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

15/12/2025

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

Índice de Contenidos

Índice de Tablas

Índice de Figuras

1. Planteamientos del problema y objetivos

En la minería existen múltiples procesos en la extracción subterránea de minerales, el proyecto se basará en desarrollar un modelo 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.

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

- Experimentar con el set Lego Spike Prime.
- Explorar las librerías de Python en base a Lego Spike Prime.
- Armar y ensamblar un modelo eficiente al momento de moverse y que pueda mantener una carga estable durante el trayecto.
- Hacer una interfaz gráfica con Tkinter apta para que el usuario pueda usarla.

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 modelo.
- El robot debe ser capaz de moverse y llevar rápida y eficientemente la carga.
- Debe controlarse a distancia, es decir 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 necesitar 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 modelo 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 donde se detallarán las instrucciones para el uso del robot a través de su interfaz gráfica y cómo usarlo de manera eficiente.
- Presentación: se detallan las distribuciones del equipo y se ofrece una vista general del robot al igual se detallarán los objetivos, retos superados y sus soluciones.

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 objetivo de optimizar el rendimiento colectivo y cumplir con los objetivos establecidos como grupo.

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 para el modelo LEGO 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
Investigar cómo funciona el programa LEGO spike.	Todo el grupo.	Mejor claridad del programa con el que se va a trabajar.
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 modelo LEGO 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.

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.

