UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@ Computación e Informática

Informe Inicial
“Robot Separador de Bloques”

Alumnos: Nicolas Olivares
Victor Breems
Sebastian Cahuachia
Gabriel Delgado
Willy Cruz

Asignatura: Proyecto I

Profesor: Baris Klobertanz

Arica, 30 de octubre 2025

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
26/09/2025	1.0	Formulación del Proyecto	**Nicolas Olivares, Sebastian Cahuachia**
02/10/2025	1.1	Recopilar datos	**Nicolas Olivares, Sebastian Cahuachia**
03/10/2025	1.2	Recopilar datos	**Nicolas Olivares, Victor Breems, Sebastian Cahuachia**
09/10/2025	1.3	Modificación del ítem Organización del Personal y Planificación del Proyecto	**Nicolas, Olivares, Victor Breems**
10/10/2025	1.4	Avance del ítem Planificación del Proyecto y Planificación de los Recursos	**Sebastian Cahuachia, Nicolas Olivares, Victor Breems**
16/10/2025	1.5	Avance del ítem Conclusión y Referencias	**Nicolas Olivares, Sebastian Cahuachia, Victor Breems**
17/10/2025	1.6	Revisión del Informe	**Nicolas Olivares, Sebastian Cahuachia, Victor Breems, Willy Cruz, Gabriel Delgado**

Tabla de Contenidos

Índice de tablas

Índice de figuras

1.Panorama General

1.1 Introducción

Durante el presente semestre se desarrolló un proyecto grupal cuyo objetivo fue diseñar y construir un robot utilizando el kit educativo LEGO SPIKE PRIME. El robot es capaz de identificar y separar bloques LEGO según su color, clasificándolos en compartimentos específicos (amarillo, azul, verde, rojo y morado).

La programación fue desarrollada en Python, buscando un funcionamiento autónomo, eficiente y de fácil uso. A lo largo del semestre se realizaron mejoras progresivas tanto en la estructura física como en el código, optimizando el desempeño general del sistema.

La construcción del robot se realizó de manera eficiente gracias a los recursos educativos disponibles de LEGO SPIKE PRIME, los cuales proporcionaron guías paso a paso para el armado del modelo. Paralelamente, el desarrollo del código en Python fue optimizado clase a clase, mejorando la velocidad y precisión del robot en sus tareas.

1.2 Objetivo

1.2.1 Objetivo General

Desarrollar un robot utilizando la plataforma LEGO SPIKE PRIME capaz de identificar y separar bloques según su color, mediante el uso de sensores y motores, implementando programación en Python para lograr un funcionamiento autónomo, preciso y eficiente.

1.2.2 Objetivo específico

- Ensamblar el robot utilizando el set LEGO SPIKE PRIME.
- Investigar la librería Python de LEGO SPIKE PRIME.
- Realizar pruebas de funcionamiento del robot.
- Diseñar una interfaz gráfica para el control del robot.
- Optimizar la velocidad y precisión en la detección de colores.
- Desarrollar el código en el lenguaje de programación Python.

1.3 Restricciones

- El sistema debe programarse en Python.
- Se debe utilizar exclusivamente el set LEGO SPIKE PRIME.
- El proyecto debe gestionarse mediante la plataforma Redmine.
- El robot debe ser capaz de moverse y separar bloques por color.
- El grupo debe contar con un mínimo de cinco integrantes.
- El proyecto cuenta con un tiempo determinado de desarrollo.
- El robot debe permitir control remoto para pausar y reanudar su funcionamiento.

1.4 Entregables

Bitácoras: Informes de carácter semanal que registran el progreso del equipo a lo largo del desarrollo del proyecto, incluyendo las actividades realizadas, las dificultades enfrentadas, las recomendaciones de mejora y las acciones ejecutadas. Elaboradas por un integrante designado, permiten disponer de una visión detallada que apoya la toma de decisiones, la asignación de responsabilidades y la identificación de temas a tratar en conjunto.

Carta Gantt: Representación gráfica de la planificación del proyecto, en la cual se visualizan las tareas, su duración y secuencia dentro de una línea de tiempo, facilitando la gestión y control de los plazos y recursos mediante el seguimiento del avance de las actividades.

Informe de Formulación: Documento que expone la organización y estrategia del equipo para el cumplimiento de los objetivos de la asignatura. En él se describen la asignación de roles, las metas planteadas y las medidas implementadas para alcanzar el propósito académico. Asimismo, se incluyen las primeras observaciones surgidas durante el proceso de desarrollo y la documentación relevante recopilada a lo largo del semestre.

Presentaciones: Instancias en las que se exponen los objetivos del proyecto, los desafíos enfrentados y las soluciones aplicadas. Además, se destacan los logros obtenidos, la distribución del equipo de trabajo y se entrega una visión general del funcionamiento del robot.

2. Organización del personal

La organización del trabajo en equipo resulta fundamental para el desarrollo eficiente del proyecto, ya que permite una distribución adecuada de las tareas y responsabilidades requeridas para alcanzar los objetivos establecidos.

