UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@ Computación e Informática

Informe Inicial
“Modelo escala LEGO - Vehículo Minero”

Alumnos : Francisca Albornoz
Brayan Cahuachia
Abraham Canaviri
Ruth Huanca
Cristofer Lazaro

Asignatura: Proyecto I

Profesor: Baris Klobertanz

DICIEMBRE - 2025
ARICA - CHILE

Historial de Cambios

Fecha	Versión	Descripción	Autores
26/09/2025	1.0	Formulación del Proyecto	Todo el equipo
01/10/2025	1.1	Recopilación de Información	Todo el equipo
01/10/2025	1.2	Planteamiento de Objetivos	Todo el equipo
06/10/2025	1.3	Distribución de Roles y planteamiento de Actividades	Todo el equipo
13/10/2025	1.4	Tabla costos de trabajador	Todo el equipo
17/10/2025	1.5	Versión preliminar del informe	Todo el equipo
24/11/2025	1.6	Modificación de las actividades del proyecto	Todo el equipo
26/11/2025	1.7	Creación de índice de tablas y figuras	Todo el equipo
28/11/2025	1.8	Corrección general del informe	Todo el equipo
10/12/2025	1.9	Agregar ítems del informe fase 2	Todo el equipo
15/12/2025	2.0	Concluir ítems informe fase 2	Todo el equipo
23/12/2025	2.1	Corrección del informe fase 2	Todo el equipo
30/12/2025	2.2	Ítem Prueba de funcionamiento del sistema y corrección general	Todo el equipo

Índice de Contenidos

Índice de Tablas

Índice de Figuras

.

1. Planteamientos del problema y objetivos

La minería representa el motor fundamental de la economía chilena y posiciona al país como líder mundial en la exportación de cobre y litio. Sin embargo, en la minería existen múltiples procesos en la extracción subterránea de minerales, el proyecto se basará en desarrollar un robot que replique el proceso de traslado, este será capaz de movilizarse con el material de carga que simulará los minerales extraídos, transportandolos de forma eficaz, asegurando la integridad del personal de trabajo y del material de carga de forma simulada.

El propósito del presente informe es documentar el diseño, la implementación técnica y la validación de este prototipo, demostrando que es posible realizar el transporte de carga de manera eficaz y segura, eliminando la necesidad de presencia humana en la zona de peligro mediante el uso de sistemas de control remoto.

1.1. Problema

El transporte minero es una de las tareas más peligrosas de la industria. El problema es que, al depender de conductores humanos, se les expone a riesgos constantes como derrumbes y accidentes por el cansancio. Esto afecta directamente la seguridad de los trabajadores y a la continuidad de las operaciones por un incidente.

Por esta razón, el desafío es encontrar la manera de salvaguardar a las personas de la zona de peligro sin detener la producción. Este proyecto busca simular un camión minero que no necesite un conductor a bordo. Validar el funcionamiento autónomo a escala conlleva un impacto enorme, ya que permitiría la continuidad de operaciones eliminando casi por completo la posibilidad de accidentes durante el traslado de carga.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar un modelo a escala de un vehículo minero utilizando el set LEGO SPIKE Prime, para simular el transporte de carga, evaluando su movilidad y control, con el fin de proponer una solución tecnológica que garantice la seguridad de los trabajadores frente a los desafíos del entorno subterráneo.

1.2.2. Objetivos Específicos

- Analizar las capacidades técnicas del set LEGO SPIKE Prime, identificando la configuración de motores más adecuada para simular la tracción y dirección de un vehículo minero.
- Investigar y aplicar las librerías de Python (como Pybricks y Bleak) que permitan establecer el protocolo de comunicación inalámbrica necesario para el control remoto del Hub.
- Diseñar e implementar la estructura mecánica del vehículo, asegurando una configuración robusta que optimice la estabilidad durante el transporte de la carga.
- Implementar una interfaz gráfica de usuario usando Tkinter que sea sencilla de usar para el usuario.

1.3. Restricciones

- Debe programarse con algún lenguaje de programación compatible con Lego Spike Prime.
- Se debe usar el set Lego Spike Prime.
- Plazo de entrega para el informe y el robot.
- El robot debe ser capaz de moverse y llevar rápida y eficientemente la carga.
- El control del robot debe realizarse de forma inalámbrica.
- Cantidad de integrantes limitada a un máximo de 5.
- Tiempo en el cual se puede utilizar el robot.
- Se debe usar Redmine para subir los documentos y la carta Gantt.
- Disponibilidad de las impresoras 3D en caso de ser necesaria la fabricación de una pieza.

1.4. Entregables

- Informe: el informe contendrá información detallada de los objetivos planteados, el contexto y alcance del trabajo, el desarrollo con las actividades y las dificultades encontradas, las acciones tomadas, las conclusiones y recomendaciones para la mejora, así como las referencias bibliográficas y anexos necesarios que respalden la documentación del proceso.
- Carta Gantt: es una representación visual de la planificación y seguimiento de un proyecto, en la que se muestran las actividades realizadas y programadas, también la fecha y la duración de las mismas, facilitando la gestión del tiempo y los recursos.
- Bitácoras: son informes semanales en los cuales se detallan los avances del robot en el cual trabajamos, abarcando los problemas y posibles soluciones que encontramos durante un lapso de tiempo determinado, además de mostrar el tiempo que se dedicara a cada actividad definida anteriormente.
- Manual de usuario: guía escrita en formato digital en donde se detallarán las instrucciones para el uso del robot a través de su interfaz gráfica, cómo usarlo de manera eficiente e instalación. Disponible en el repositorio de GitHub, como un archivo README.md.
- Presentación: se detallan las distribuciones del equipo y se ofrece una vista general del proyecto y del robot al igual se detallarán los objetivos, retos superados y sus soluciones, actividades definidas, implementaciones y una conclusión. Se apoya de un recurso visual.

2. Organización del Personal

La organización del personal en este proyecto se realizó considerando las habilidades, fortalezas y áreas de conocimiento de cada integrante del equipo. A cada miembro se le asignaron áreas de trabajo específicas en función de lo que podía aportar de manera más efectiva basándonos en los requerimientos del proyecto, con el propósito de maximizar la eficiencia del equipo y asegurar el cumplimiento de los objetivos del proyecto.

2.1. Descripción de los roles definidos

Jefe del Proyecto: es responsable de representar al grupo, coordinar las actividades del equipo y asegurarse de que cada miembro cumpla su rol. También supervisa el progreso del proyecto y facilita la comunicación entre los integrantes.

Documentador: se encarga de documentar los avances del grupo y redactarlo en los informes, además de los avances semanales redactados en las bitácoras.

Ensamblador: es el encargado de diseñar y dar la forma que tomará el robot para que cumpla con su propósito de forma eficiente.

Programador: se encargará de desarrollar el código y en conjunto con el ensamblador ver que los requerimientos del robot sean cumplidos.

2.2. Asignación de roles

Tabla 1: *Roles y responsables del proyecto*

Rol	Responsable
Jefe del Proyecto	Cristofer Lazaro
Documentador	Ruth Huanca
Ensamblador	Cristofer Lazaro
Programador	Brayan Cahuachia

2.3. Canales de comunicación

El principal medio de comunicación del equipo, por el momento, es la aplicación de mensajería WhatsApp, a través del cual se compartirá información relevante, ideas de diseño y líneas de código que contribuyan al desarrollo del proyecto. Asimismo, se utilizará este canal para notificar cualquier dificultad con los plazos de entrega o la imposibilidad de asistir a clases de algún integrante.

