UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería
Computación e Informática

PRESENTACIÓN INFORME

“ROBOT SEPARADOR DE BLOQUES”

Integrantes:

Nicolas Olivares
Sebastián Cahuachia
Víctor Breems
Gabriel Delgado
Willy Cruz

Asignatura: Proyecto I
Profesor: Baris Nikolai Klobertanz Quiroz
Plan de Proyecto – Fase 3

DICIEMBRE 2025

Índice de CONTENIDOS

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 cual optimiza el tiempo y reduce personal para una empresa de paqueterías para separar de forma óptima los paquetes según su tipo.

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.

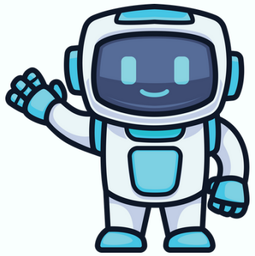
La construcción del robot se realizó de manera eficiente gracias a los recursos educativos disponibles de LEGO SPIKE PRIME, los cuales proporcionaron guías paso a paso para el armado del modelo.

Paralelamente, el desarrollo del código en Python fue optimizado clase a clase, mejorando la velocidad y precisión del robot en sus tareas.

PROBLEMAS ENCONTRADOS Y SOLUCIONADOS

1. Problema: Dificultades para conectar la PC y el Hub de LEGO SPIKE PRIME.

Solución: Se implementa la librería pybricksdev en Python. Esta librería optimiza la comunicación vía BLE (Bluetooth Low Energy), permitiendo la ejecución de scripts directamente en el Hub de LEGO SPIKE PRIME, eliminando el retraso que generaba el firmware original.



2. Problema: El sensor arrojaba lecturas erróneas debido a las condiciones de luz ambiental, confundiendo colores similares.
Solución: Se utilizó la clase ColorSensor de la librería Pybricks, la cual permite acceder a los valores crudos de reflexión y HSV (Matiz, Saturación). Esto permitió calibrar el sensor por software para filtrar el ruido lumínico, en lugar de usar la detección de color básica predeterminada.

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.

Objetivo Especifico

- 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.

DESCRIPCIÓN DE LOS ROLES Y RESPONSABILIDADES DE LOS MIEMBROS DEL EQUIPO DE TRABAJO

Jefe de Proyecto: Nicolas Olivares

Representante del equipo, supervisa y organiza el progreso del proyecto además asegurar que los objetivos se cumplan de manera eficiente.

Ensamblador: Willy Cruz

Encargado del montaje y el armado de las piezas, monitorea el cumplimiento de las funcionalidades del robot, en conjunto con el programador para asegurar que el diseño físico sea compatible con la lógica del código.

Programador: Sebastián Cahuachia
Encargado del área de la codificación y funcionamiento del robot, en colaboración del ensamblador para asegurar coherencia entre el software y el hardware.

Documentador: Gabriel Delgado
Encargado de registrar el avance del proyecto, junto con la redacción de los informes.

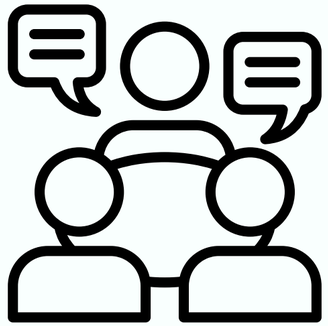
Diseñador: Victor Breems

Encargado de la creación del logotipo y la estética del proyecto además de estar encargado el diseño del robot.