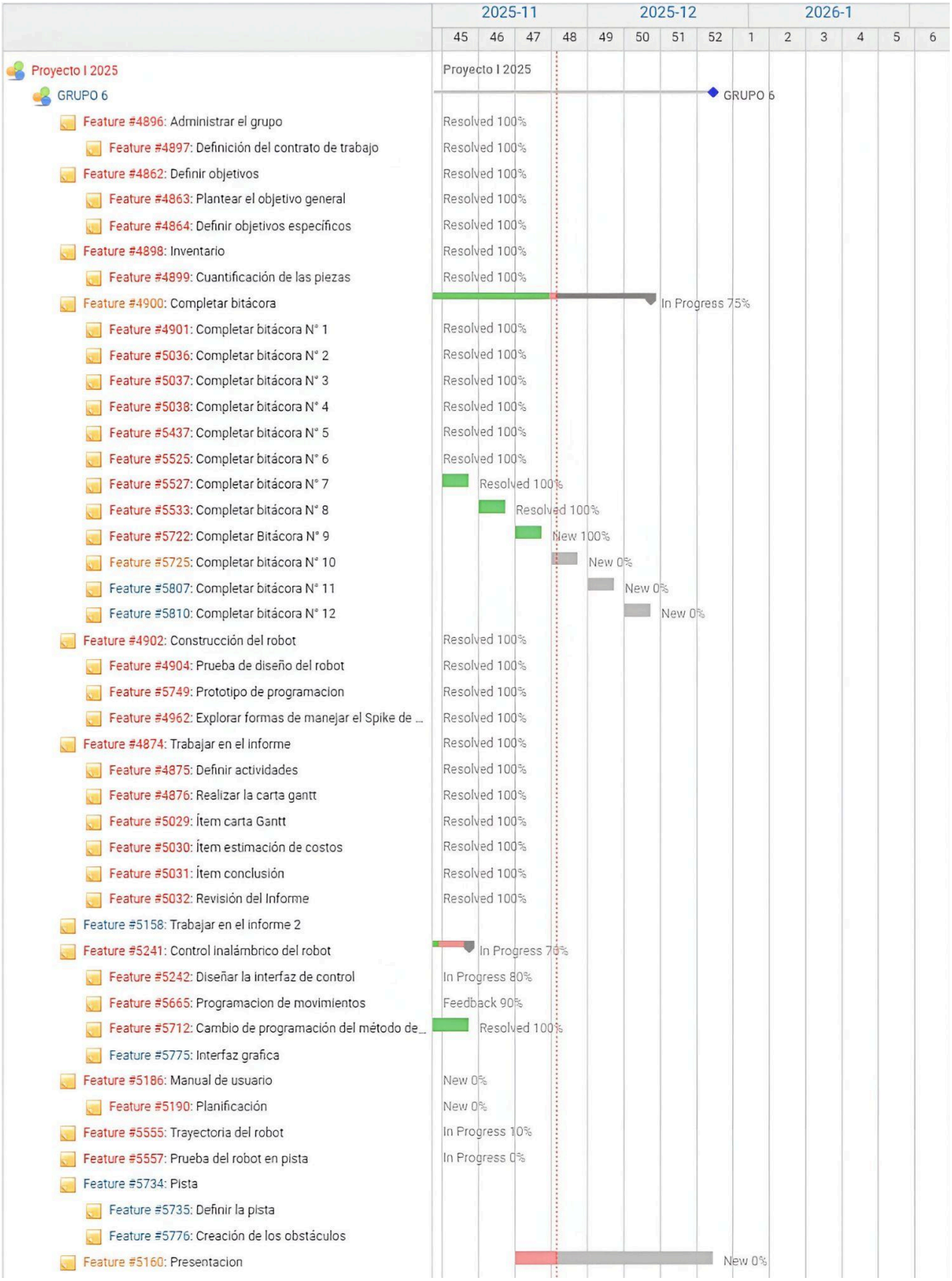


Figura 1: *Carta Gantt del proyecto*

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 modelo 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.

4.3. Recursos humanos

Se presentan una serie de tablas 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 (5)
Visual Studio Code	$0 (6)
Documentos de Google	$0 (7)
Pybricks	$0 (8)
Redmine	$0 (9)
Total:	$0

Costo de Trabajador:

Tabla 6: *Costo de trabajador*

Rol	Horas	Horas extras	Precio/Hora (CLP)
Jefe del Grupo	68 horas	—	$8.250
Documentador	68 horas	—	$4.382
Ensamblador	68 horas	—	$11.935
Programador	68 horas	3 horas	$5.282
Total:	272	3	$2.045.578

Destacado:

- *Cálculo de Precio/Hora: (horas *4)/referencia de precio.*
- *Las horas son contabilizadas desde la primera clase con el robot (semana del 29 de septiembre).*
- *En las referencias se encontrarán referenciados los sitios de donde se obtuvieron los valores referencia de precio.*

Total de costos:

Tabla 7: *Total costos*

Costo hardware	Costo software	Costo empleados	Total (CLP)
$2.498.450	$0	$2.045.578	$4.544.028

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 modelo a escala de un vehículo 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 desplazarse en 4 direcciones básicas: 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 proporcionar un botón que permite 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

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 y la facilidad para realizar mejoras futuras.

Se eligió el modelo Cliente-Servidor. La razón principal de esta elección es la separación de responsabilidades: queremos que la interfaz gráfica (en el PC) se encargue solo de interactuar con el usuario, mientras que el robot se dedique exclusivamente a mover los motores.

Si necesitamos mejorar el diseño de la interfaz o agregar botones nuevos, podemos hacerlo sin riesgo de dañar el código que controla el equilibrio del robot. Esto garantiza un desarrollo más seguro y ordenado.

- Cliente (Control): Es el componente de alto nivel ejecutado en el computador. Desarrollado en Python con la librería Tkinter, actúa como interfaz entre el usuario y el sistema. Su función es capturar los eventos generados por el operador (inputs) y transformarlos en solicitudes, además de recibir y visualizar los datos de respuesta enviados por el robot (logs y estado de conexión).
- Comunicación (Bluetooth): Es el medio de comunicación entre el cliente y el servidor. Se utiliza el protocolo BLE (Bluetooth Low Energy) gestionado por las librerías de Pybricks. Este canal es bidireccional: permite el envío de instrucciones de control hacia el

robot y, simultáneamente, el retorno de información de estado hacia el computador.

- Servidor (hub): El Hub del robot opera como el servidor de ejecución. Utilizando el firmware de Pybricks, se mantiene a la espera de recibir los comandos enviados por el cliente. Al recibir una solicitud, la interpreta y ejecuta las instrucciones sobre los motores, enviando retroalimentación al cliente sobre la ejecución de las acciones.

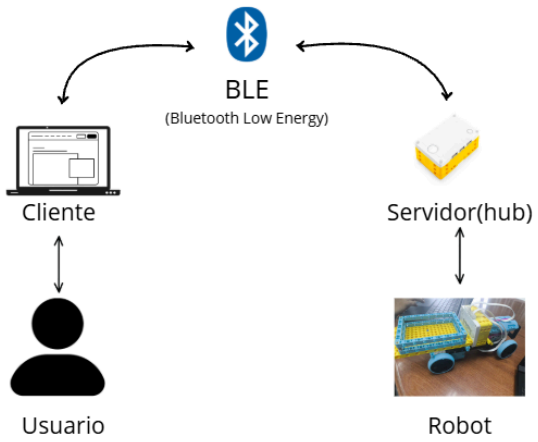


Figura 2: *Diagrama de la arquitectura Cliente-Servidor*

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.