3. Planificación del Proyecto

Definición de las actividades necesarias para el desarrollo del proyecto.

3.1. Actividades definidas

Tabla 2: *Actividades definidas del proyecto*

Actividad	Responsables	Resultado
Organización de los roles en el proyecto.	Todo el grupo.	Se distribuyen los roles.
Experimentar con el Programa de LEGO Spike Prime.	Todo el grupo.	Una mejor comprensión de lo que ofrece el programa de LEGO spike prime.
Establecer una forma de manipular el robot a distancia.	Todo el grupo.	Investigar la forma en la cual el Spike se controlará de manera remota.
Prototipo del robot a escala.	Cristofer Lazaro Brayan Cahuachia	Prototipo de Modelo del robot a escala.
Avance del primer Informe.	Abraham Canaviri Ruth Huanca Francisca Albornoz	Avance del primer Informe.
Programación de movimientos bases (Avanzar, girar y retroceder).	Brayan Cahuachia Abraham Canaviri Ruth Huanca Cristofer Lazaro	Programación de movimientos bases (Avanzar, girar y retroceder).
Primer modelo del robot.	Cristofer Lazaro Brayan Cahuachia	Primer modelo del robot definitivo.
Implementación del mando del ps4.	Brayan Cahuachia	El mando es compatible con la PC y LEGO spike.

Cambio del método de programación de bloques a Python.	Abraham Canaviri	Se implementa un lenguaje de programación rápido y práctico para usarla con el robot.
Programación de la interfaz gráfica del control.	Abraham Canaviri Brayan Cahuachia Ruth Huanca	La interfaz es sencilla, práctica de comprender y usar.
Definir los elementos de la pista, sus obstáculos y su recorrido.	Todo el grupo.	Quedan planteados los obstáculos que se usarán, al igual que la distribución de la pista.
Impresión de los obstáculos 3D.	Cristofer Lazaro	Los obstáculos se imprimen sin problemas.
Primera práctica en la pista.	Abraham Canaviri Ruth Huanca Cristofer Lazaro	El auto no presenta fallas al momento de girar o frenar durante su recorrido en la pista.
Presentación.	Todo el grupo.	Primera presentación.
Manual de usuario	Abraham Canaviri	

3.2. Carta Gantt

La carta Gantt en un proyecto es un diagrama de barras utilizado para planificar y realizar un seguimiento de las actividades definidas, esto permite la coordinación del equipo.

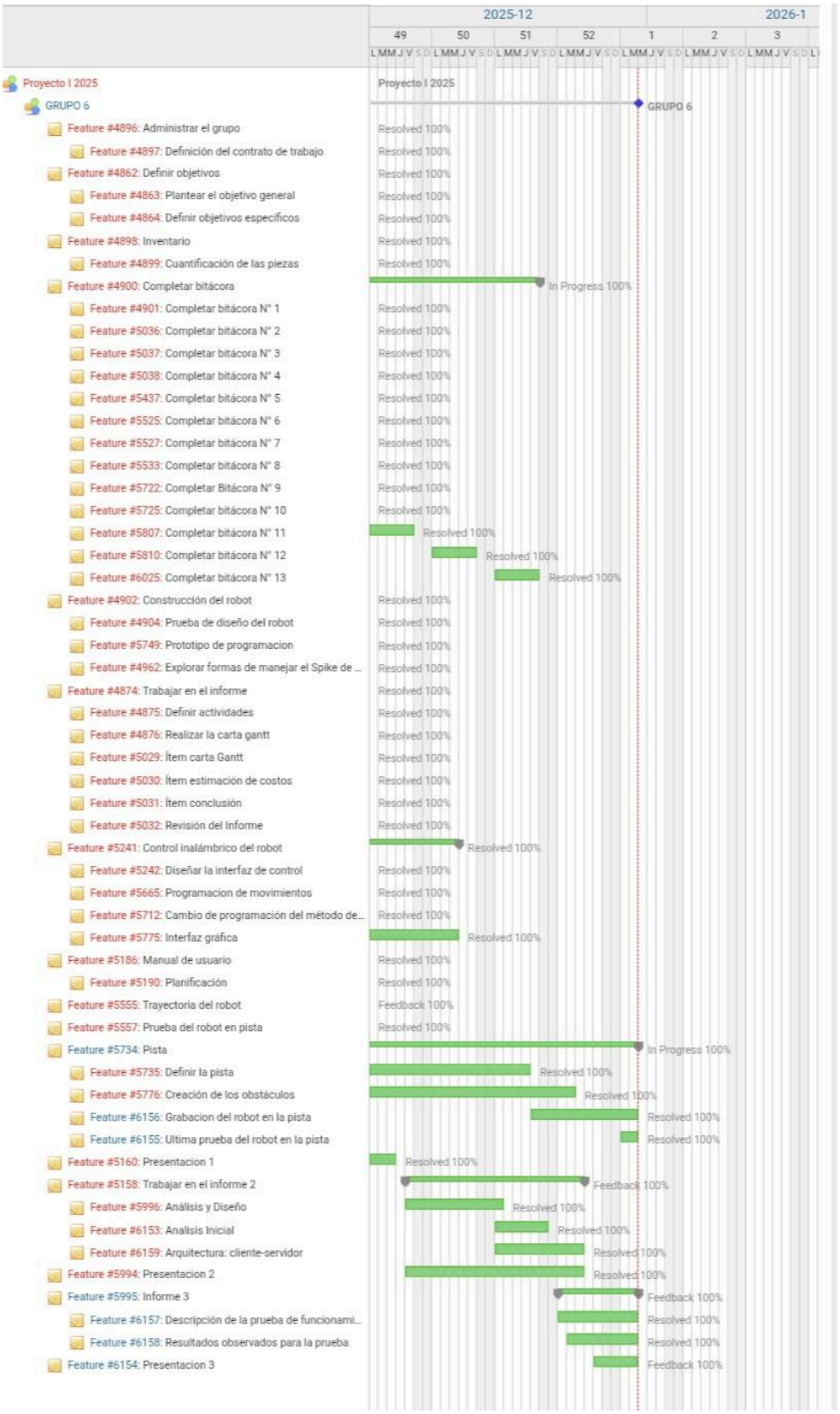

Figura 1: *Carta Gantt del proyecto*

3.3. Gestión de Riesgos

Cada riesgo conlleva un nivel de impacto al proyecto, en caso de que ocurran pueden provocar un efecto negativo al desempeño del proyecto. Los niveles de impacto se definen en función de tipos de daños, estos son:

1. Daño catastrófico: se deben resolver en el momento para evitar que el proyecto pueda colapsar o se detenga.
2. Daño crítico: se debe actuar rápidamente para evitar que el proyecto se pueda detener.
3. Daño circunstancial: se deben tomar medidas inmediatas para evitar que el proyecto pueda sufrir retrasos en sus distintas etapas.
4. Daño irrelevante: se pueden tomar medidas a largo plazo, ya que este no afecta de manera significativa al proyecto en sus distintas etapas.

