



# MAQUINARIA DE CLASIFICACIÓN DE MATERIALES

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

Asignatura: Proyecto I

Docente: Baris Klobertanz

# Contenidos



- Introducción
- Problema a resolver
- Objetivo General
- Objetivo Especifico
- Estructura Organizacional
- Carta Gantt
- Gestión de Riesgos
- Fundamentos de los Movimientos
- Requerimientos
- Arquitectura
- Implementación
- Prueba Funcional
- Manual de Usuario
- Conclusión

# Introducción

En esta presentación veremos los puntos mas importantes que conllevaron todo el proceso del proyecto asignado desde el problema a resolver hasta las conclusiones obtenidas al finalizar el robot.



# Problemática

En la minería, la clasificación de materiales ineficiente provoca retrasos y expone a los trabajadores a entornos de alto peligro, con lo cual genera una necesidad que requiere un sistema que garantice la eficiencia sin arriesgar la integridad humana.

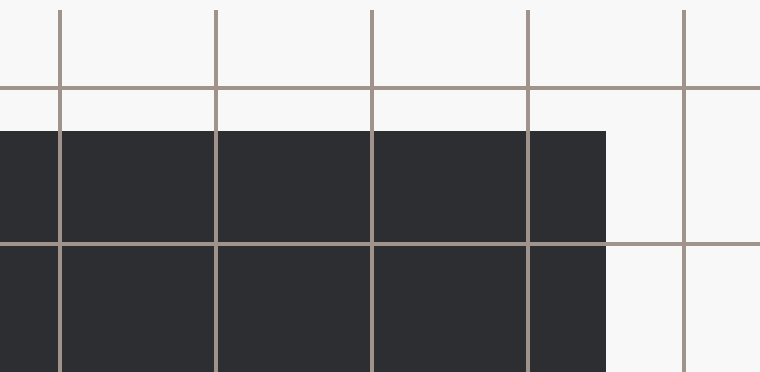
La implementación de esta solución impacta directamente en tres ámbitos clave:

- Mejora la productividad general.
- Optimiza los procesos de clasificación.
- Eleva los estándares de seguridad del equipo.



# Objetivo General

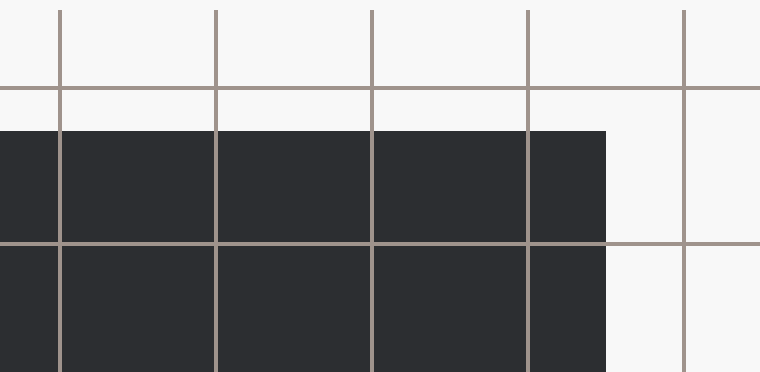
Desarrollar y elaborar un robot con el Set Lego Spike Prime que sea capaz de clasificar un bloque de Lego segun su color en diferentes casillas designadas para cada color, simulando asi la clasificación de materiales de un empresa minera.