MÉTODOS DE COMUNICACIÓN

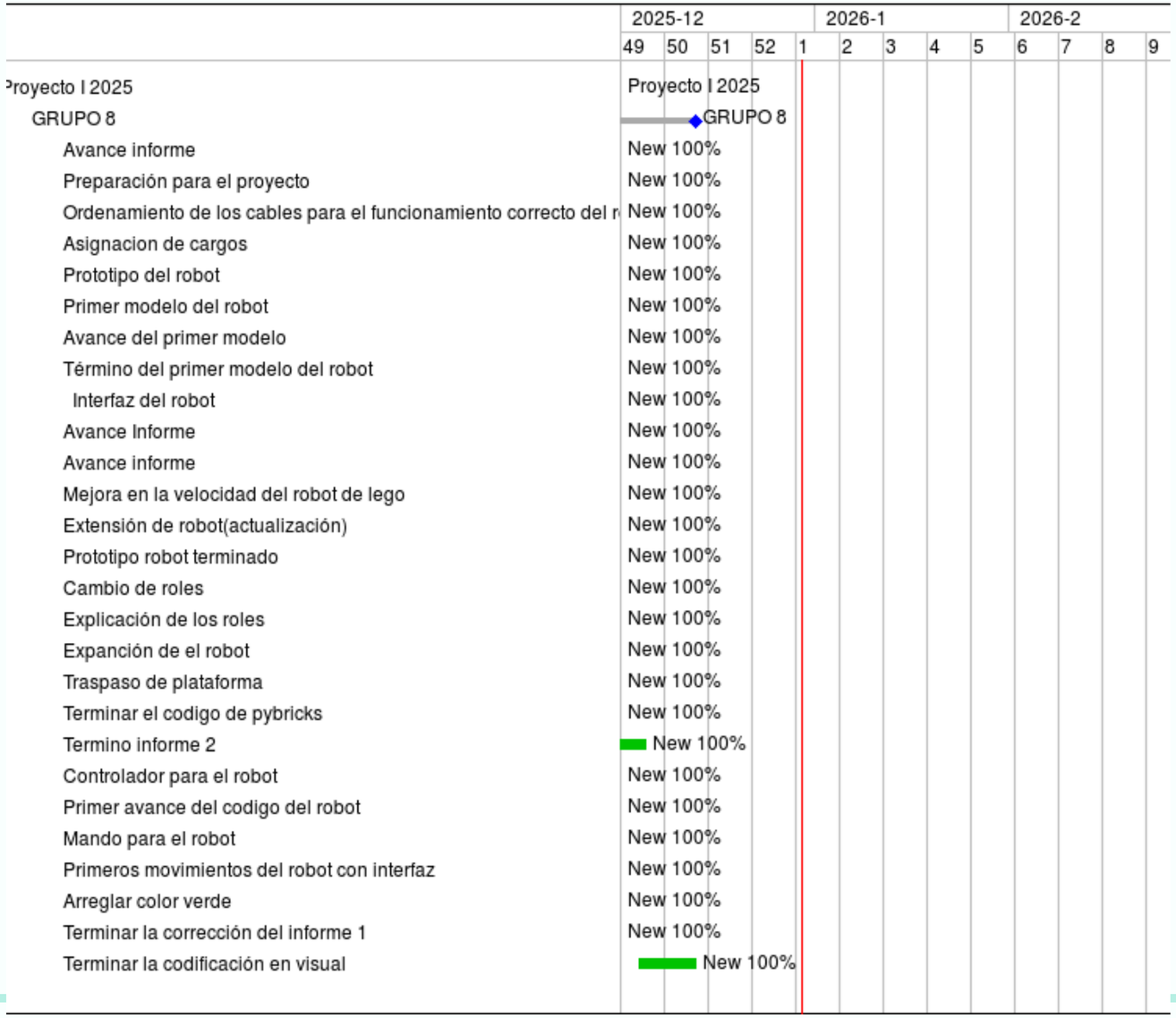
01 **WhatsApp**

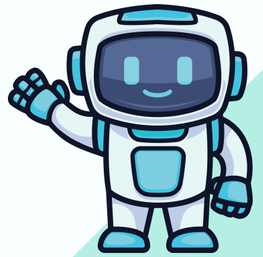
02 **REUNIONES (PRESENCIALES)**



03 **Discord**

CARTA GANTT


2025-12
2026-1
2026-2
49 50 51 52 1 2 3 4 5 6 7 8 9
Proyecto I 2025
GRUPO 8
Avance informe
Preparación para el proyecto
Ordenamiento de los cables para el funcionamiento correcto del r
Asignacion de cargos
Prototipo del robot
Primer modelo del robot
Avance del primer modelo
Término del primer modelo del robot
Interfaz del robot
Avance Informe
Avance informe
Mejora en la velocidad del robot de lego
Extensión de robot(actualización)
Prototipo robot terminado
Cambio de roles
Explicación de los roles
Expanción de el robot
Traspaso de plataforma
Terminar el codigo de pybricks
Termino informe 2
Controlador para el robot
Primer avance del codigo del robot
Mando para el robot
Primeros movimientos del robot con interfaz
Arreglar color verde
Terminar la corrección del informe 1
Terminar la codificación en visual
New 100%



Gestión de Riesgos

Para la gestión de riesgos, utilizamos una escala de 5 niveles, donde el nivel 1 representa un daño catastrófico para el proyecto y el nivel 5 un daño recurrente o menor.

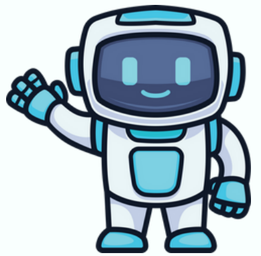
1. Daño catastrófico: Requiere la aplicación inmediata de medidas correctivas, ya que puede provocar la detención del proyecto o un retraso significativo, incluso implicando la necesidad de reiniciar su desarrollo desde el inicio.