Tabla 3: *Riesgos y acciones remediales del proyecto*

Riesgo	Nivel de Impacto	Acción remedial
Salida inesperada de un integrante del equipo.	1	Redistribuir las tareas críticas.
No poder acceder al set de LEGO Spike Prime.	2	Pedirlo a los ayudantes.
Falta de tiempo en alguna actividad importante.	2	Buscar otros horarios, para avanzar la actividad.
Falta de un elemento necesario para la construcción o diseño del modelo.	3	Pedir más piezas o el complemento del set lego.
Falta de algún integrante en una sesión.	3	Se pone en contacto con el resto de miembros para saber cuáles son las actividades avanzadas y en cuáles trabajar.
Falla del robot al momento de ejecutar un movimiento.	4	Revisar si el problema proviene del control a distancia o del código.
Necesidad de alguna pieza extra.	4	Pedir imprimirla en la impresora 3D.

4. Identificación de los recursos y costos asociados

La planificación de recursos es la identificación y gestión de los recursos que el proyecto, durante su ejecución, demanda para cumplir los objetivos definidos de manera eficiente. En este caso, se han dividido estos recursos en hardware y software, ambos, necesarios para cada rol del equipo.

4.1. Hardware

El hardware se refiere a los componentes físicos utilizados en el proyecto, este es necesario para desarrollar el robot y los entregables.

- Set de Lego Spike Prime: es un set de robótica que posee piezas de construcción Lego, una unidad de control programable e incluye un conjunto de sensores y motores, estos son utilizados para el desarrollo del robot.
- Computadores: es necesario para el acceso a los diferentes programas que se utilizarán, además de permitir el trabajo colaborativo del equipo.
- Manilla de PS4: durante la primera fase del proyecto, será necesario el uso de este recurso para controlar al robot a distancia, por lo que este debe ser compatible con un computador y el robot.

4.2. Software

Se considera el software como un conjunto de programas que serán utilizados para determinadas actividades, definidas en la carta Gantt, que lo requieran para su debida ejecución.

- Redmine: aplicación web destinada a la gestión de proyectos, utilizada para el seguimiento de las actividades por medio de una carta Gantt integrada y subir las bitácoras correspondientes a cada semana de trabajo.
- Lego Education Spike Prime: aplicación de escritorio para la programación en MicroPython del set Lego Spike Prime, empleada para la introducción al set y el uso de recursos disponibles en esta.
- Documentos de Google: procesador de texto que permite la creación y edición de documentos, utilizado para la redacción de los informes y bitácoras del proyecto.
- Visual Studio Code: entorno de desarrollo integrado o editor de código, se utiliza para el desarrollo e implementación del código MicroPython de Pybricks.

- Pybricks: implementación del firmware MicroPython en el hub de Lego. Su función es ejecutar la lógica de movimiento y secuencia del robot, además se encarga de gestionar la comunicación BLE (Bluetooth Low Energy) para recibir instrucciones.
- Canva: plataforma que permite la creación de contenido visual. Se utiliza para la creación de las presentaciones asociadas a cada fase del proyecto.

4.3. Recursos humanos

Se presentan las tablas **Tabla 4**, **Tabla 5**, **Tabla 6**, **Tabla 7** con los costos asociados al desarrollo del proyecto, incluyendo el hardware, el software y el trabajador, correspondiente a los roles del equipo y una tabla con el total de costos.

Costo de Hardware:

Tabla 4: *Costos de hardware*

Producto	Cantidad de ítems	Precio (CLP)
Set Lego® Spike Prime de Education.	1	$ 385.500 (1)
Lenovo V14 G4 IRU.	1	$832.990 (2)
IdeaPad Gaming 3 15IMH05.	1	$479.990 (3)
Asus TUF Gaming F16.	1	$679.990 (4)
Tablet Samsung Galaxy A9.	1	$99.990 (5)
Mando PS4.	1	$19.990 (6)
Total:	6	$2.498.450

Costo de Software:

Tabla 5: *Costos de software*

Producto	Precio (CLP)
LEGO Education Spike App v.3.5.1	$0 (7)
Visual Studio Code	$0 (8)
Documentos de Google	$0 (9)
Pybricks	$0 (10)
Redmine	$0 (11)
Total:	$0

Costo de Trabajador:

Tabla 6: *Costo de trabajador*

Rol	Horas de Trabajo	Horas Extras	Precio/Hora (CLP)
Jefe del Grupo	54 horas	—	$10.887 (12)
Documentador	54 horas	—	$6.331 (13)
Ensamblador	54 horas	—	$6.875 (14)
Programador	54 horas	3 horas	$6.124 (15)
Total:	216	3	$1.650.090

Destacado:

- *Las horas de trabajo son contabilizadas desde la primera clase con el robot, semana del 29 de septiembre hasta la semana del 15 de diciembre (total de 12 semanas). Se tienen 3 clases de 1 hora y 30 minutos cada semana, por lo tanto las horas totales de cada rol se definen como:*

  *12 semanas * 4,5 horas = 54 horas.*

- *Las horas extras se consideran según las horas trabajadas fuera de clases, estas pueden diferir del tiempo real, pues es complicado recordar guardar y anotar estas horas, es decir, contabilizarlas.*

- *Para el cálculo del Precio/Hora: se observa que los sueldos son mensuales en los sitios consultados, por lo que se hace el siguiente cálculo para obtener el Precio/Hora. Al ser el sueldo una valor externo, se considera que este corresponde a un estándar de 160 horas por mes (40 horas/semana * 4 semanas). Finalmente:*

  Precio/Hora = sueldo referencia/ horas por mes estándar.

- *Para calcular el costo total de los sueldos de cada rol se realiza:*

  *Costo total del rol = Precio/hora * Horas de Trabajo (se considera el mismo precio del Precio/Hora para las horas extras).*

  Con esto se puede obtener el total del costo a invertir únicamente en el pago de sueldos, el cual se calcula como la suma de estos:

  Jefe del Grupo: $587.898 CLP.

  Documentador: $341.874 CLP.

  Ensamblador: $371.250 CLP.

  Programador: $349.068 CLP.

Total de costos:

Tabla 7: *Total costos*

Costo hardware	Costo software	Costo empleados	Total (CLP)
$2.498.450	$0	$1.650.090	$4.148.540

5. Análisis y diseño

En este apartado se verán parte de sus requerimientos fundamentales, no fundamentales al igual que un diseño de la interfaz gráfica y se ilustrara como se comunican entre sí los componentes.

5.1. Especificación de requerimientos

Para establecer los requerimientos del sistema, es fundamental comprender el problema: la necesidad de retirar a los operadores humanos de las zonas de riesgo en la minería subterránea sin detener la producción. La solución consiste en un robot minero controlado remotamente.

Usuario: corresponde al Operador de Sala de Control. Es la persona encargada de manipular la interfaz gráfica (GUI) para guiar el vehículo, monitorear su estado y ejecutar las acciones de transporte. Este usuario requiere un sistema intuitivo que no demande conocimientos de programación para ser operado.

Cliente: corresponde a la compañía minera. Es la entidad que solicita y financia el desarrollo del proyecto, además decide si la solución propuesta satisface adecuadamente el problema.

5.1.1. Requerimientos funcionales

Los requerimientos funcionales del proyecto son las especificaciones que se deben cumplir para que el robot ejecute su función principal, en base a la solución propuesta y las personas interesadas e involucradas en el proyecto.