# Objetivos Específicos

Los objetivos específicos alineados para cumplir el objetivo general y dar una correcta solución a la problemática planteada anteriormente. Algunos de los principales elegidos para esta sección son:

- Prototipo funcional inicial
- Programación del sistema de clasificación
- Construcción del robot final

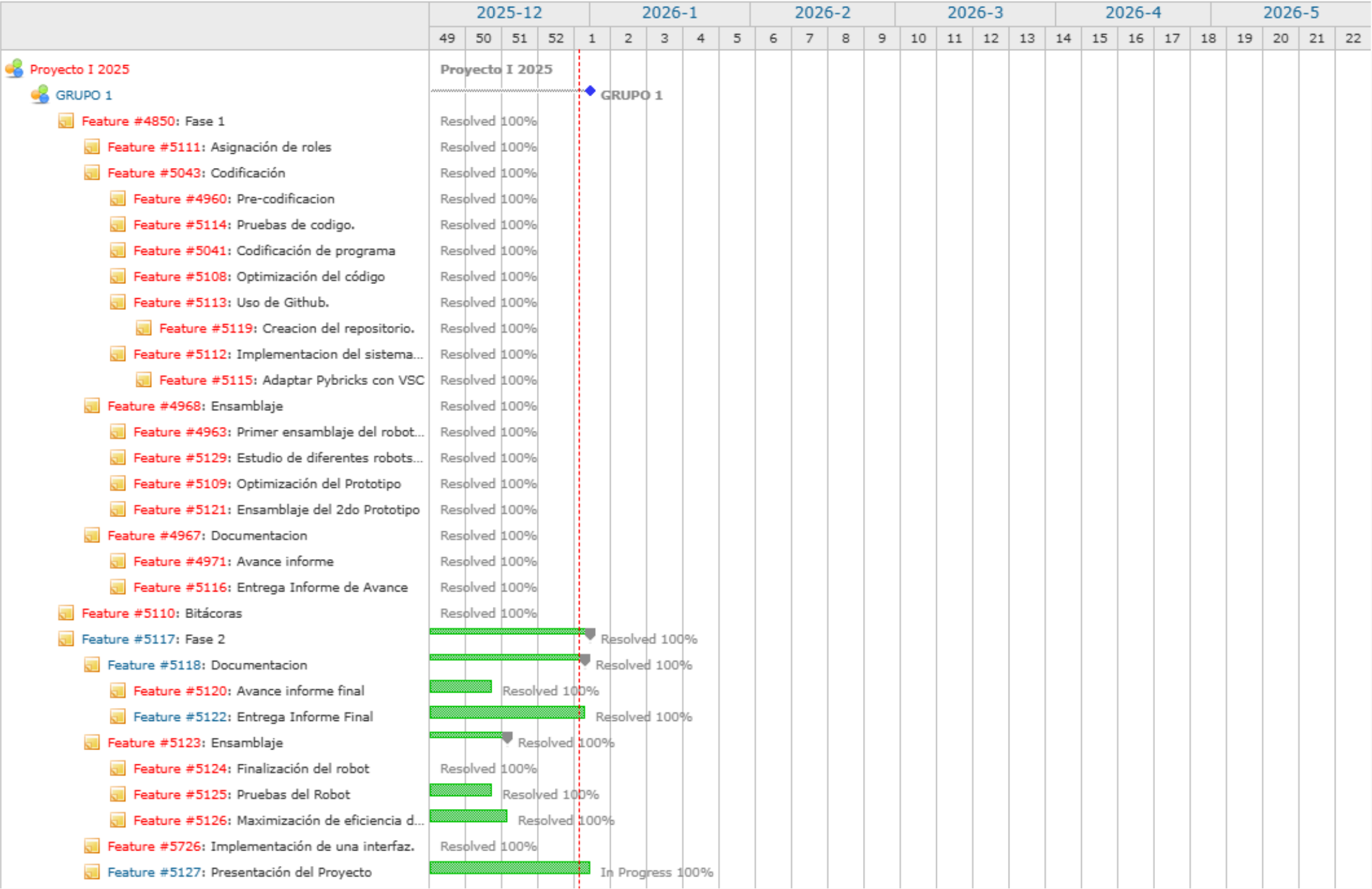


# **Estructura Organizacional**

- **Jefe de Proyecto**
- **Ensamblador**
- **Documentador**
- **Programador**



# Carta Gantt





# Gestión de Riesgos

Objetivo: Garantizar el cumplimiento de los plazos y metas del proyecto.