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 en poco tiempo, ya que puede afectar el desarrollo de una etapa fundamental del proyecto.

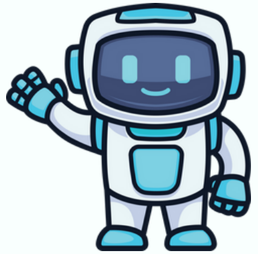
4. Daño irrelevante: Corresponde a un imprevisto de baja importancia, que no requiere atención prioritaria y puede ser solucionado en cualquier momento.

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.



Dentro de nuestro análisis, los riesgo con mayor impacto son estos tres:

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.
Pérdida de legos	3	Solicitar esas piezas al encargado de los legos en caso de que no encontremos la pieza.
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.



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:

$\omega = \theta / t$

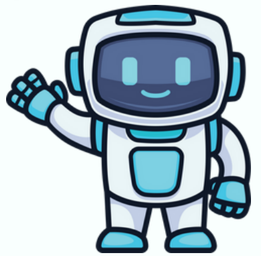
Donde:

$\theta = 90° = \pi/2$ 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.

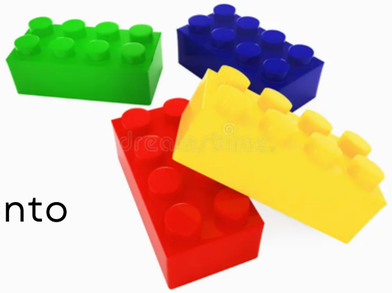


REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales describen las acciones y comportamientos que el sistema es capaz de realizar:

RF-01 – Detección de color
El sistema detecta automáticamente el color de una pieza LEGO utilizando el sensor de color del kit LEGO SPIKE PRIME.

RF-02 – Clasificación por color
El sistema clasifica las piezas detectadas en compartimentos físicos separados según su color: amarillo, azul, verde, rojo y morado.



RF-03 – Desplazamiento controlado
El robot se desplaza de forma controlada para posicionarse correctamente frente al compartimento correspondiente al color detectado.

RF-04 – Control mediante interfaz gráfica
El sistema permite iniciar, mover, pausar y reanudar el proceso de clasificación mediante una interfaz gráfica de usuario (GUI).

RF-05 – Operación autónoma
Una vez iniciado el proceso, el robot opera de forma autónoma, sin intervención del usuario durante el ciclo de clasificación.

REQUERIMIENTOS NO FUNCIONALES

Se redefinieron los RNF como atributos medibles y verificables:

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

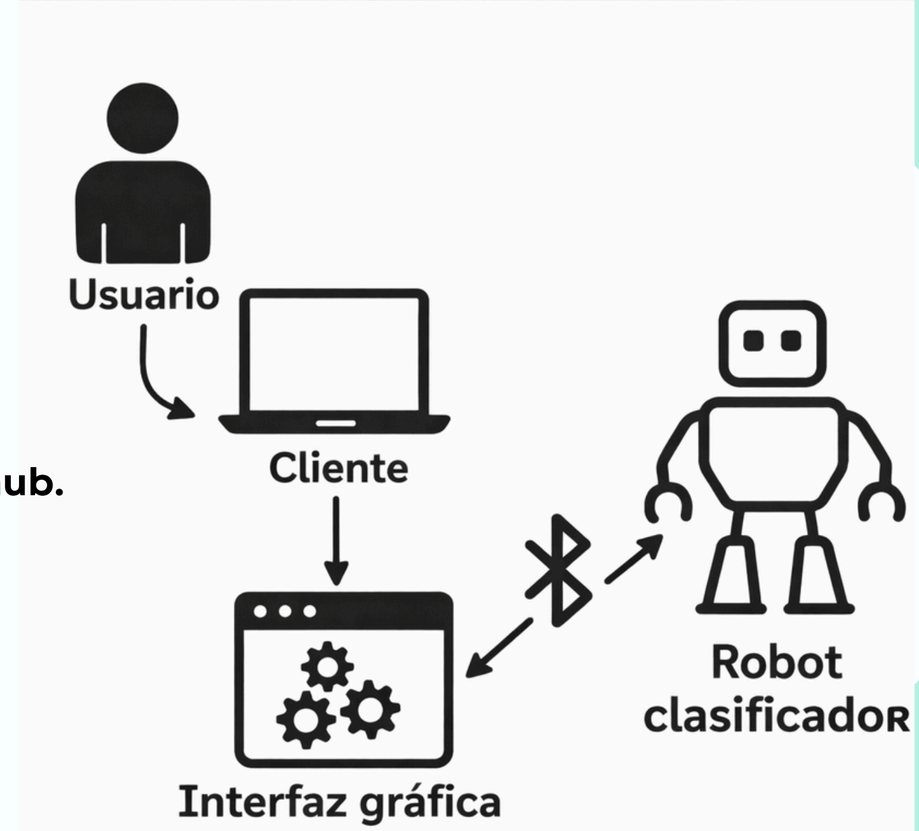
RNF-02 (Robustez):
El sistema continúa operando ante lecturas erróneas ocasionales del sensor.