Los requerimientos funcionales definidos son:

- RF1: El robot debe ser capaz de ejecutar cuatro movimientos básicos: avanzar, retroceder, girar a la izquierda y girar a la derecha.
- RF2: El sistema debe permitir al usuario iniciar una conexión inalámbrica entre el cliente y el servidor.
- RF3: El sistema debe permitir al usuario finalizar la conexión inalámbrica de manera manual.
- RF4: El sistema debe mostrar visualmente al usuario el estado actual de la conexión (conectando, conectado o desconectado).

- RF5: La GUI debe mostrar un historial histórico de las acciones o comandos ejecutados.
- RF6: La Interfaz Gráfica de Usuario (GUI) debe mostrar los controles de dirección disponibles para operar el robot.
- RF7: La GUI debe enviar una orden de realineación de las ruedas a su posición central.
- RF8: El sistema debe ser capaz de reubicar el sistema de dirección a su ángulo cero al recibir la orden de realineación.

5.1.2. Requerimientos no funcionales

Basándose en una lista de atributos de calidad (Wikipedia contributors, 2025) (16), los requerimientos no funcionales que se han requerido para este proyecto son:

- RNF1: La interfaz gráfica debe ser lo suficientemente intuitiva para que un nuevo usuario logre operar el robot con un máximo de 2 intentos fallidos en su primer uso.
- RNF2: El sistema debe garantizar una disponibilidad operativa del 95% durante una sesión de uso continuo de 10 minutos, siempre que el Hub cuente con nivel de batería suficiente y se mantenga dentro del rango de cobertura Bluetooth.
- RNF3: El tiempo de respuesta entre la acción del usuario (clic en la interfaz) y la reacción del robot debe ser inferior a 1 segundo, minimizando el retraso (lag) para permitir un control preciso en tiempo real.
- RNF4: La conexión entre cliente y servidor debe realizarse utilizando el protocolo Bluetooth Low Energy (BLE).
- RNF5: El robot debe utilizar los motores de tracción para realizar las acciones de RF1.

5.2. Arquitectura de software

Definir una arquitectura clara es fundamental para que el sistema sea ordenado y fácil de mantener en el tiempo. No se trata solo de conectar cables, sino de asegurar ciertos atributos de calidad como la estabilidad, la baja latencia y la facilidad para realizar mejoras futuras.

Se eligió el modelo Cliente-Servidor con comunicación por Flujo de Datos (Streaming). La razón principal de esta elección es la separación de responsabilidades y la necesidad de respuesta en tiempo real: la interfaz gráfica (PC) gestiona la lógica de control, mientras que el robot se dedica exclusivamente a la actuación motora inmediata.

Si necesitamos mejorar el diseño de la interfaz o agregar botones nuevos, podemos hacerlo sin detener ni reprogramar el firmware del robot. Esto garantiza un desarrollo más seguro y modular.

- Cliente: Es el componente de alto nivel ejecutado en el computador. Desarrollado en Python con la librería CustomTkinter (para una UI moderna) y Asyncio, actúa como el cerebro del sistema. Su arquitectura interna separa la interfaz visual del procesamiento lógico mediante hilos (threads), lo que permite capturar eventos de teclado y ratón sin congelar la ventana, transformándolos en caracteres de control instantáneos.

- Comunicación: Es el canal de enlace entre cliente y servidor. Se utiliza el protocolo BLE gestionado por las librerías de Pybricks, específicamente aprovechando el servicio de UART (Nordic UART Service). A diferencia de enfoques tradicionales de transferencia de archivos, este canal funciona como un flujo continuo (stream) bidireccional: permite inyectar caracteres ("F", "B", "S") en tiempo real hacia el robot y recibir logs de depuración simultáneamente.

- Servidor: El Hub del robot opera como un servidor de ejecución continua. Al iniciar, se carga en él un script "Gateway" residente (HUB_GATEWAY_CODE). Este programa mantiene un bucle infinito que monitorea la entrada estándar (stdin) cada 10ms. Al detectar un carácter en el buffer de entrada, ejecuta la instrucción motora correspondiente de inmediato, eliminando los tiempos de compilación y carga por cada movimiento.

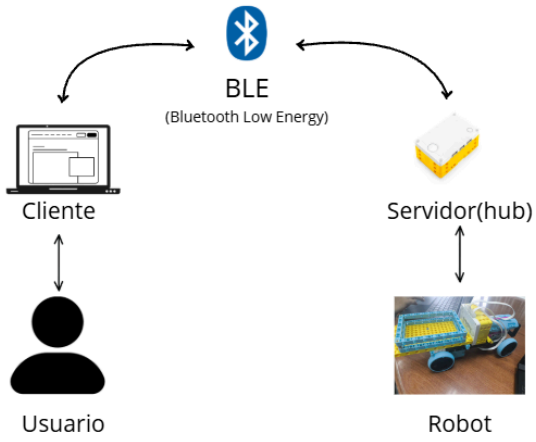

Figura 2: *Diagrama de la arquitectura Cliente-Servidor*

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

En la **Figura 2** se encuentra el bosquejo de la interfaz gráfica de usuario, es decir, el wireframe asociado al proyecto.

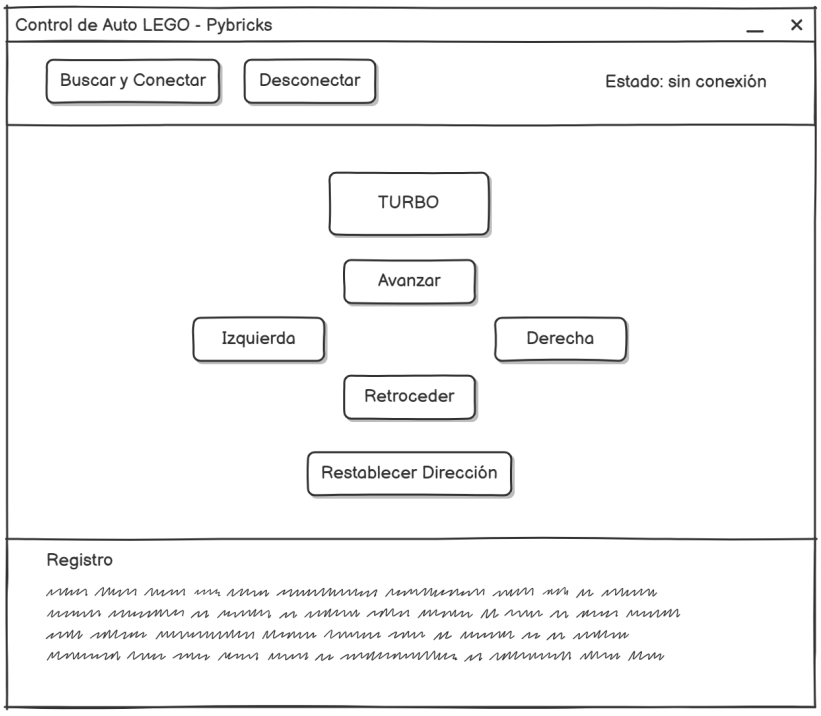

Figura 3: *Wireframe de baja fidelidad del GUI*

6. Implementación

6.1. Fundamentos de los movimientos

En esta sección se justifica la configuración seleccionada para el robot utilizando conceptos fundamentales de física, tal como se estudia en la asignatura FI 035.