Función: Identificar, analizar y clasificar los posibles contratiempos que afecten el desarrollo del prototipo robótico y la planificación.



# Gestión de Riesgos



- Impacto Crítico:
  - Máxima Prioridad.
  - Podría causar un reinicio total del proyecto o comprometer la entrega



- Impacto Alto:
  - Gravedad Máxima.
  - Compromete la viabilidad del proyecto.
  - Requiere acciones inmediatas.



- Impacto Medio:
  - Genera retrasos significativos en etapas clave.
  - Requiere respuesta prioritaria para proteger la fecha final.



- Impacto Bajo:
  - Riesgo menor o cotidiano.
  - Se resuelve con acciones simples sin alterar los entregables ni la ruta principal.

# Fundamentos de los movimientos

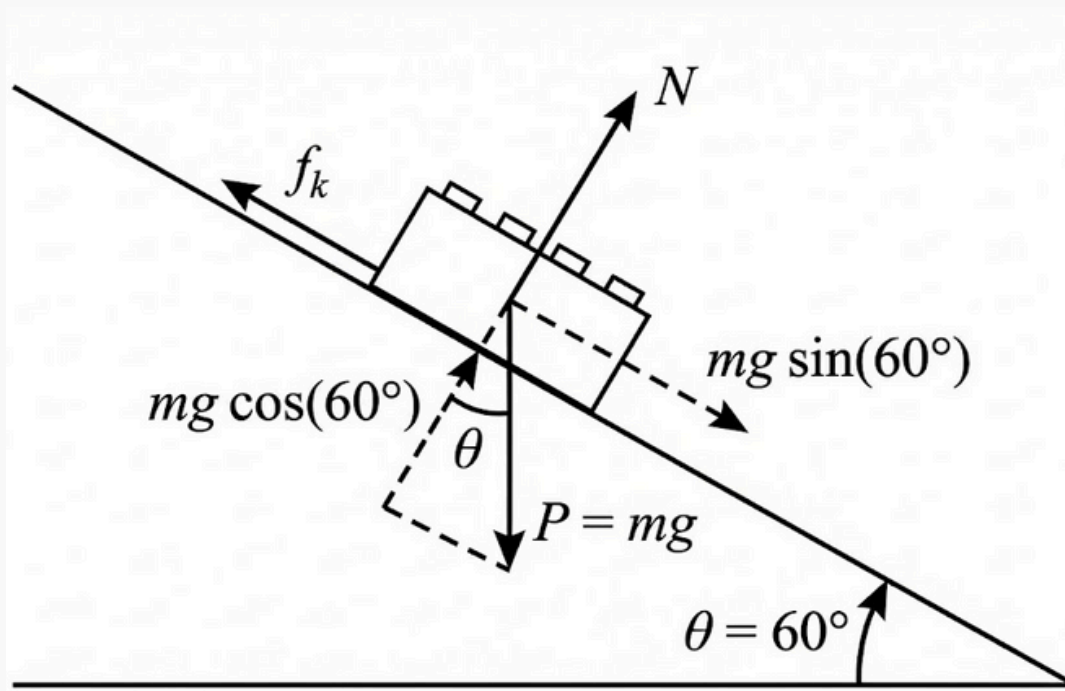
El fundamento de los movimientos responde a la gran incertidumbre si el proceso de cada movimiento del robot es eficiente respecto a tiempo de ejecución y manejo de errores. Esta respuesta se divide en tres respuestas simples para cada tema, los cuales son:

- Caída del bloque
- Eficiencia del motor de empuje y posicionamiento



# Caída del bloque

En esta sección es importante aplicar la segunda ley de newton para evitar errores y optimizar desde donde parte el bloque a clasificar hasta llegar al sensor de detección de color. Se extraen datos como el peso del bloque, el roce, la misma gravedad y el ángulo de la maquinaria para obtener la aceleración misma en la siguiente tabla.