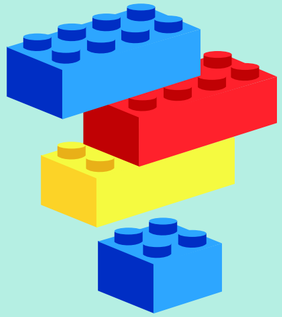
RNF-03 (Rendimiento):
Clasificación de una pieza en un tiempo máximo de 5 segundos.

RNF-04 (Usabilidad):
La interfaz permite el uso sin capacitación previa.

ARQUITECTURA DE SOFTWARE

- Usuario
- Cliente
- Aplicación Python con interfaz gráfica (GUI).
- Servidor
  - Hub LEGO SPIKE PRIME que ejecuta instrucciones.
  - Pybricksdev generar los archivos temporales enviados al hub.
- Comunicación
  - Bluetooth Low Energy (BLE) mediante librería Pybricksdev.





HARDWARE

- **Set Lego SPIKE PRIME.**
- **Computador con el sistema operativo para poder programar con el robot.**

SOFTWARE

- **Redmine, página de organización del Proyecto.**
- **Sistema Operativo de Windows, para programar el robot.**
- **Aplicación de lego SPIKE para programar el robot.**
- **Pybricks.**
- **Visual Studio Code.**
- **Python.**

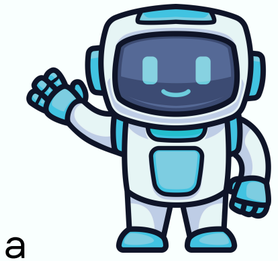
ARQUITECTURA DE SOFTWARE

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

- **Cliente**: Corresponde al hardware del robot. Su responsabilidad es ejecutar las instrucciones a bajo nivel a través de su firmware para activar el motor y sensor.

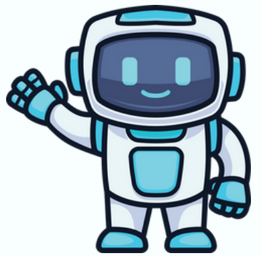
- **Servidor**: Está constituida con la librería de Pybricksdev que genera el archivos temporales. Su función es mandar esos archivos temporales a la Hub de LEGO SPIKE PRIME.

- **Comunicación**: La comunicación se realiza mediante una conexión con bluetooth con la librería de Pybricksdev que está encargado de enviar comando a la Hub de LEGO SPIKE PRIME