Para determinar la magnitud relevante del funcionamiento del vehículo, se ha planteado el siguiente objetivo:

- "Determinar la aceleración media necesaria para que el vehículo recorra una distancia de 1 metro en el menor tiempo posible."

Para este cálculo, se establece un tiempo meta de 2.0 segundos. Además, se asume un modelo ideal donde se desprecian las fuerzas de roce (fricción) con la superficie y la resistencia del aire, considerando que la fuerza neta del motor se traduce íntegramente en movimiento.

6.1.1. Identificación de variables.

Distancia a recorrer (d): $1\ m$.
Tiempo objetivo (t): $2.0\ s$.
Velocidad inicial (v_0): $0\ m/s$ (el robot parte del reposo).
Incógnita: Aceleración media (a).

6.1.2. Cálculo de la aceleración.

Se utiliza la ecuación de itinerario del Movimiento Rectilíneo Uniformemente Acelerado (MRUA):

$$d = v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2$$

Dado que la velocidad inicial es cero ($v_0 = 0$), la ecuación se simplifica a:

$$d = \frac{1}{2} \cdot a \cdot t^2$$

Despejando la aceleración:

$$a = \frac{2d}{t^2}$$

Sustituimos lo valores definidos:

$$a = \frac{2 \cdot 1m}{(2s)^2} = 0.5\, m/s^2$$

Conclusión:
Bajo las condiciones ideales planteadas (sin considerar pérdidas por roce), el cálculo determina que el vehículo debe mantener una aceleración media constante de $0.5\, m/s^2$ para cubrir la distancia en el tiempo objetivo.

6.2. Descripción del sistema

El sistema se ha implementado utilizando el lenguaje Python 3.12, integrando librerías modernas para la gestión de interfaces gráficas y comunicaciones asíncronas. La arquitectura se basa en un script maestro que gestiona la interacción con el usuario y, simultáneamente, genera y transmite micro-programas al robot en tiempo real.

Repositorio del Proyecto: El código fuente completo del sistema, incluyendo el control de versiones y las iteraciones previas se encuentran en el repositorio del proyecto (AbrahamCode24, s.f.) (17).

6.2.1. Cliente

1-Ventana de Selección (DeviceSelectWindow): Este módulo actúa como el punto de entrada al sistema. Su función es gestionar el descubrimiento de hardware antes de permitir el acceso al panel de control.

- Escaneo y Filtrado: Utiliza la librería BleakScanner en un hilo secundario para buscar dispositivos Bluetooth Low Energy (BLE) en el entorno. Implementa un filtro lógico que descarta dispositivos sin nombre o señales débiles, presentando al usuario únicamente los Hubs LEGO viables para la conexión.

- Gestión de Eventos: Al seleccionar un dispositivo de la lista, esta ventana retorna el objeto de conexión (device) a la interfaz padre y se autodestruye para liberar recursos.

```python
class DeviceSelectWindow(ctk.CTkToplevel):
    def __init__(self, parent, callback):
        super().__init__(parent)
        self.callback = callback
        self.title("Buscar HUB")
        self.geometry("400x450")
        self.attributes("-topmost", True)
        self.grab_set()

        self.scroll = ctk.CTkScrollableFrame(self)
        self.scroll.pack(expand=True, fill="both", padx=10, pady=10)

        threading.Thread(target=self._scan, daemon=True).start()

    def _scan(self):
        loop = asyncio.new_event_loop()
        devices = loop.run_until_complete(BleakScanner.discover(timeout=3))
        loop.close()
        self.after(0, lambda: self._show(devices))

    def _show(self, devices):
        for d in devices:
            if d.name:
                ctk.CTkButton(
                    self.scroll,
                    text=d.name,
                    command=lambda dev=d: self._select(dev)
                ).pack(pady=5, fill="x")

    def _select(self, device):
        self.callback(device)
        self.destroy()
```

Figura 4: *Clase DeviceSelectWindow*

2-Interfaz Principal (LegoNitroGUI) Es el núcleo operativo del cliente (main). Desarrollada con CustomTkinter, centraliza los comandos de navegación y la visualización del estado del robot. Esta clase encapsula la lógica de concurrencia y gestión de estados:

- Gestión de Concurrencia Interna (Worker): Para evitar que la interfaz gráfica se congele durante la comunicación Bluetooth, la clase LegoNitroGUI inicializa internamente un gestor de tareas asíncronas (BLEWorker). Este gestor opera en un hilo paralelo (daemon thread) y utiliza una Cola FIFO (Queue) para procesar las órdenes. Cuando el usuario interactúa con la interfaz, la GUI no envía el dato directamente, sino que lo "encola", permitiendo que el hilo secundario maneje la transmisión sin bloquear la respuesta visual.

```python
class LegoNitroGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Control de Camión Minero - LEGO SPIKE")
        self.root.geometry("600x600")

        # Variables de estado
        self.keys_pressed = set()
        self.log_queue = Queue()

        # Iniciar Worker
        self.worker = BLEWorker(self.log_queue)
        self.worker.start()

        # Construir Interfaz
        self._build_ui()
```

Figura 5: *Clase LegoNitroGUI.*

- Captura de Entradas Híbrida: El sistema implementa dos métodos de control simultáneos:

  Botones Virtuales: Reconfigurados como interruptores momentáneos. Envían el comando de activación al presionar (Press) y el de parada al soltar (Release).

  Teclado: Se han integrado "listeners" (<KeyPress> y <KeyRelease>) que mapean las flechas direccionales y la tecla espacio a los comandos del robot, permitiendo una conducción fluida y ergonómica.

- Lógica de Seguridad (Interlock): La interfaz gestiona los estados de los controles mediante el método set_controls_enabled (implícito en la lógica de conexión). Los comandos de tracción permanecen inactivos hasta que el hilo interno confirma el "handshake" exitoso con el Hub, previniendo el envío de datos al vacío.

6.2.2. Servidor

En esta arquitectura, el "Servidor" es el Hub LEGO SPIKE Prime ejecutando el firmware de Pybricks.

1. Lógica de Escucha Activa (HUB_GATEWAY_CODE), se implementa un bucle de eventos infinito (while True) alojado en la memoria del robot.

Polling (Sondeo): Utilizando la librería uselect y usys, el servidor monitorea el buffer de entrada estándar (stdin) cada 10 milisegundos (poll.poll(10)).

Decodificación: Al detectar la llegada de datos, el sistema lee un solo carácter (byte). Este carácter actúa como una "llave" que activa bloques de código pre-cargados (ej: al recibir 'F', ejecuta el bloque de avance; al recibir 'S', corta la energía).