2.1 Descripción de los roles y responsabilidades

Jefe de proyecto: Representante del equipo, responsable de supervisar y coordinar el avance del proyecto, además de velar por el cumplimiento eficiente de los objetivos establecidos.

Ensamblador: Encargado del montaje y armado de las piezas del robot, supervisa el correcto funcionamiento de sus componentes y colabora con el programador para asegurar que el diseño físico sea compatible con la lógica del software.

Programador: Responsable del desarrollo de la codificación y del correcto funcionamiento del robot, trabajando en conjunto con el ensamblador para garantizar la coherencia entre el software y el hardware.

Documentador: Responsable de registrar el progreso del proyecto y de la elaboración de los informes correspondientes.

Diseñador: Encargado de la creación del logotipo y del diseño estético general del proyecto.

2.2 Personal que cumplirá los Roles

En la **Tabla 1** muestra los roles de cada integrante del Grupo:

Tabla 1: *Roles asignados al personal*

Rol	Responsable
Jefe de Proyecto	Victor Breems
Ensamblador	Sebastian Cahuachia
Programador	Gabriel Delgado
Diseñador	Willy Cruz
Documentador	Nicolas Olivares

2.3 Métodos de Comunicación

Los métodos de comunicación utilizados por el equipo son WhatsApp, empleado para el envío de mensajes ante situaciones emergentes, y Discord, plataforma utilizada para la realización de reuniones remotas y coordinación de actividades tanto dentro como fuera de la universidad.

3. Planificación de Proyecto

3.1 Actividades

Tabla 2: *Actividades*

Nombre	Responsable
Preparar y programar hub del robot	Todos los integrantes del grupo.
Desarrollar Prototipo del robot en lego con python	Gabriel Delgado Willy Cruz
Organizar roles y planificar proyecto del robot	Todos los integrantes del grupo.
Probar movimientos y funcionamiento del robot(separador de color)	Gabriel Delgado Willy Cruz
Armar primer modelo base del robot	Nicolas Olivares Victor Breems Sebastian Cahuachia
Armar separador de bloques por color con sensor	Nicolas Olivares Victor Breems

	Sebastian Cahuachia
Terminar primer modelo de separador de bloques con sensor	Nicolas Olivares Victor Breems Sebastian Cahuachia
Programar interfaz inalámbrica para control manual del robot	Nicolas Olivares Victor Breems Sebastian Cahuachia

3.2 Carta Gantt

Se presentan las tareas del proyecto junto con su estado de avance y las fechas correspondientes en las que se están desarrollando.