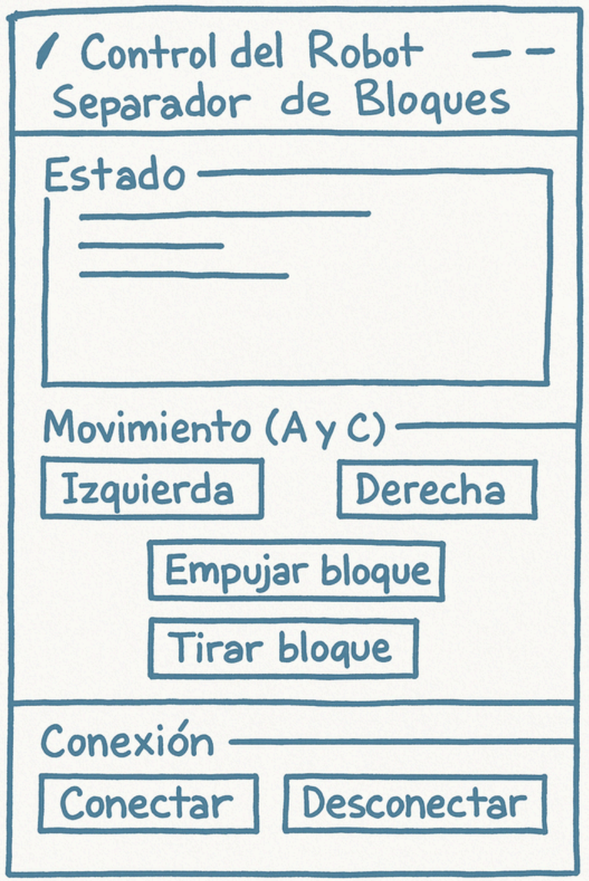


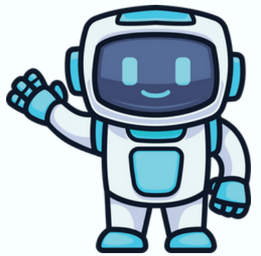
IMPLEMENTACION SERVIDOR

- El cliente genera dinámicamente programas Pybricks.
- Los archivos .py se envían temporalmente al hub.
- El hub ejecuta las instrucciones y luego se eliminan los archivos.
- Se aclaró el rol del servidor como ejecutor de comandos y controlador de motores y sensores.



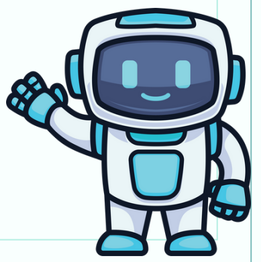
Diseño inicial de la interfaz gráfica de usuario (GUI)





Generación dinámica del programa que ejecuta el robot (create_program)

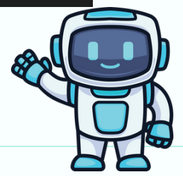
```python
def create_program(drive_cmd: str) -> str:
    drive_commands = {
        'izquierda': "motorC.run_angle(-250, -60)",
        'derecha': "motorC.run_angle(250, -45)",
        'empujar': "motorF.run(500)",
        'tirar': "motorF.run(-500)",
        'leer_color': "",
        'inicio': "motorC.run_target(250, 0)"
    }

    drive_code = drive_commands.get(drive_cmd, "")

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

hub = PrimeHub()

motorC = Motor(Port.C)
motorF = Motor(Port.F)
sensor = ColorSensor(Port.E)

{drive_code}

color = sensor.color()
print("COLOR:", color)

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



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

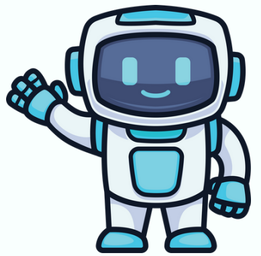
```python
class DeviceSelectWindow(tk.Toplevel):
    def __init__(self, parent, on_select_callback):
        super().__init__(parent)
        self.on_select_callback = on_select_callback

        self.title("Buscador de Hubs")
        self.geometry("400x450")
        self.attributes("-topmost", True)
        self.grab_set()

        ttk.Label(
            self,
            text="Buscando dispositivos...",
            font=("Arial", 14, "bold")
        ).pack(pady=10)

        self.listbox = tk.Listbox(self)
        self.listbox.pack(fill="both", expand=True, padx=10, pady=10)

        ttk.Button(self, text="Escanear", command=self.start_scan).pack(pady=5)
        ttk.Button(self, text="Conectar", command=self.select_device).pack(pady=5)

        self.devices = []
        self.start_scan()
```

```python
    async def connect(self):
        try:
            self.hub = PybricksHubBLE(self.device)
            await self.hub.connect()
            self.running = True
            self.log("🔗 Conectado al hub.")

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

        except Exception as e:
            self.log(f"❌ Error BLE: {e}")