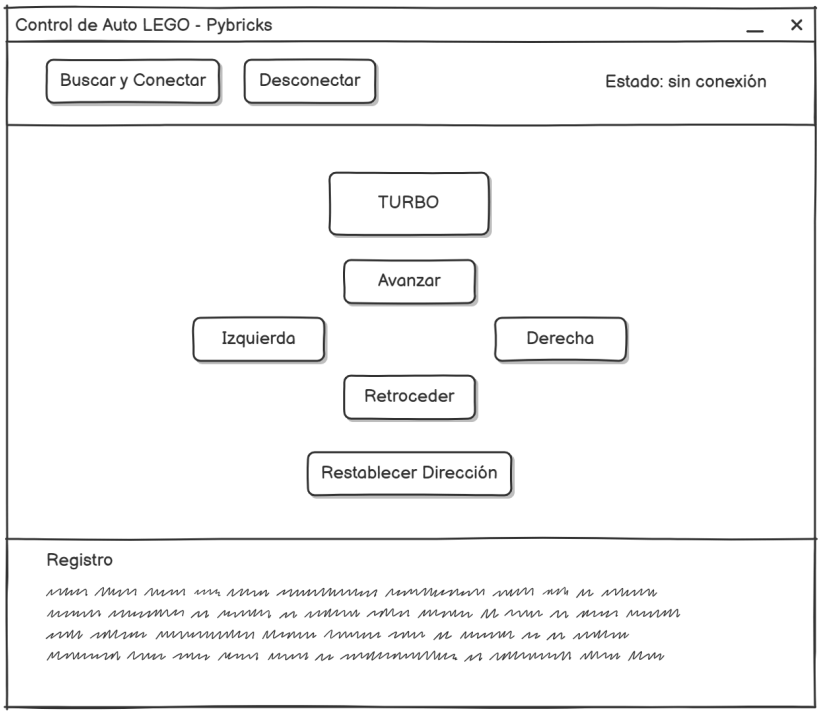


Figura 3: *Wireframe de baja fidelidad del GUI*

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. Gestión de Concurrencia (Clase BLEWorker): Dado que las operaciones de red (conectar, enviar datos) son bloqueantes por naturaleza, ejecutarlas en el hilo principal congelaría la interfaz gráfica. Para solucionar esto, se implementó un patrón de concurrencia utilizando Hilos (threading) y programación asíncrona (asyncio).

- Funcionamiento Detallado: La clase BLEWorker inicializa un bucle de eventos (event loop) en un hilo separado y utiliza una estructura de datos tipo Cola FIFO (Queue). Cuando el usuario presiona un botón en la clase LegoGUI, el sistema ejecuta la validación del evento y encola el comando correspondiente. El hilo secundario procesa secuencialmente esta cola, gestionando la transmisión segura al robot sin afectar la fluidez de la interfaz.

```python
async def _runner(self):
    try:
        while True:
            self.log("Esperando selección de dispositivo...")
            await self._connect_request.wait()

            device = self._target_device
            if not device:
                self._connect_request.clear()
                continue

            try:
                self.log(f"Conectando a {device.name}...")
                self.hub = PybricksHubBLE(device)
                await self.hub.connect()
                self.log("Conectado. Listo para conducir.")
                self.running.set()

                while self.running.is_set():
                    drive_cmd = await self.queue.get()
                    await execute_command(self.hub, drive_cmd, self.log)

            except Exception as e:
                self.log(f"Error de conexión/ejecución: {e}")
            finally:
                if self.hub:
                    try:
                        await self.hub.disconnect()
                        self.log("Hub desconectado.")
                    except:
                        pass
                self.hub = None
                self.running.clear()
                self._connect_request.clear()
                self._target_device = None

    except asyncio.CancelledError:
        pass
```

Figura 4: *Parte de la Clase BLEWorker*

2. Gestión de Estados: El método “set_controls_enabled” deshabilita físicamente los botones de tracción si no existe una conexión confirmada ("handshake") con el Hub.

```python
def set_controls_enabled(self, enabled: bool):
    state = "normal" if enabled else "disabled"
    buttons = [
        self.btn_turbo,
        self.btn_avanzar,
        self.btn_retro,
        self.btn_izquierda,
        self.btn_derecha,
        self.btn_restablecer
    ]
    for btn in buttons:
        btn.configure(state=state)
```