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

hub = PrimeHub()
motorB = Motor(Port.B)
motorF = Motor(Port.F)
motorD = Motor(Port.D)

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

hub.display.char('G')

while True:
    if poll.poll(10):
        cmd = usys.stdin.read(1)
        if cmd == 'F':
            motorB.run(-800)
            motorF.run(800)
        elif cmd == 'T':
            motorB.run(-1100)
            motorF.run(1100)
```

Figura 6: *Parte del método HUB_GATEWAY_CODE.*

2. Control de Actuadores y Concurrencia La lógica de control de motores se ha optimizado para permitir movimientos compuestos.

Tracción (Motores B y F): Se activan indefinidamente al recibir el comando de inicio y solo se detienen al recibir explícitamente el comando de parada ('S' o 'X'). Esto transfiere la responsabilidad de la duración del movimiento al usuario (cuánto tiempo mantiene la tecla presionada).

Dirección Asíncrona (Motor D): Se utiliza el parámetro wait=False en los comandos de viraje (run_target). Esto es crítico: permite que el motor de dirección busque su ángulo objetivo (-25° o 25°) en segundo plano sin bloquear el procesador, permitiendo que el robot siga procesando comandos de aceleración mientras gira.

```
while True:
    if poll.poll(10):
        cmd = usys.stdin.read(1)
        if cmd == 'F':
            motorB.run(-800)
            motorF.run(800)
        elif cmd == 'T':
            motorB.run(-1100)
            motorF.run(1100)
        elif cmd == 'B':
            motorB.run(400)
            motorF.run(-400)
        elif cmd == 'L':
            motorD.run_target(500, -25, wait=False)
        elif cmd == 'R':
            motorD.run_target(500, 25, wait=False)
        elif cmd == 'C':
            motorD.run_target(500, 0, wait=True)
            motorD.stop()
        elif cmd == 'S':
            motorB.stop()
            motorF.stop()
        elif cmd == 'X':
            motorB.stop()
            motorF.stop()
            motorD.stop()
    wait(10)
```

Figura 7: *Lógica de despacho (Dispatcher) del Hub.*

3. Mecanismo de transmisión: Esta es la etapa crítica de comunicación que enlaza el entorno lógico con el físico. A diferencia de un control remoto simple que envía una señal de radio, este sistema opera mediante la inyección de código y se divide en 2 fases.

- Fase 1: Despliegue (Inyección de Código): Al iniciar la conexión, el PC crea un archivo temporal una única vez. Utilizando el protocolo estándar de Pybricks, transfiere el script HUB_GATEWAY_CODE completo a la memoria RAM del Hub y lo ejecuta.
- Fase 2: Streaming de Comandos (Operación): Una vez el script está corriendo, el canal Bluetooth cambia a modo de transmisión de datos (UART).

  Envío (PC): La función hub.write(cmd.encode()) convierte el carácter de comando (ej: "T") en bytes binarios y los inyecta directamente en el canal Nordic UART Service (NUS).

  Recepción (Hub): El firmware de Pybricks redirige estos bytes entrantes desde el módulo Bluetooth hacia el flujo de entrada del sistema (sys.stdin), donde el bucle de escucha los captura instantáneamente.

```
temp_path = None
while True:
    await self._connect_request.wait()
    try:
        self.log(f"Conectando a {self._target_device.name}...")
        self.hub = PybricksHubBLE(self._target_device)
        await self.hub.connect()

        with tempfile.NamedTemporaryFile(mode='w', suffix='.py', delete=False) as tf:
            tf.write(HUB_GATEWAY_CODE)
            temp_path = tf.name

        self.queue = asyncio.Queue()
        asyncio.create_task(self.hub.run(temp_path))

        self.running.set()
        self.log("¡LISTO! Control habilitado.")