    async def disconnect(self):
        self.running = False
        if self.hub:
            await self.hub.disconnect()
            self.hub = None
            self.log("🔌 Hub desconectado.")
```

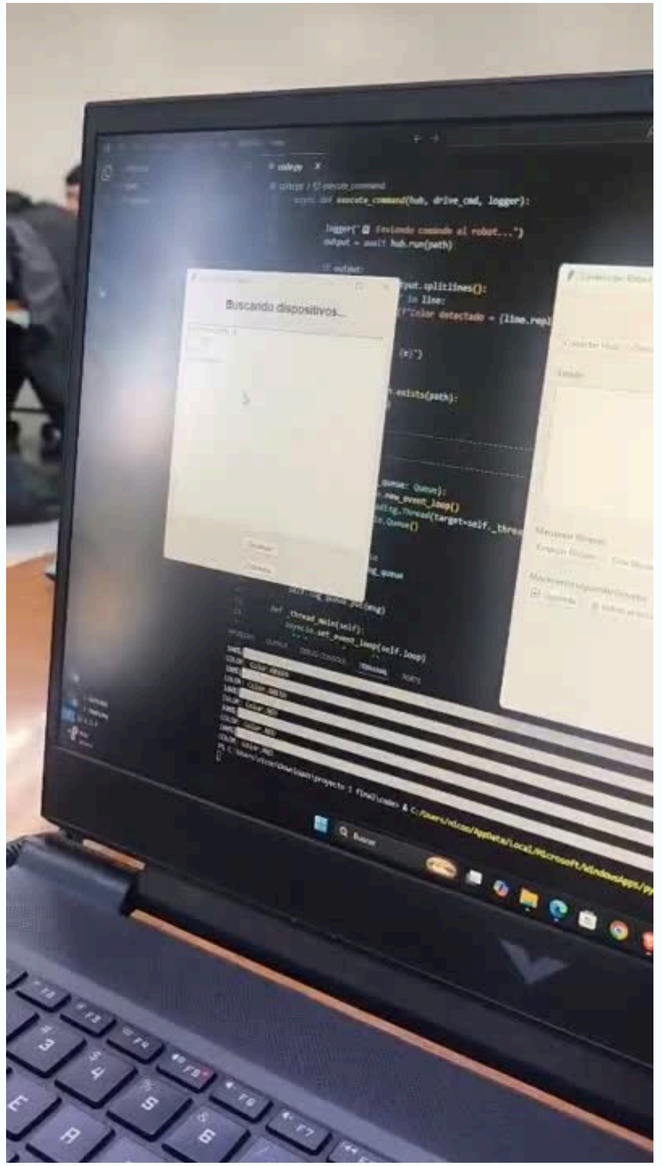


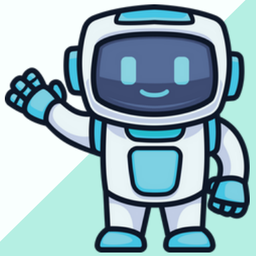
Interfaz gráfica: controles de movimiento (botones)

```
# Los botones de movimiento
frame_left = ttk.LabelFrame(root, text="Movimiento Izquierda/Derecha")
frame_left.pack(fill="both", padx=10, pady=10)