FUERZAS	FORMULAS	RESULTADOS
FUERZA ROCE	$\mu_k * N$	7.44E-3
PX	$P * \sin 60$	1.969E-2
FUERZA NORMAL	$P * \cos 60$	1.137E-2
ACELERACIÓN	$(PX - \text{FUERZA ROCE}) / P$	5.28 m/s**2

# Eficiencia del motor de empuje y posicionamiento

El robot opera en un rango simétrico óptimo de  $35^\circ$  a  $-35^\circ$  en el motor posición. El ángulo se determinó tras validar diferentes ángulos, en un recorrido menor el motor no llegaba a la casilla de ordenamiento y un desplazamiento mayor simplemente era ineficiente por más tiempo de desplazamiento, teniendo en cuenta ello aplicamos la fórmula de Tiempo real de posicionamiento.

Distancia =  $70^\circ$

Velocidad =  $300^\circ/\text{s}$

	FORMULA	RESULTADO
TIEMPO	DISTANCIA/VELOCIDAD	0.23 S

# Eficiencia del motor de empuje y posicionamiento

El motor de empuje posiciona los bloques por color usando un recorrido de  $180^\circ$ , optimizando el tiempo y la precisión. Los cálculos de velocidad y fuerza, basados en  $300^\circ/\text{s}$ , un radio de 4 cm y un torque de  $18 \text{ N}\cdot\text{cm}$ , validan que el sistema puede mover la carga adecuadamente.

	FORMULA	RESULTADO
TIEMPO POR CICLO	DESPLAZAMIENT O/VELOCIDAD	0.6 S
FUERZA DE EMPUJE	TORQUE/RADIO	4.5 N

# Requerimientos Funcionales


Antes de presentar los requerimientos funcionales y requerimientos no funcionales, se debe identificar correctamente al cliente y al usuario, para este contexto del proyecto se identifico como :

- Cliente : El cual es identificado como la empresa minera a la cual se le presenta el proyecto con el fin de resolver la problemática.
- Usuario : Siendo parte del Cliente para este contexto, se puede identificar como un operador de maquinaria perteneciente a la empresa minera. El cual será el encargado de operar y supervisar el robot.



# Requerimientos Funcionales

Los Requerimientos Funcionales son procesos que el robot debe ser capaz de cumplir como requisito base para el correcto cumplimiento de la problemática planteada.

- Correcta identificación del color del bloque de Lego.
  - El código debe procesar la lectura del sensor de color y enviar la orden correcta a los motores según el color detectado.
  - Almacenamiento de los bloques de Lego ya identificados en su respectivo compartimiento.
  - Automatización de los procesos anteriores.
- 
- A decorative graphic in the bottom right corner consisting of several concentric, semi-circular lines in shades of gray, creating a ripple effect.



# Requerimientos no Funcionales

Los requerimientos no funcionales, se pueden definir como atributos de calidad del software, los cuales son esenciales para que los Stakeholders puedan cualificar el proyecto:

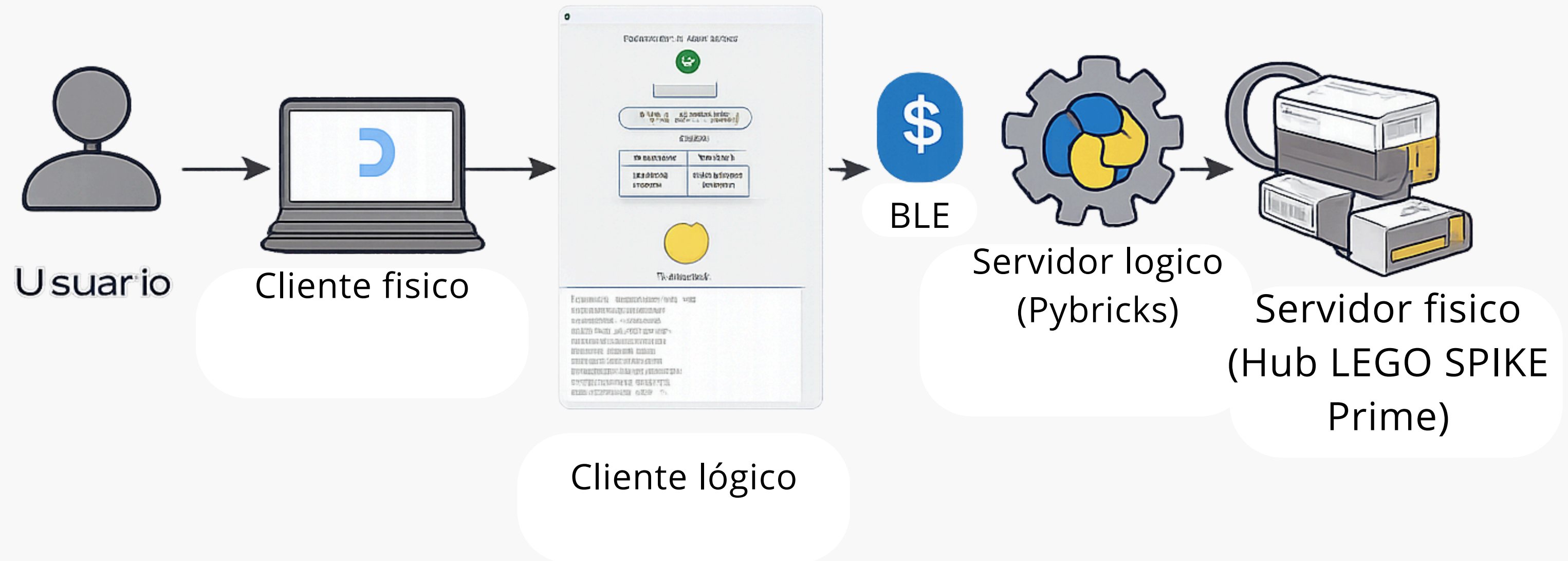
- Robustez:  
Manejo de errores internos y externos.

- Usabilidad :  
Interfaz amigable con el usuario.

- Disponibilidad  
Operación mínima del 98%.

- Rendimiento :  
Capacidad de respuesta de menos de un segundo.