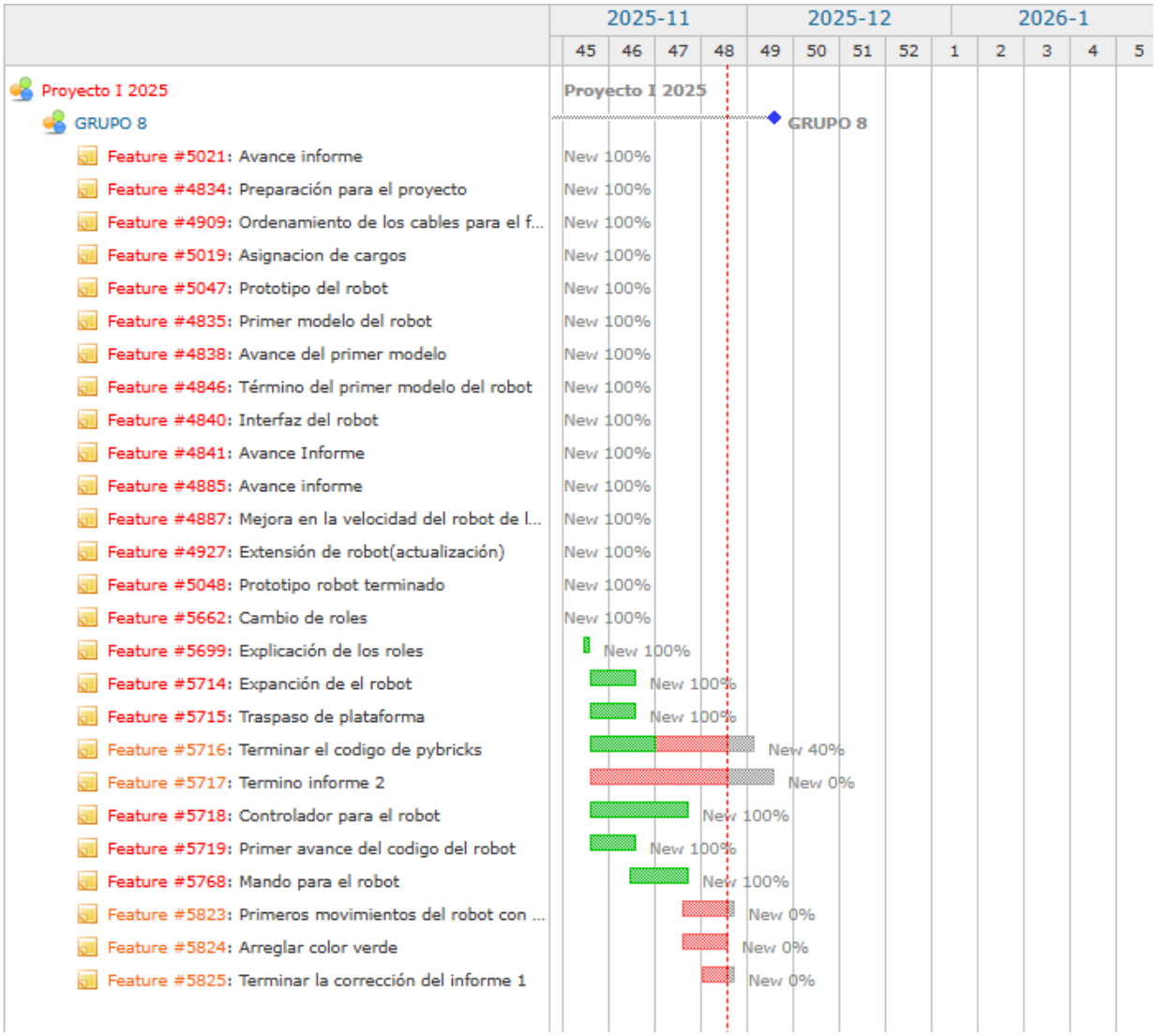

Figura 1: *Carta Gantt*

3.3. Gestión de Riesgos

A continuación, se presenta una tabla que muestra un desglose de los problemas identificados durante la primera fase del proyecto. Dicha tabla resume el impacto de cada desafío, clasificando los riesgos en cinco niveles distintos, los cuales se asocian a diferentes tipos de daño

1. Daño catastrófico: Requiere la aplicación inmediata de medidas correctivas, ya que puede provocar la detención del proyecto o un retraso significativo, incluso implicando la necesidad de reiniciar su desarrollo desde el inicio.

2. Daño crítico: Exige la implementación de acciones correctivas oportunas, debido a que puede generar retrasos en varias etapas del proyecto.

3. Daño circunstancial: Debe resolverse de manera inmediata, ya que puede afectar el desarrollo de una etapa fundamental del proyecto.

4. Daño irrelevante: Corresponde a un imprevisto de baja importancia, que no requiere atención prioritaria y puede ser solucionado en cualquier momento.

5. Daño recurrente: No representa un impacto significativo, sin embargo, se presenta de forma reiterada y puede generar retrasos en las sesiones de trabajo, aunque no en las etapas del proyecto.

Tabla 3: *Gestión de riesgos*

Riesgo	Nivel de impacto	Acción Remedial
Descarga de batería del robot.	5	Utilizar su cargador y volver a utilizarlo cuando tenga una carga considerable para su uso.
Error en la codificación	5	Corregir los principales errores lógicos y sintácticos identificados e implementar nuevas soluciones
Inasistencias en horas de trabajo del proyecto	5	Organizar y coordinar los horarios disponibles entre todos.
Pérdida de legos	3	Solicitar esas piezas al encargado de los legos en

		caso de que no encontremos la pieza le avisame al encargado.
Problemas con la conexión a internet.	3	Buscar una red estable o compartir internet desde un teléfono celular.
Desempeño del robot no es eficiente a causa de fallas no previstas	2	Buscar hacer un robot más adecuado a lo que estamos buscando y sea eficiente.
Caída accidental del robot	2	Recuperar las piezas y rearmar el robot en caso que se rompa una pieza le debemos avisar al encargado para que pedir un repuesto.
Pérdida de información necesaria para presentar el proyecto	2	Hacer un respaldo de la información importante o necesaria en un usb o disco duro.

4. Planificación de los Recursos

La planificación del proyecto requiere la organización de los recursos que serán utilizados, con el fin de realizar un análisis detallado de los gastos asociados, los cuales se dividen en tres categorías: hardware, software y costos de personal.

4.1 Hardware

- Set LEGO SPIKE PRIME.
- Computador con sistema operativo compatible para la programación y control del robot.

4.2 Software

- Redmine, plataforma de gestión y organización del proyecto.
- Sistema Operativo Windows, utilizado para la programación del robot.
- Aplicación LEGO SPIKE, empleada para el desarrollo y carga del código en el robot.
- Pybricks.
- Visual Studio Code.
- Python.

4.3 Estimación de Costos

La **Tabla 4** muestra el costo de hardware:

Tabla 4: *Costos de Hardware*

Hardware	Cantidad	Precio a pesos Chilenos
Set Lego SPIKE PRIME	1	$ 764.925 clp
Kit de extensión de Lego SPIKE PRIME	1	$ 157.990 clp
Lenovo Thinkpad x390 yoga	5	$ 599.990 clp
Total:	–	$ 3.922.865 clp

La **Tabla 5** muestra Costo de Software:

Tabla 5: *Costo de Software*

Producto	Precio
Licencia Microsoft Office	$ 14.990 clp
Pybricks	$ 0 clp
Visual Studio Code	$ 0 clp
Python	$ 0 clp
Total:	$ 14.990

La **Tabla 6** muestra Costo de Trabajador:

Tabla 6: *Costo de Trabajador*

Rol	Horas	Horas Extras	Precio/Hora
Jefe del Proyecto	56 horas	15 horas	$ 30.000 clp
Programador	48 horas	12 horas	$ 28.000 clp
Ensamblador	48 horas	4 horas	$ 23.000 clp
Documentador	54 horas	14 horas	$ 25.000 clp
Total:	206 horas	45 horas	$ 106.000 clp

Destacado:

- La contabilización de las horas trabajadas se considera a partir de la conformación del grupo de trabajo.
- Para la categorización de las horas de trabajo, se tomó en cuenta el tiempo correspondiente a las actividades realizadas durante las clases.
- En el caso de las horas extras, se consideró el tiempo trabajado fuera del horario de clases, pero dentro del mismo departamento.
- Para la estimación del valor por hora del costo de personal, se tomó como referencia el promedio de remuneración de un ingeniero civil informático en Chile, ajustado según los roles desempeñados por cada integrante, información obtenida a partir de consultas en foros y fuentes disponibles en internet.

La **Tabla 7** muestra Total Costo:

Tabla 7: *Total Costo*

Costo Hardware	$ 3.997.855 clp
Costo Software	$ 14.990 clp
Costo Empleados	$ 6.706.000 clp
Total:	$ 10.643.855 clp

5. Análisis y diseño

5.1 Especificación de requerimientos

Antes de definir los requerimientos del sistema, es necesario considerar los siguientes aspectos fundamentales.

- En primer lugar, se debe comprender claramente el problema a resolver, así como los objetivos del proyecto y la solución propuesta. En este caso, el objetivo es desarrollar un robot capaz de identificar y clasificar bloques LEGO según su color, utilizando el set LEGO SPIKE PRIME y programación en Python.
- Asimismo, es importante identificar al usuario y al cliente del sistema, quienes forman parte de los stakeholders del proyecto.
    - El usuario: corresponde a la persona que utilizará con mayor frecuencia el sistema, interactuando directamente con el robot y su interfaz gráfica.
    - El cliente: corresponde a la entidad académica que solicita y evalúa el desarrollo del proyecto en el contexto de la asignatura Proyecto I.
- La correcta identificación de estos actores permite definir requerimientos coherentes con las necesidades del proyecto y con los objetivos académicos planteados.

5.1.1 Requerimientos funcionales

Los requerimientos funcionales describen las acciones y comportamientos que el sistema será capaz de realizar, considerando tanto la solución propuesta como las expectativas de los stakeholders identificados previamente.

Los requerimientos funcionales del sistema se enumeran a continuación, permitiendo su verificación durante las pruebas del proyecto:

RF-01: El sistema debe detectar automáticamente el color de una pieza LEGO utilizando el sensor de color del kit LEGO SPIKE PRIME.

RF-02: El sistema debe clasificar las piezas detectadas en compartimentos físicos separados según su color (amarillo, azul, verde, rojo y morado).

RF-03: El robot debe desplazarse de forma controlada para posicionarse correctamente frente al compartimento correspondiente.

RF-04: El sistema debe permitir iniciar, mover, pausar y reanudar el proceso de clasificación mediante la interfaz gráfica de usuario.

RF-05: Una vez iniciado el proceso, el robot debe operar de forma autónoma sin intervención del usuario.

Quedan fuera del alcance de esta fase funcionalidades más avanzadas, tales como la incorporación de señales sonoras asociadas al reconocimiento de colores por parte del sensor, las cuales permitirían realizar la clasificación de manera no visual, incluso de forma remota y sin apoyo de una interfaz gráfica. Estas funcionalidades podrán ser abordadas en etapas posteriores del proyecto.

5.1.2 Requerimientos no funcionales

Los requerimientos no funcionales corresponden a atributos de calidad que permiten evaluar cómo debe funcionar el sistema, considerando su desempeño, estabilidad y facilidad de uso. Estos atributos describen propiedades medibles y verificables del sistema.

RNF-01 (Disponibilidad): El robot debe operar de forma continua durante al menos 30 minutos sin fallas durante la demostración del proyecto.

RNF-02 (Robustez): El sistema debe continuar operando ante errores menores, como lecturas erróneas ocasionales del sensor de color, sin detener completamente su ejecución.

RNF-03 (Rendimiento): El robot debe clasificar una pieza LEGO en un tiempo máximo de 5 segundos desde su detección.

RNF-04 (Usabilidad): La interfaz gráfica debe permitir al usuario comprender y controlar el sistema sin necesidad de capacitación previa, mediante botones claramente identificados.

5.2 Arquitectura de software

En el presente proyecto se utiliza una arquitectura de tipo cliente-servidor.

- El cliente corresponde a la aplicación desarrollada en Python que incluye la interfaz gráfica de usuario, desde la cual el usuario puede enviar instrucciones al robot.
- El servidor corresponde al robot implementado con LEGO SPIKE PRIME, el cual recibe las instrucciones del cliente y ejecuta las acciones correspondientes mediante sensores y motores.

Esta arquitectura permite una separación clara de responsabilidades y facilita la comunicación entre el usuario y el sistema físico.

La **Figura 2** se presenta un diagrama que ilustra la arquitectura del sistema y la interacción entre sus componentes.