ttk.Button(frame_left, text="⬅ Izquierda", command=lambda: enviar_con_mensaje("izquierda", "⬅ Botón presionado: Izquierda")).grid(row=0, column=0, padx=5)
ttk.Button(frame_left, text="➡ Derecha", command=lambda: enviar_con_mensaje("derecha", "➡ Botón presionado: Derecha")).grid(row=0, column=2, padx=5)
ttk.Button(frame_left, text="🔄 Volver al inicio", command=lambda: enviar_con_mensaje("inicio", "🔄 Botón presionado: Volver al inicio")).grid(row=0, column=1, padx=5)
```

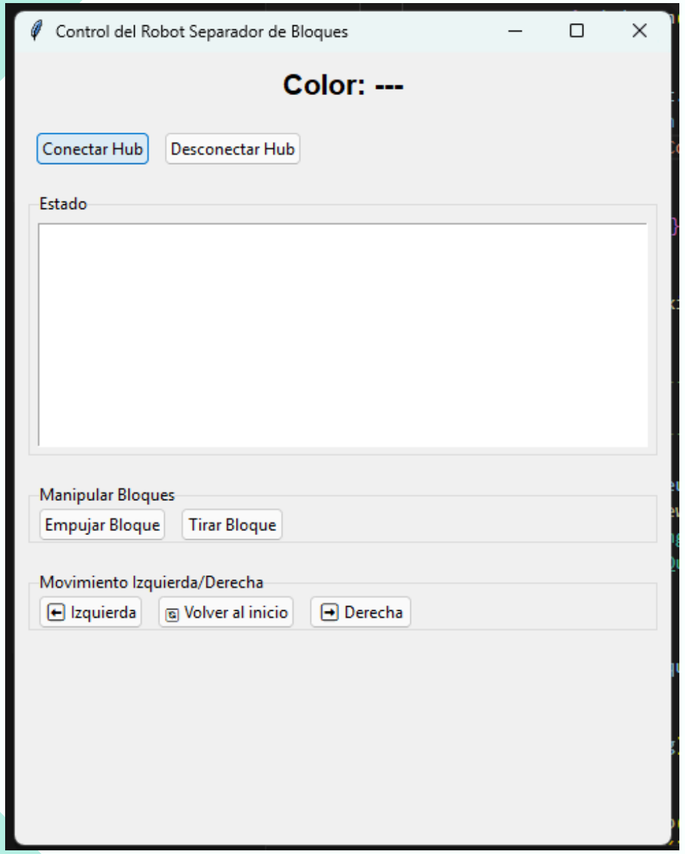


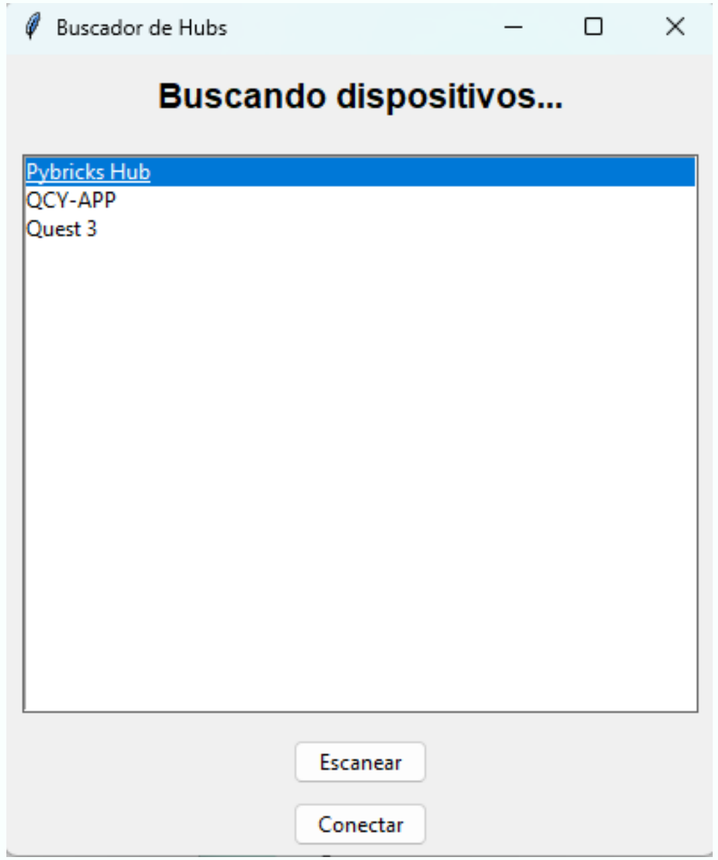
La demo

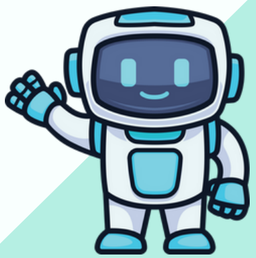




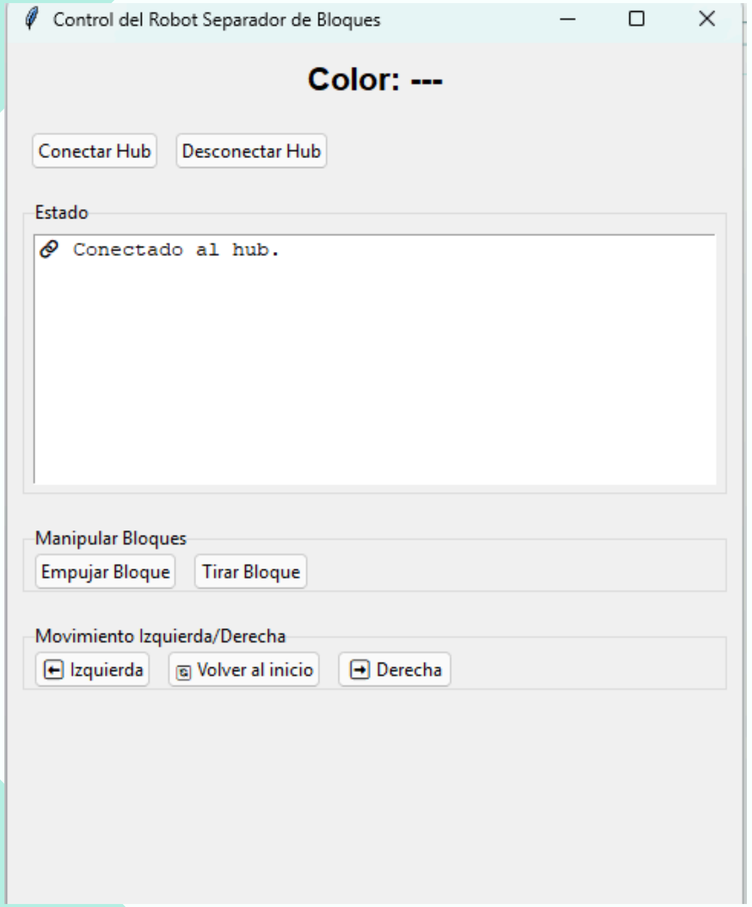
Guía de uso

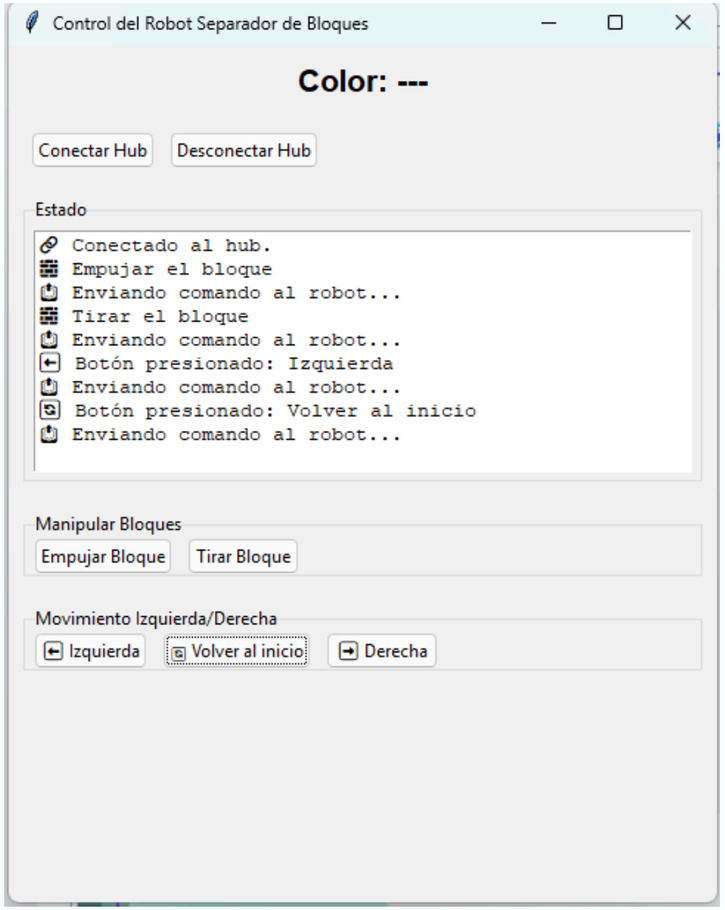

Control del Robot Separador de Bloques
Color: ---
Conectar Hub
Desconectar Hub
Estado
Manipular Bloques
Empujar Bloque
Tirar Bloque
Movimiento Izquierda/Derecha
Izquierda
Volver al inicio
Derecha

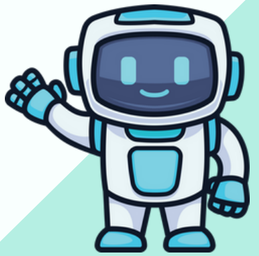

Buscador de Hubs
Buscando dispositivos...
Pybricks Hub
QCY-APP
Quest 3
Escanear
Conectar



Guía de uso

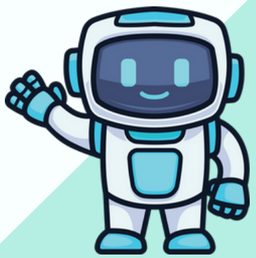

Control del Robot Separador de Bloques
Color: ---
Conectar Hub
Desconectar Hub
Estado
Conectado al hub.
Manipular Bloques
Empujar Bloque
Tirar Bloque
Movimiento Izquierda/Derecha