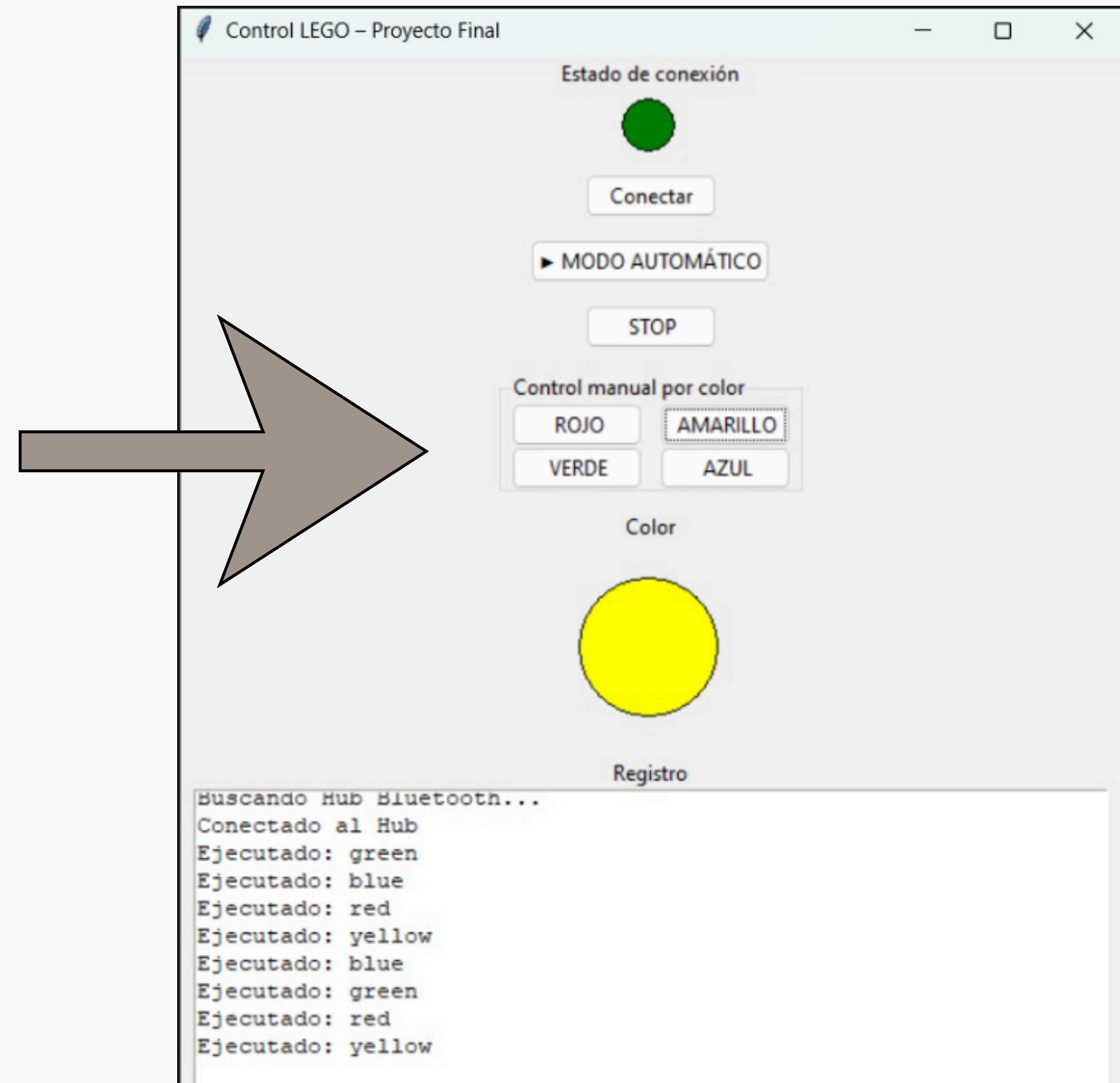
# Arquitectura



# Implementación

Sobre la implementación vamos a tomar de ejemplo el control manual por color.

El usuario puede elegir un color correspondiente a una casilla, en nuestro caso nosotros disponemos de 4 casillas, una para cada color.



Acercas de como se interactua ese botón con el código:

```
180     ttk.Button(frame, text=" ROJO", command=lambda: self.send_color("red")).grid(row=0, column=0, padx=5)
181     ttk.Button(frame, text=" AMARILLO", command=lambda: self.send_color("yellow")).grid(row=0, column=1, padx=5)
182     ttk.Button(frame, text=" VERDE", command=lambda: self.send_color("green")).grid(row=1, column=0, padx=5)
183     ttk.Button(frame, text=" AZUL", command=lambda: self.send_color("blue")).grid(row=1, column=1, padx=5)
```

```
210     def send_color(self, color):
211         self.worker.send(color)
212         self.update_color(color.upper())
```