        while self.running.is_set():
            cmd = await self.queue.get()
            await self.hub.write(cmd.encode())
    except Exception as e:
        self.log(f"Error: {e}")
    finally:
        if temp_path and os.path.exists(temp_path):
            os.unlink(temp_path)
        if self.hub:
            await self.hub.disconnect()
        self.running.clear()
        self._connect_request.clear()
        self.log("Hub Desconectado.")
```

Figura 8: *Transmisión de datos en tiempo real.*

6.2.3. Interfaz gráfica de usuario (GUI)

La interfaz de operación ha sido desarrollada utilizando la librería CustomTkinter, la cual proporciona controles modernos de alto contraste (modo oscuro). Este diseño fue seleccionado para mejorar la legibilidad y reducir la fatiga visual del operador durante las sesiones de prueba. La ventana principal organiza los controles en tres zonas lógicas: gestión de conexión, mando de navegación y registro de eventos.

A continuación, se presenta una captura de la interfaz implementada, seguida de la descripción técnica de sus componentes:

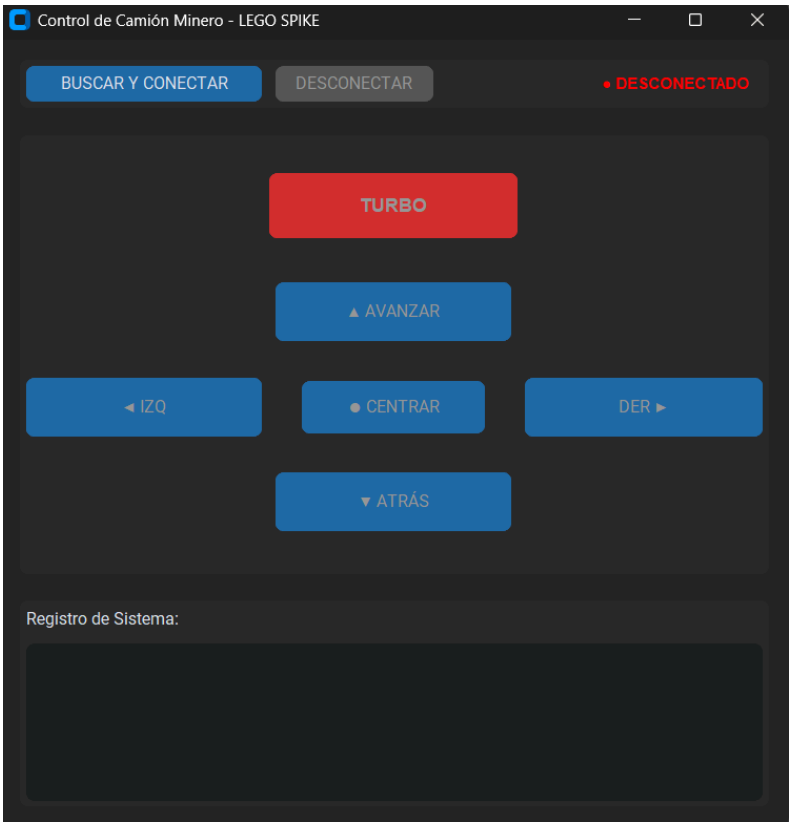

Figura 9: *Interfaz gráfica de usuario*

Descripción de Componentes:

1. Panel Superior de Conexión: ubicado en la parte superior de la ventana, es el encargado de gestionar el enlace inalámbrico con el robot.

Botón "Buscar y Conectar": al ser presionado, despliega una ventana emergente (DeviceSelectWindow) que escanea el entorno en busca de dispositivos Bluetooth Low Energy (BLE). Incorpora un filtro que muestra únicamente los Hubs LEGO activos, facilitando la identificación del equipo en entornos con interferencia.
Botón "Desconectar":permite cerrar la sesión de control de forma segura, enviando una señal de término al hilo de comunicación (BLEWorker) para liberar el canal Bluetooth y detener el robot.

Etiqueta de Estado: un indicador de texto dinámico que informa al operador sobre la situación actual del enlace ("Conectando...", "Conectado", "Desconectado"), proporcionando retroalimentación inmediata sobre la disponibilidad del sistema.

2. Panel Central de Navegación: es el núcleo operativo del sistema. Contiene botones virtuales para el control de movimiento. Este panel cuenta con una lógica de seguridad (Safety Interlock) que mantiene todos los controles bloqueados (en estado deshabilitado) hasta que se confirma una conexión exitosa.

Botón TURBO (Rojo): es el control de mayor jerarquía visual y funcional. Configurado con un color de alerta rojo permite activar el modo de alta velocidad (1100) para maniobras de desplazamiento rápido o para superar obstáculos por inercia.

Botones de Desplazamiento (Avanzar/Retroceder): envían comandos de movimiento longitudinal a velocidad estándar (800 RPM) con una duración controlada, ideal para una navegación precisa.

Botones de Viraje (Izquierda/Derecha): controlan el sistema de dirección, enviando instrucciones al motor para girar las ruedas a un ángulo fijo de 25 grados.

Botón "Restablecer Dirección": una función crítica de mantenimiento que envía el comando para realinear las ruedas a su posición central (0 grados), corrigiendo cualquier desviación mecánica en la trayectoria.

3. Panel Inferior de Registro (Log):

Área de Texto de Registro: un cuadro de texto de solo lectura que actúa como la "caja negra" del sistema. Muestra en tiempo real los mensajes de depuración, confirmando la ejecución de cada comando enviado (ej: "Ejecutado: run_forward") o alertando sobre posibles errores de comunicación. Esto permite al operador verificar que sus órdenes están siendo recibidas y procesadas correctamente por el robot.

7. Resultados

En este apartado se mostrará el estado actual del proyecto, es decir, el avance desarrollado con respecto a los requerimientos funcionales, requerimientos no funcionales y los entregables. Además, de los problemas que hasta ahora se han encontrado y solucionado, incluyendo el riesgo asociado.

7.1. Estado actual del proyecto

Con los entregables de la primera fase finalizados, el proyecto ha cumplido con parte de las funcionalidades definidas para el robot. A continuación, se describen los avances con respecto a los requerimientos funcionales:

- Diseño de modelo del robot:

  El diseño del robot se ha concluido. Se considera la posibilidad de cambios derivados de la prueba en pista con los obstáculos; sin embargo, estas modificaciones son mínimas y no afectarán de manera significativa al desarrollo del proyecto.

- Control:

  En relación con el RF1, los movimientos del robot se ejecutan a través de la interfaz de usuario. Actualmente se están realizando pruebas de control y de respuesta del hub.

- Gestión de Conexión Inalámbrica:

  Considerando los RF2 y RF3, la conexión con el robot se realiza correctamente de manera inalámbrica, sin requerir una conexión física. En consecuencia, se puede implementar el RF4 en la GUI.

- Interfaz de usuario (GUI):

  Los RF4, RF5, RF6 Y RF7 se vinculan con la interfaz gráfica de usuario. La GUI actúa como el método de interacción entre el usuario y el robot, sin necesidad de poseer conocimientos previos de programación. En la fase actual, la interfaz es funcional y cumple con los RF asociados para realizar las pruebas con el robot. Se considera la opción de implementar un diseño más elaborado.

- Restablecimiento de dirección:

  El RF8 se ha cumplido mediante la implementación de una función en el código, lo que permite la incorporación del botón que ejecuta la función de realineación de ruedas, correspondiente al RF7. Actualmente se están realizando pruebas del comportamiento del robot al ejecutar la función.

Respecto a los RNF definidos, estos se definen de tal manera que sea posible probarlo en la fase práctica del proyecto.

Los entregables del proyecto se han ido completando sin dificultades, dado que, al tratarse de un trabajo semanal, tanto la bitácora como la carta Gantt se desarrollan en periodos de tiempo acotados, con la participación del equipo. Asimismo, el informe se elabora fuera del horario de clases de la asignatura, lo que permite al equipo avanzar en su desarrollo en distintos momentos.

7.2. Problemas encontrados y solucionados

Durante la segunda fase del proyecto se han encontrado problemas que afectan de manera relevante al desarrollo y avance del proyecto, por lo que se han implementado soluciones. Además, cada problema se ha clasificado en relación con los riesgos previamente identificados en la gestión de riesgos. En la **Tabla 8** se mencionan dichos problemas.

Tabla 8: *Problemas encontrados, soluciones y riesgos asociados*

Problema Encontrado	Solución	Riesgo asociado
Comunicación para la creación de la pista	Hablar con algún representante del grupo para poder llegar a un acuerdo sobre la pista.	Falta de tiempo en alguna actividad importante.
Envío de comandos en vacío: El envío de instrucciones de movimiento sin verificar el estado actual del robot provocaba comportamientos erráticos o bloqueos en el Hub.	Implementación de Bloqueo de Estado (Safety Interlock).	Falla del robot al momento de ejecutar un movimiento.
Conexión Automática "Ciega": El sistema intentaba conectar automáticamente al primer dispositivo encontrado, generando errores si había otros Hubs o señales Bluetooth cerca.	Selección manual por el usuario: Se implementa un método que fuerza el despliegue de una ventana con la lista de dispositivos disponibles.	Falla del robot al momento de ejecutar un movimiento.

8. Prueba de funcionamiento del sistema

8.1. Descripción de la prueba de funcionamiento

La prueba de funcionamiento tiene como objetivo validar la funcionalidad mínima del sistema.

Para definir dicha funcionalidad, es necesario considerar la etapa del proceso de extracción subterránea que el robot simula dentro del proyecto. En este caso, el sistema corresponde a un vehículo minero asociado a la etapa de transporte.

La prueba de funcionamiento consiste en verificar que el robot responda correctamente a los comandos enviados de forma inalámbrica desde la interfaz gráfica de usuario, estableciendo previamente la conexión entre el cliente y el servidor. Una vez establecida la comunicación, el robot debe ejecutar las acciones de movimiento definidas en el requerimiento funcional RF1.