Izquierda
Volver al inicio
Derecha


Control del Robot Separador de Bloques
Color: ---
Conectar Hub
Desconectar Hub
Estado
Conectado al hub.
Empujar el bloque
Enviando comando al robot...
Tirar el bloque
Enviando comando al robot...
Botón presionado: Izquierda
Enviando comando al robot...
Botón presionado: Volver al inicio
Enviando comando al robot...
Manipular Bloques
Empujar Bloque
Tirar Bloque
Movimiento Izquierda/Derecha
Izquierda
Volver al inicio
Derecha



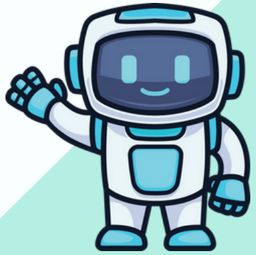
Manual de usuario

https://github.com/nicoog261/-ROBOT-SEPARADOR-DE-BLOQUES-/tree/main



C O N C L U S I Ó N

En conclusion en el proyecto se utilizó una arquitectura cliente-servidor, donde el servidor físico corresponde al hub LEGO SPIKE PRIME, encargado de ejecutar el firmware y controlar directamente los motores y sensores del robot. El servidor lógico está representado por el software Pybricksdev junto con Pybricks, el cual se ejecuta en el computador y se encarga de generar los programas, gestionar la comunicación Bluetooth y coordinar la ejecución de las instrucciones en el hub. Esta separación permitió una comunicación clara entre el usuario y el sistema físico, facilitando el control y mantenimiento del robot.



MUCHAS
GRACIAS