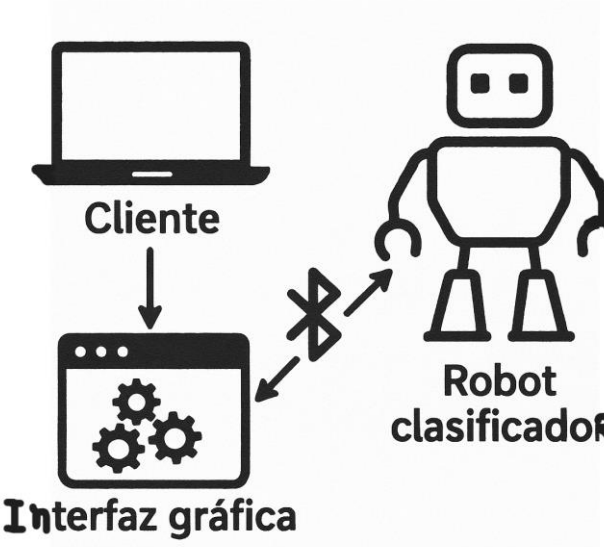

Figura 2: *Diagrama*

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

En esta sección se presenta el diseño inicial de la interfaz gráfica de usuario mediante un wireframe de baja fidelidad. Este diseño corresponde a un bosquejo que permite planificar la estructura básica de la interfaz y la disposición de sus elementos principales.

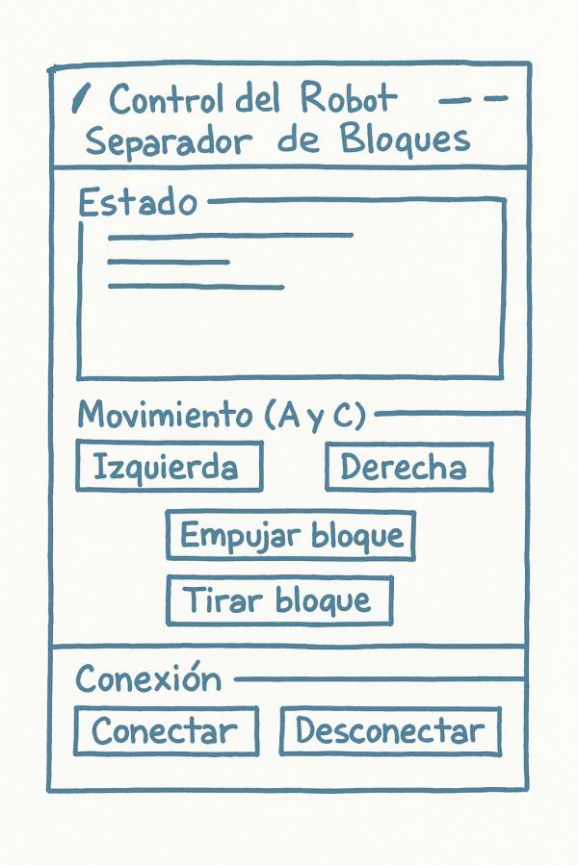

Figura 3: *Wireframe de baja fidelidad*

6. Implementación

En esta sección se presentan los avances obtenidos hasta el momento en el desarrollo del proyecto. Se incluye la justificación de la configuración del robot a partir de principios físicos básicos, así como una descripción de los componentes más relevantes del sistema implementado.

6.1 Fundamentos de los movimientos

Para justificar el movimiento de giro del robot, se considera un giro de 90° realizado en un tiempo aproximado de 1,5 segundos.

La velocidad angular se calcula mediante la fórmula:

$\omega = \theta / t$

Donde:

$\theta = 90° = \pi/2$ rad

$t = 1{,}5$ s

$\omega = (\pi/2) / 1{,}5 \approx 1{,}05$ rad/s

Este valor permite un giro controlado y preciso, evitando errores en la alineación del robot frente a los compartimentos.

6.2 Descripción del sistema

6.2.1 Cliente

El cliente del sistema corresponde a la aplicación desarrollada en Python que permite la interacción entre el usuario y el robot.

El código del cliente se encuentra disponible en el siguiente repositorio de GitHub:

Repositorio GitHub:
[https://github.com/pescolo/Robot_Separador_de_Bloques/blob/main/robot_clasificador.py]

- **Generación dinámica del programa que ejecuta el robot (create_program)**

El cliente genera automáticamente un programa en Pybricks según el comando que el usuario seleccione en la interfaz (por ejemplo: izquierda, derecha, empujar o tirar). Para ello, la función create_program(drive_cmd, claw_cmd) convierte cada comando en una instrucción de movimiento para los motores mediante run(...). Luego, el programa creado configura el hub (PrimeHub), define los motores según los puertos utilizados y ejecuta la acción solicitada. Finalmente, se incorpora un tiempo de espera breve y se detienen los motores, con el fin de mantener un movimiento controlado y evitar que el robot continúe funcionando sin supervisión.

```
def create_program(drive_cmd: str, claw_cmd: str) -> str:
    drive_commands = {
        'izquierda': "motorC.run(-300)",
        'derecha': "motorC.run(300)",
        'empujar': "motorF.run(500)",
        'tirar': "motorF.run(-500)"
    }

    drive_code = drive_commands.get(drive_cmd, "motorC.stop(); motorF.stop()")

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

hub = PrimeHub()

motorC = Motor(Port.C)
motorF = Motor(Port.F)

{drive_code}

wait(300)
motorC.stop()
motorF.stop()
"""
    return program
```