La prueba se considera exitosa si el robot ejecuta cada uno de los movimientos: avanzar, retroceder, girar a la izquierda y girar a la derecha, lo cual evidencia su capacidad de movilidad y a su vez el correcto funcionamiento de la comunicación inalámbrica.

8.2. Resultados observados para la prueba de funcionamiento

Durante la ejecución de la prueba se logró una comunicación estable entre la interfaz y el robot. El sistema respondió correctamente a los comandos enviados desde la GUI.

Detalles de la ejecución y desempeño:

Conectividad: El módulo de escaneo (DeviceSelectWindow) identificó correctamente el HUB LEGO. La conexión se establece en menos de 3 segundos, habilitando los controles de la interfaz inmediatamente tras el handshake.

Sincronización Interfaz-Teclado: Se observó una latencia imperceptible entre la acción del usuario y la respuesta del sistema. Un resultado destacado fue la retroalimentación visual en tiempo real: al presionar las

teclas físicas (ej: Flecha Arriba), el botón virtual "AVANZAR" en la pantalla cambió a su estado activo (color azul oscuro) simultáneamente al movimiento del robot. Esto confirmó el correcto funcionamiento de los listeners de eventos configurados en el código.

Desempeño de Carga y Seguridad: El vehículo transportó la carga simulada manteniendo la estabilidad estructural en todo momento, incluso durante los virajes y cambios de velocidad. Se cumplió con el requerimiento de seguridad al completar el circuito exclusivamente mediante teleoperación, sin necesidad de reposicionamiento manual ni contacto físico por parte del operador durante la prueba.

Navegación y Maniobrabilidad:

- El robot navegó exitosamente por la pista, evadiendo los obstáculos cilíndricos y adaptándose a la superficie del terreno.

- La función de Dirección Asíncrona permite al robot girar las ruedas mientras avanzaba, logrando curvas fluidas en lugar de movimientos segmentarios.

- El comando de "Restablecer Dirección" (Tecla Espacio / Botón Centrar) corrigió eficazmente la orientación de las ruedas a 0 grados exactos gracias al uso de los encoders absolutos de los motores.

Cierre de Sesión: Al finalizar la prueba, se validó la funcionalidad del botón "Desconectar". Al accionarlo, el sistema ejecutó exitosamente el protocolo de cierre seguro: envió el comando de parada de emergencia a los motores, liberó el canal de comunicación Bluetooth y restauró la interfaz al estado inicial de búsqueda, confirmando que el ciclo de vida del software se gestiona correctamente de principio a fin.

9. Conclusión

El proyecto consistió en el diseño de un robot, utilizando Lego Spike Prime, asociado a la etapa de transporte del proceso de extracción subterránea en la minería. El robot debe simular el transporte de carga y ser controlado inalámbricamente mediante una interfaz gráfica de usuario, para esto se utilizó Pybricks, encargado de gestionar la comunicación con el robot a través del protocolo BLE (Bluetooth Low Energy) y CustomTkinter para un diseño de alto contraste, mejorando la legibilidad del usuario durante el uso del programa.

Asimismo, el proyecto permitió reforzar la importancia del trabajo colaborativo, la planificación mediante la carta Gantt que facilitó la organización de las actividades definidas en el proyecto y el uso de Github para el control de versiones. Se evidenció que la gestión de riesgos es un recurso útil para el desarrollo del proyecto, permitiendo identificar problemas asociados a un riesgo y en consecuencia facilitar la toma de decisiones en conjunto para obtener una solución, evitando retrasos mayores y asegurando la continuidad del trabajo.

A partir de las actividades definidas, se presentaron problemas que afectaron en el tiempo determinado para cada una de estas, por lo que se idearon soluciones apoyadas de la gestión de riesgos para afrontar de manera realista el evento.

Los problemas que se encontraron son, en mayoría, relacionados con el control del robot. En primera instancia se contempló el uso de una manilla de PS4 para el control del robot, la implementación de esta actividad se extendió más de lo determinado, pero finalmente se completó permitiendo observar si la lógica del código era adecuada; Otros de los problemas encontrados fueron el envío de comandos vacíos y la conexión automática “ciega”, por lo que se implementó el Bloqueo de Estado (Safety Interlock) y un método que despliega una ventana con una lista de los dispositivos disponibles como soluciones a cada problema, respectivamente.

Como mejora futura, se propone extender el manejo de errores del sistema, incorporando mecanismos de detección de pérdida de comunicación y recuperación automática de la conexión Bluetooth.

En conclusión, el proyecto desarrolló habilidades técnicas y organizacionales, fortaleciendo el trabajo en equipo, toma de decisiones y el uso de herramientas tecnológicas para implementar soluciones.

10. Referencias

1. Lego (1 de Octubre del 2025) *Set SPIKE™ Prime de LEGO® Education*

   https://www.lego.com/es-us/product/lego-education-spike-prime-set-45678

2. *Notebook Lenovo V14 G4 IRU i7-13620H 16GB SSD 512Gb W11P Lenovo | Paris.cl*. (s.f.). https://www.paris.cl/notebook-lenovo-v14-g4-iru-i7-13620h-16gb-ssd-512gb-w11p-MKM7LHYL5X.html?utm_source=google&utm_medium=organic&utm_campaign=organicshopping

3. Ripley (s.f.). *NOTEBOOK GAMER LENOVO IDEAPAD GAMING 3 15IMH05 INTEL CORE I5 8 GB RAM 1 TB HDD 256 GB SSD NVIDIA GTX 1650-TI 15.63 15IMH05*

   https://simple.ripley.cl/notebook-gamer-lenovo-ideapad-gaming-3-15imh05-intel-core-i5-8-gb-ram-1-tb-hdd-256-gb-ssd-nvidia-gtx-1650-ti-156-2000392412564p?s=mdco

4. *ASUS TUF Gaming F16 (2024)*. (2024). ASUS Chile. https://www.asus.com/cl/laptops/for-gaming/tuf-gaming/asus-tuf-gaming-f16-2024/

5. *Galaxy Tab A9 64GB*. (14 de octubre de 2023). Samsung Cl. https://www.samsung.com/cl/tablets/galaxy-tab-a/galaxy-tab-a9-wifi-graphite-64gb-sm-x110nzaal07/

6. *Control Joystick Inalámbrico doble Shock P4 Negro Levo* (s.f.). Me lo llevo. Recuperado el 24 de Octubre del 2025 de https://melollevo.cl/products/control-joystick-inalambrico-dualshock-para-ps4-levo?srsltid=AfmBOoqiEx0UjcUh6pMlOoxi1ENU1do5VYgNFyF06ZAJGb2fG88CPQpk

7. *Descarga de la app SPIKE™ | LEGO® Education*. (s.f.). LEGO® Education.

   https://education.lego.com/es-es/downloads/spike-app/software/

8. *Download Visual Studio Code - Mac, Linux, Windows.* (2021, 3 noviembre). https://code.visualstudio.com/download

9. Google Workspace. (2025). *Documentos de Google: Editor de documentos y archivos PDF en línea | Google Workspace.* Google Workspace. https://workspace.google.com/intl/es-419/products/docs/

10. Valk, L. (2025). *Installing Pybricks*. Pybricks. https://pybricks.com/learn/getting-started/install-pybricks/

11. *Overview - Redmine.* (s.f.). https://www.redmine.org/

12. *Salarios de Project manager en 2025*. (s.f.). Computrabajo.com. Recuperado el 27 de noviembre de 2025, de https://cl.computrabajo.com/salarios/project-manager

13. *Sueldo de Analista de proyecto en Chile.* (2025). Indeed.com. https://cl.indeed.com/career/analista-de-proyecto/salaries

14. *Sueldo Desarrollador Full Stack en Chile 2025: en pesos chilenos y dólares*. (2025). Coderhouse.com. https://www.coderhouse.com/pe/sueldos/sueldo-desarrollador-full-stack-chile-2025

15. Kurth, M. (27 agosto de 2025). Programador en Chile: mirá cuánto podés ganar hoy. *Instituto Profesional Providencia*. https://ipp.cl/general/cuanto-gana-un-programador-en-chile/#:~:text=%C2%BFCu%C3%A1nto%20gana%20un%20programador%20en%202025?.en%20la%20compensaci%C3%B3n%20que%20reciben

16. Wikipedia contributors. (2025, octubre 25). *List of system quality attributes*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=List_of_system_quality_attributes&oldid=1318760832

17. AbrahamCode24. (s.f.). *Lego_controller_pybricks: Interfaz gráfica (GUI) en Python para controlar motores LEGO mediante Pybricks y comunicación asíncrona*. GitHub. https://github.com/AbrahamCode24/Lego_controller_pybricks