Mandan un mensaje entre hilos.

```
122     def send(self, cmd):
123         self.loop.call_soon_threadsafe(self.queue.put_nowait, cmd)
```

```
143     while True:
144         cmd = await self.queue.get()
145         await execute(self.hub, cmd, self.log)
```

```
86  async def execute(hub, cmd, log):
87      |      program = create_program(cmd)
88
```

```
17  def create_program(cmd: str) -> str:
18      |
19      |      base = ""
20      |      from pybricks.hubs import PrimeHub
21      |      from pybricks.pupdevices import Motor, ColorSensor
22      |      from pybricks.parameters import Port, Color
23      |      from pybricks.tools import wait
24      |
25      |      hub = PrimeHub()
26      |      motor_pos = Motor(Port.D)
27      |      motor_emp = Motor(Port.A)
28      |      sensor = ColorSensor(Port.B)
29      |      ""
```

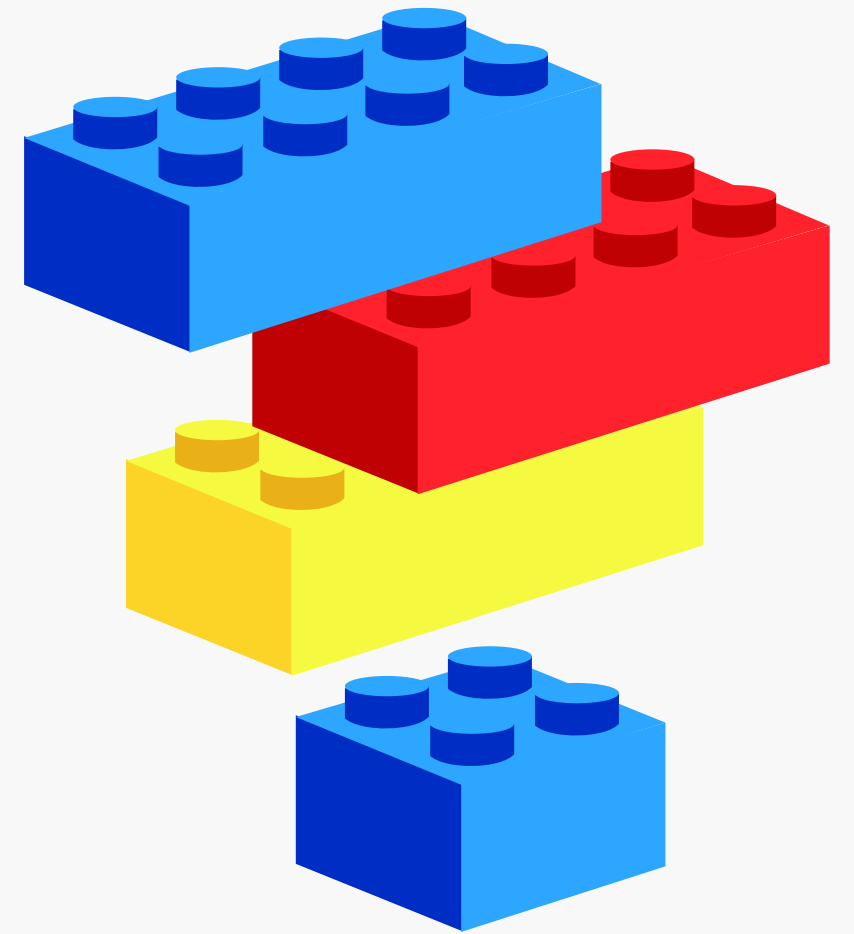
```
72      |      if cmd in commands:
73      |          |      return base + commands[cmd] + "\nwait(100)"
74
```