Figura 5: *Método set controls enable*

3. Modo Turbo: Se ha integrado un control especial de alta prioridad que permite al usuario solicitar una sobremarcha temporal. Este botón dispara un evento específico (cmd_turbo_click) que se procesa con prioridad en la lógica de generación de comandos.

```python
def cmd_turbo_click(self):
    if self.worker.running.is_set():
        self.worker.send_command("run_turbo")
```

Figura 6: *Botón turbo*

4. Escaneo y Filtrado BLE (Clase DeviceSelectWindow): Para garantizar la conexión en entornos ruidosos (con múltiples dispositivos Bluetooth), se implementó un escáner selectivo. Este módulo filtra los dispositivos que no transmiten un nombre identificable, presentando al usuario una lista limpia de Hubs disponibles para la vinculación.

```python
def _update_list(self, devices):
    self.label.configure(text="Selecciona tu Hub:")
    self.btn_refresh.configure(state="normal")

    found_any = False
    for dev in devices:
        name = dev.name if dev.name else "Desconocido"
        address = dev.address

        if name != "Desconocido":
            found_any = True
            btn = ctk.CTkButton(
                self.scroll_frame,
                text=f"{name}\n({address})",
                command=lambda d=dev: self._on_device_click(d),
                height=50,
                anchor="w"
            )
            btn.pack(pady=5, padx=5, fill="x")

    if not found_any:
        lbl = ctk.CTkLabel(self.scroll_frame, text="No se encontraron dispositivos con nombre.")
        lbl.pack(pady=20)

def _on_device_click(self, device):
    self.on_select_callback(device)
    self.destroy()
```

Figura 7: *Parte de la clase DeviceSelectWindow*

6.2.2. Servidor

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

1. Generador de Perfiles de Movimiento (create_program): Esta función es el núcleo lógico del sistema. Transforma la intención del usuario (ej: "Avanzar", "Turbo") en código ejecutable MicroPython.

- Perfiles de Velocidad: El sistema distingue entre navegación de precisión y navegación rápida.
- Modo Normal: Configura los motores a 800 RPM con una duración de 2000 ms.
- Modo Turbo: Fuerza los motores a su límite físico (1100) con una duración reducida de 500 ms para evitar colisiones por inercia excesiva.
- Inversión de Polaridad: El código generado invierte automáticamente la polaridad de uno de los motores (motorB.run(-800) vs motorF.run(800)) para compensar que los motores están montados en espejo en el chasis del robot.

```
drive_commands = {
    'run_forward': "motorB.run(-800)\nmotorF.run(800)",
    'run_turbo':   "motorB.run(-1100)\nmotorF.run(1100)", # <--- COMANDO TURBO
    'run_backward': "motorB.run(400)\nmotorF.run(-400)",
    'stop': "motorB.stop()\nmotorF.stop()\nmotorD.stop()",
    'izquierda': "motorD.run_target(300, -35)\nmotorD.stop()",
    'derecha': "motorD.run_target(300, 35)\nmotorD.stop()",
    'centro': "motorD.run_target(300, 0)\nmotorD.stop()",
}
```

Figura 8: *Parte del método create_program*

2. Control Diferencial vs. Posicional: El script generado aplica diferentes teorías de control según el actuador:

- Control de Velocidad: Para los motores de tracción, se envían comandos de potencia constante por tiempo (wait), asumiendo que el terreno es plano.
- Control de Posición : Para el motor de dirección (Motor D), se utiliza el método run_target(velocidad, ángulo). Esto utiliza los encoders internos del motor para girar exactamente 35 grados y frenar activamente (hold) en esa posición, simulando una dirección mecánica real y garantizando que el robot no pierda la trayectoria curva.

```
    program = f"""
from pybricks.hubs import PrimeHub
from pybricks.pupdevices import Motor
from pybricks.parameters import Port
from pybricks.tools import wait

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

{drive_code}
{wait_code}
motorB.stop()
motorF.stop()
"""
    return program
```