Figura 4: *Generación dinámica del programa.*

- **Envío y ejecución del programa en el hub por Bluetooth (execute_command)**

La ejecución del movimiento en el robot se realiza mediante la función asíncrona execute_command(hub, drive_cmd, claw_cmd, logger). Esta función recibe el programa generado previamente, lo guarda temporalmente como un archivo con extensión .py y lo envía al hub para su ejecución utilizando hub.run(...). Durante este proceso, el cliente registra mensajes en el panel de estado de la interfaz, informando si el comando fue enviado correctamente o si ocurrió algún error. Al finalizar, el archivo temporal se elimina para mantener el sistema ordenado y evitar la acumulación de archivos innecesarios.

```
async def execute_command(hub, drive_cmd, claw_cmd, logger):
    program = create_program(drive_cmd, claw_cmd)

    try:
        with tempfile.NamedTemporaryFile(delete=False, suffix=".py") as temp_file:
            temp_file.write(program.encode("utf-8"))
            temp_file_path = temp_file.name

        logger(f'Enviando programa al hub desde archivo temporal: {temp_file_path}')

        await hub.run(temp_file_path)
        logger(f'Comando ejecutado: drive={drive_cmd}, claw={claw_cmd}')

    except Exception as e:
        logger(f'ERROR ejecutando el comando: {e}')
    finally:
        if 'temp_file_path' in locals() and os.path.exists(temp_file_path):
            os.remove(temp_file_path)
```

Figura 5: *Envío y ejecución del programa en el hub por Bluetooth.*

- **Gestión de conexión y envío de comandos por Bluetooth (BLEWorker._runner)**

La conexión Bluetooth y el envío de comandos al robot se gestionan mediante un componente asíncrono que funciona como “worker”. En el método _runner() se busca el hub disponible utilizando find_device(), luego se crea la conexión con PybricksHubBLE y se establece mediante connect(). Una vez conectado, el sistema se mantiene en

ejecución esperando nuevas instrucciones a través de una cola (asyncio.Queue). Cada comando recibido se procesa enviándolo al robot mediante la función execute_command(...), y el estado del proceso (conexión, ejecución o errores) se informa mediante mensajes registrados en el panel de estado.

```
async def _runner(self):
    try:
        self.log("Buscando hub Bluetooth...")
        device = await find_device()
        if not device:
            self.log("No se encontró hub.")
            return

        self.hub = PybricksHubBLE(device)
        await self.hub.connect()
        self.log("Conectado al hub. Listo para mover.")
        self.running.set()

        while True:
            drive_cmd = await self.queue.get()
            await execute_command(self.hub, drive_cmd, self.avanzar_activo, self.log)

    except asyncio.CancelledError:
        pass
    except Exception as e:
        self.log(f"Error en worker: {e}")
    finally:
        if self.hub:
            try:
                await self.hub.disconnect()
                self.log("Hub desconectado.")
            except Exception as e:
                self.log(f"Error al desconectar: {e}")
        self.running.clear()
```

Figura 6: *Gestión de conexión y envío de comandos.*

- **Interfaz gráfica: panel de estado y actualización de logs (update_logs)**

La interfaz gráfica incorpora un panel de estado en el cual se muestran mensajes relevantes del sistema, tales como la búsqueda del hub, el estado de la conexión Bluetooth, la ejecución de comandos y posibles errores. La función update_logs() obtiene estos mensajes desde una cola (Queue) y los despliega de manera periódica en el cuadro de texto mediante el uso de root.after(...). Este mecanismo permite mantener la interfaz gráfica responsiva, aun cuando se estén ejecutando tareas de comunicación Bluetooth en segundo plano.

```
def main_gui():
    root = tk.Tk()
    root.title("Control del Robot Separador de Bloques")
    root.geometry("420x520")

    log_queue = Queue()
    worker = BLEWorker(log_queue)
    worker.start()

    frame_log = ttk.LabelFrame(root, text="Estado")
    frame_log.pack(fill="both", padx=10, pady=10)

    txt = tk.Text(frame_log, height=10, state="disabled")
    txt.pack(fill="both", padx=5, pady=5)

    def update_logs():
        while not log_queue.empty():
            msg = log_queue.get()
            txt.config(state="normal")
            txt.insert("end", msg + "\n")
            txt.see("end")
            txt.config(state="disabled")
        root.after(100, update_logs)
```

