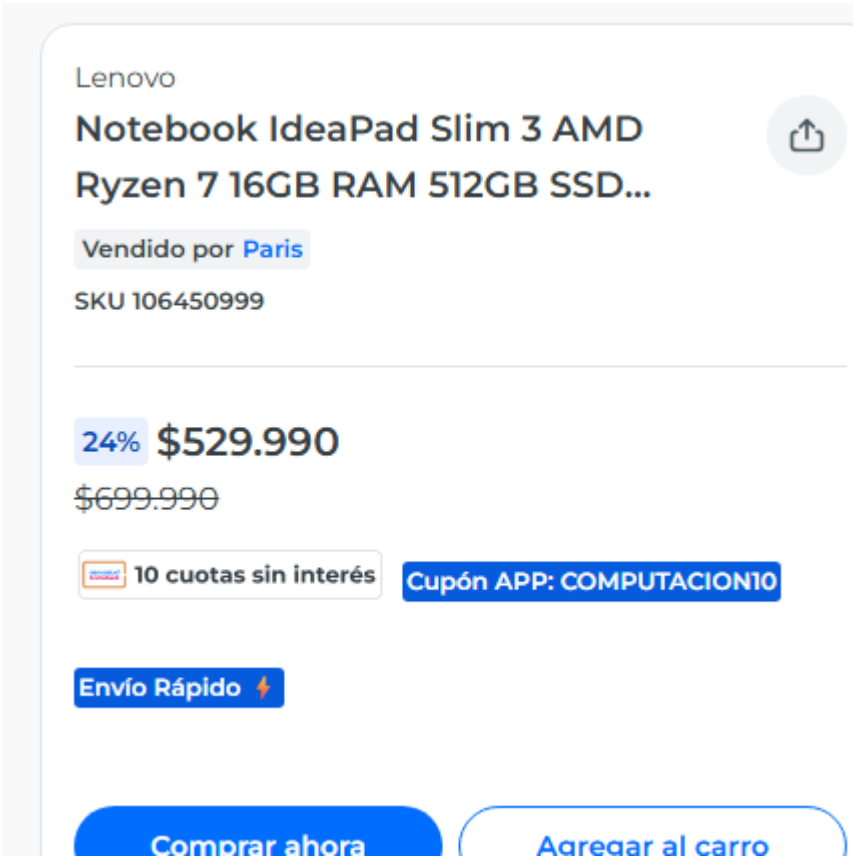


Ilustración N°16 "IDEAPAD 3 15".

6. NOTEBOOK ASPIRE G.

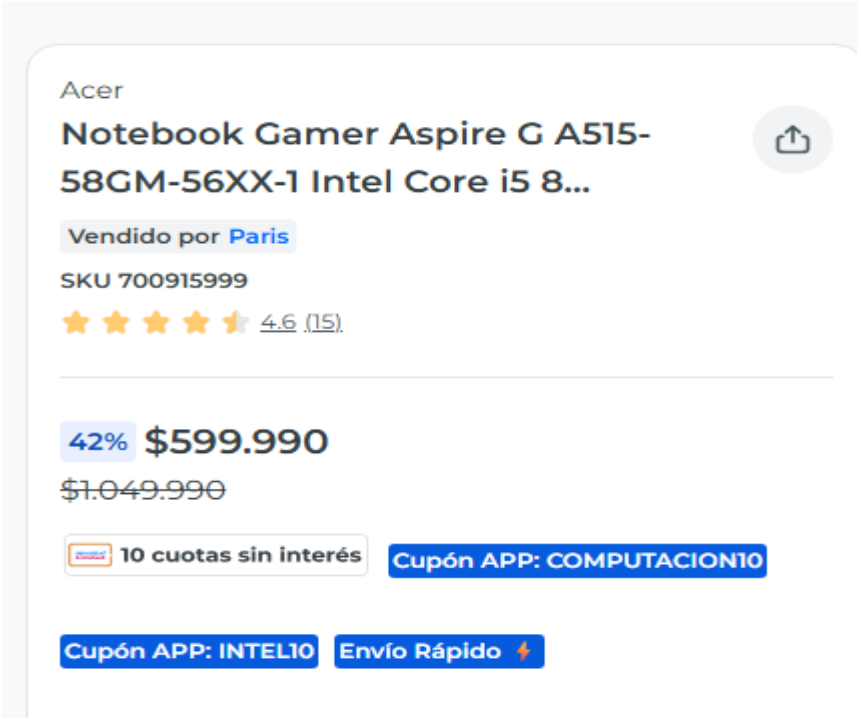


Ilustración N°17 “GAMER ASPIRE 5”.

7. HP PAVILION LAPTOP.

Ilustración N°18 “HP PAVILION”.

8. TAMAÑO DEL LEGO REAL PARA LA OBTENCIÓN DE DATOS APLICANDO LA FÍSICA

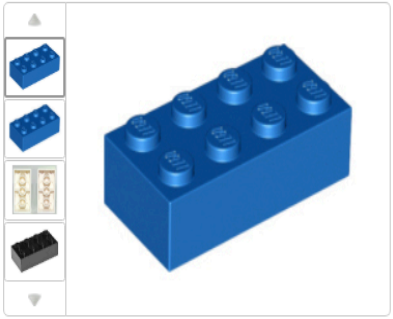
Catálogo: Piezas: Ladrillo: 3001

Ladrillo 2 x 4

Artículo nº: 3001 Artículo alternativo: 3001f1, 3556, 15589, 54534, 72841

Consulta la guía de precios | Compra

Seleccionar color



Información del artículo Años de lanzamiento: 1978 - 2025
Peso: 2,32g
Vigas Dim.: 2 x 4 x 1 pulgada Pack
. Dim.: 1,6 x 3,2 x 1,15 cm

El ítem consta de
N/A

El objeto aparece en
3929 Sets
, 19 Minifiguras
, 14 Partes
, 33 Libros,
17 Equipo

Inventario de mi tienda
Añadir a mi inventario
145534 lotes a la venta

Mi lista de buscados
Añadir a mi lista
de buscados en 1814872 listas de buscados

Mi colección
Añadir a mi colección
En 16532 Colecciones

Notas adicionales: Colapso ▲

Esta parte tiene una vertiginosa variedad de variantes, de las cuales solo unas pocas tienen entradas de catálogo separadas.

Elementos relacionados: Colapso ▲

Este artículo es una variante de molde del(los) siguiente(s) artículo(s):

- Parte 3001 especial de ladrillo 2 x 4 especiales (ladrillos especiales, ladrillos de prueba y/o prototipos)
- Parte bhol04 Ladrillo 2 x 4 sin tubos inferiores
- Parte 3001old Brick 2 x 4 sin soportes transversales
- Parte 3001oldb Ladrillo 2 x 4 sin soportes cruzados, con agujero en la parte superior

Ilustración N°19 "Bloques usados".