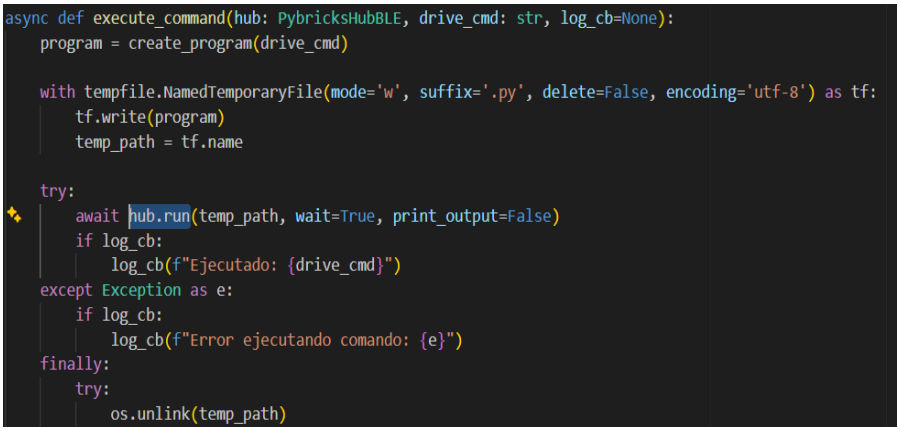
Figura 9: *Fragmento de código del Servidor - Instanciación de hardware y ensamblaje de la lógica de control.*

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.

- Creación de Archivo Temporal: El método execute_command toma el código generado (string) y lo escribe en un archivo temporal (tempfile) en el sistema de archivos del computador.

- Interacción con Librería pybricksdev: Se invoca el método hub.run() de la librería pybricksdev. Esta librería actúa como el controlador del protocolo de comunicación.

- Envío vía Bluetooth: pybricksdev lee el archivo temporal, compila las instrucciones y las transmite vía Bluetooth Low Energy (BLE) hacia el servicio GATT del Hub. No se envía el archivo completo como documento, sino el flujo de comandos ejecutables que el firmware interpreta en tiempo real.



```python
async def execute_command(hub: PybricksHubBLE, drive_cmd: str, log_cb=None):
    program = create_program(drive_cmd)

    with tempfile.NamedTemporaryFile(mode='w', suffix='.py', delete=False, encoding='utf-8') as tf:
        tf.write(program)
        temp_path = tf.name

    try:
        await hub.run(temp_path, wait=True, print_output=False)
        if log_cb:
            log_cb(f"Ejecutado: {drive_cmd}")
    except Exception as e:
        if log_cb:
            log_cb(f"Error ejecutando comando: {e}")
    finally:
        try:
            os.unlink(temp_path)
```

Figura 10: *Fragmento de execute_command mostrando la creación del archivo temporal y la llamada a hub.run.*

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 11: *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 35 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 lal 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 no han sido probados de manera individual. En la fase actual se ha priorizado el cumplimiento de los requerimientos funcionales del proyecto; posteriormente, se analizarán los requerimientos no funcionales.

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	Implementación de Bloqueo de Estado (Safety Interlock).	Falla del modelo al momento de ejecutar un movimiento.
Conexión Automática "Ciega"	Selección manual por el usuario: Se implementa un método que fuerza el despliegue de una ventana con la lista de dispositivos disponibles.	Falla del modelo al momento de ejecutar un movimiento.

8. Conclusión

Como segunda fase del proyecto, se han abordado diferentes puntos del proyecto que han permitido avanzar en el desarrollo del control del robot que se encuentra realizando pruebas de respuesta de las instrucciones enviadas desde la interfaz, la conexión inalámbrica realizada a través del protocolo BLE (Bluetooth Low Energy) gestionado por librerías de Pybricks, la interfaz gráfica de usuario diseñada con una librería basada en Tkinter (CustomTkinter) que es funcional y el restablecimiento de dirección para la corrección de desviaciones en la trayectoria.

El periodo de entrega próximo ha influido en las decisiones tomadas por el equipo para una entrega dentro del plazo establecido.

Los problemas encontrados actualmente se relacionan con el control del robot, por lo cual se realizan pruebas para observar y analizar su comportamiento, además la comunicación para el diseño de la pista se ha dificultado por una deficiente relación, por lo que se decidió realizar una reunión con el encargado de la pista del otro grupo de la asignatura, para llegar a un acuerdo con respecto a este.

Actualmente quedan actividades pendientes, es decir, que aún no han iniciado. Ante la eventualidad de nuevas actividades, se planificará con el equipo para evitar la prolongación fuera de plazo del proyecto. Estas actividades se mencionan a continuación:

- Presentación 2.
- Primera práctica en la pista.
- Manual de usuario.
- Trabajar en el informe 3.
- Presentación 3.

En conclusión, el proyecto avanza conforme las funcionalidades están siendo cumplidas. La cercanía de la finalización del proyecto afecta las decisiones del equipo y en la posibilidad de explorar soluciones que sean extensas en implementación y que causen un desfase en la entrega, comprometiendo el cumplimiento de los objetivos definidos en la primera fase del proyecto.

9. 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. *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

13. *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

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

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