Figura 7: *Panel de estado y actualización de logs.*

- **Interfaz gráfica: controles de movimiento (botones)**

El control manual del robot se implementa mediante botones en la interfaz gráfica, los cuales envían comandos al worker a través de worker.send_command(...). De esta

forma, el usuario puede ejecutar acciones de movimiento, como desplazarse hacia la izquierda o derecha, y también activar funciones mecánicas específicas, como empujar o tirar bloques. Este diseño permite un control directo y sencillo del robot desde la GUI, manteniendo una interacción clara entre el usuario y el sistema.

```
frame_move = ttk.LabelFrame(root, text="Movimiento (A y C)")
frame_move.pack(fill="both", padx=10, pady=10)

ttk.Button(frame_move, text="⬅ Izquierda", command=lambda: worker.send_command("izquierda")).grid(row=1, column=0, pady=5)
ttk.Button(frame_move, text="➡ Derecha", command=lambda: worker.send_command("derecha")).grid(row=1, column=2, pady=5)
ttk.Button(frame_move, text="Empujar bloque", command=lambda: worker.send_command("empujar")).grid(row=2, column=1, pady=5)
ttk.Button(frame_move, text="Tirar bloque", command=lambda: worker.send_command("tirar")).grid(row=3, column=1, pady=5)
```

Figura 7: *Controles de movimiento (botones).*

6.2.2 Servidor

El servidor corresponde al programa ejecutado directamente en el robot LEGO SPIKE PRIME. Este componente recibe las instrucciones enviadas desde el cliente y controla los motores y sensores necesarios para la clasificación de los bloques.

El código del servidor implementa la lógica principal del sistema, incluyendo la lectura del sensor de color y la activación de los mecanismos de separación.

En la **Imagen 8** se muestra el servidor que se encontraría dentro del hub, ya que es allí donde se recibe las instrucciones enviadas desde el cliente.

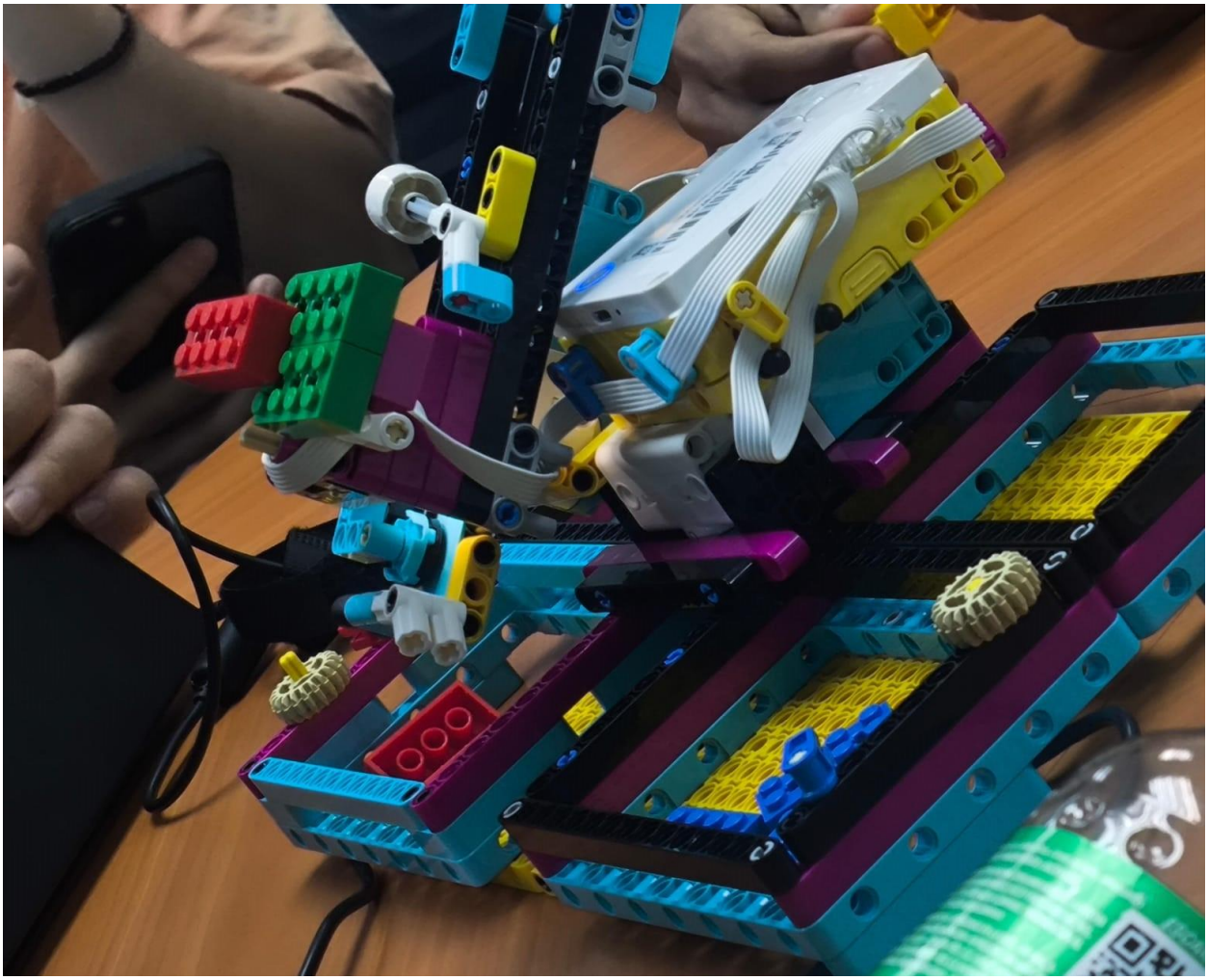

Figura 8: Servidor

6.2.3 Interfaz gráfica de usuario (GUI)

La interfaz gráfica de usuario permite controlar el funcionamiento del robot separador de bloques de manera manual y supervisar su estado de conexión. En la sección superior

se muestra el estado del sistema, donde se visualizan mensajes relacionados con la búsqueda y conexión del robot mediante Bluetooth.

La interfaz incluye controles de movimiento que permiten desplazar el robot hacia la izquierda o derecha, facilitando su posicionamiento durante el proceso de clasificación. Asimismo, se incorporan botones específicos para ejecutar acciones mecánicas, tales como empujar y retirar bloques, permitiendo un control directo de estas funciones.

Finalmente, la sección de conexión dispone de botones para establecer o finalizar la comunicación entre el computador y el robot, otorgando al usuario un control claro y directo sobre la interacción con el sistema.

Figura 9: *Interfaz gráfica de control del robot separador de bloques*

7. Resultados

7.1 Estado actual del proyecto

Actualmente, el robot es capaz de identificar y clasificar bloques LEGO por color de manera funcional.

Además se implementó de manera correcta la interfaz gráfica en la cual se puede controlar el robot LEGO SPIKE de manera manual por el usuario.

7.2 Problemas encontrados y solucionados

Durante el desarrollo del proyecto se presentaron diversos problemas, tales como errores iniciales en la detección de colores, problemas conexión bluetooth. Estas dificultades fueron abordadas mediante la búsqueda de información, para poder solucionarlas. Y esto permitió mejorar la estabilidad y precisión del sistema.

8. Conclusión

En conclusión, el desarrollo del robot clasificador de bloques utilizando la plataforma LEGO SPIKE PRIME representó una experiencia de aprendizaje integral y colaborativa para el equipo de trabajo. A lo largo del semestre, cada integrante aportó sus

conocimientos desde su rol correspondiente, permitiendo alcanzar los objetivos planteados al inicio del proyecto de manera progresiva y organizada.

Durante las primeras etapas, el trabajo se centró en el ensamblaje del robot y la comprensión de sus componentes físicos, proceso que presentó diversas dificultades iniciales relacionadas con la estructura y el posicionamiento de los mecanismos. No obstante, estas fueron superadas mediante el trabajo en equipo y la iteración constante sobre el diseño, logrando una base física estable y funcional para el sistema.

Posteriormente, el enfoque se trasladó al desarrollo del software, donde se implementó una solución basada en programación en Python, integrando una arquitectura cliente-servidor mediante comunicación Bluetooth. En este contexto, el computador actúa como cliente a través de una interfaz gráfica, mientras que el hub LEGO SPIKE PRIME cumple el rol de servidor, ejecutando las instrucciones recibidas y controlando los motores y sensores del robot. Esta separación de responsabilidades permitió un control más claro del sistema y facilitó la interacción entre el usuario y el robot.

Uno de los principales aprendizajes obtenidos fue el desarrollo y depuración del código, especialmente durante la implementación de la interfaz gráfica y la comunicación Bluetooth. Las instancias en las que el sistema no respondía según lo esperado obligaron al equipo a analizar el funcionamiento interno del programa, revisar el flujo de ejecución y corregir errores lógicos, fortaleciendo así las habilidades de análisis y resolución de problemas.

Finalmente, los resultados obtenidos demuestran que el robot es capaz de identificar y clasificar bloques LEGO por color de manera funcional, además de permitir su control manual mediante una interfaz gráfica, cumpliendo con los requerimientos definidos para esta fase del proyecto. Si bien existen funcionalidades avanzadas que quedan fuera del alcance actual, el sistema desarrollado constituye una base sólida para futuras mejoras y ampliaciones, consolidando los conocimientos adquiridos en robótica, programación y trabajo colaborativo.

Actividades futuras:

- Terminar el código en visual studio.
- Terminar actividades de la carta gantt.

9. Referencia

LEGO Education. (s.f.). *LEGO® Education SPIKE™ Prime (45678)*. https://www.lego.com

LEGO Education. (s.f.). *LEGO® Education SPIKE™ Prime expansion set (45681)*. https://www.lego.com

Full Compras. (s. f.). *LEGO Education SPIKE Prime (45678)*. https://www.fullcompras.cl

Mercado Libre. (s.f.). *Notebook Lenovo Ideapad Gaming 3, Intel Core i7*. https://www.mercadolibre.cl

Licencias Originales. (s.f.). *Microsoft Office 2024 Professional Plus*. https://www.licenciasoriginales.cl

Paris. (s. f.). *Control DualShock verde camuflado*. https://www.paris.cl

Python Software Foundation. (s.f.). *Python programming language*. https://www.python.org

Microsoft. (s.f.). *Visual Studio Code*. https://code.visualstudio.com

Pybricks. (s.f.). *Pybricks documentation*. https://pybricks.com