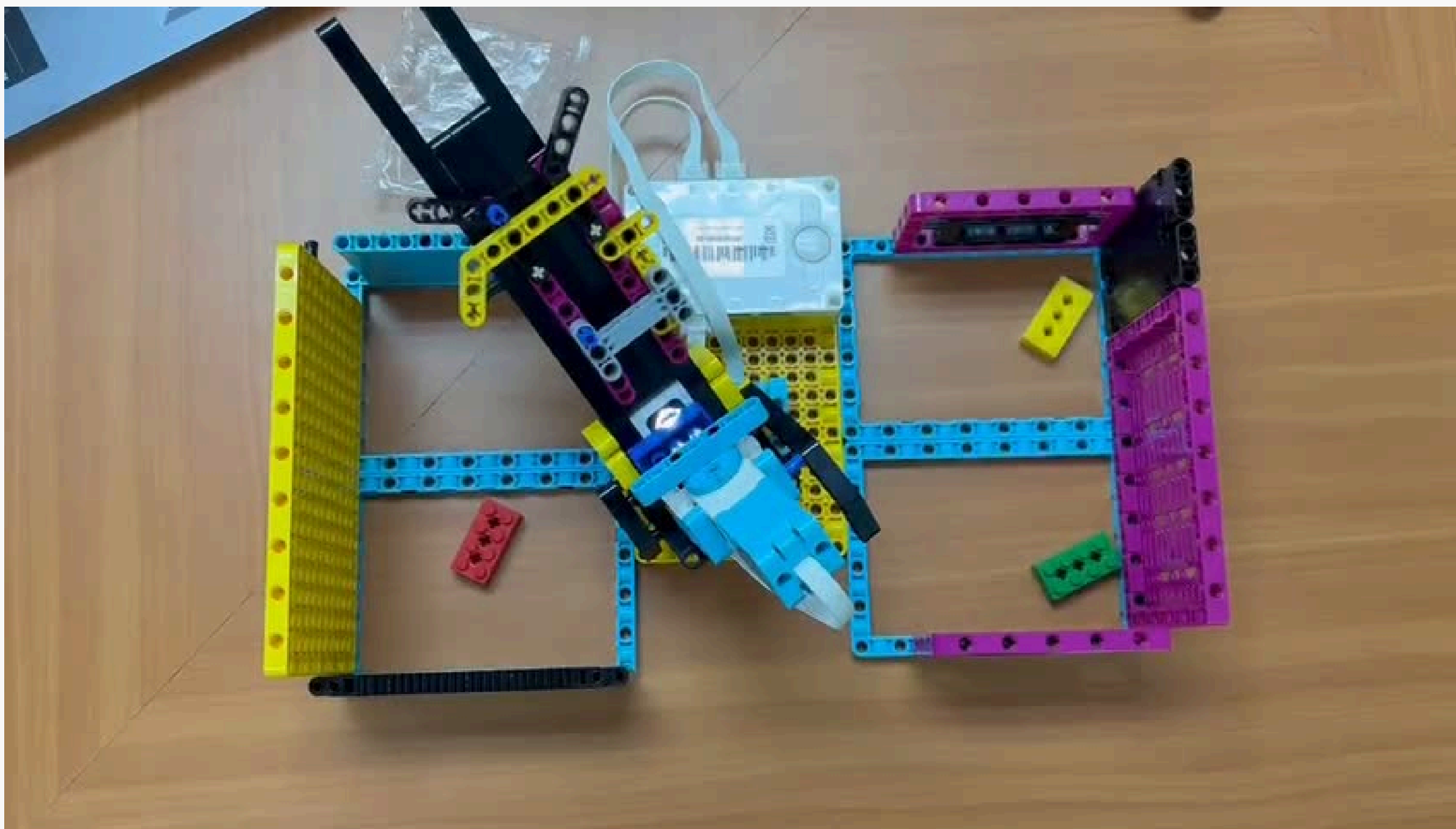
```
61      |      commands = {
62      |          |      "red": "print('COLOR:RED'); motor_pos.run_target(300,-35); motor_emp.run_angle(1000,-180)",
63      |          |      "yellow": "print('COLOR:YELLOW'); motor_pos.run_target(300,-35); motor_emp.run_angle(1000,180)",
64      |          |      "green": "print('COLOR:GREEN'); motor_pos.run_target(300,35); motor_emp.run_angle(1000,180)",
65      |          |      "blue": "print('COLOR:BLUE'); motor_pos.run_target(300,35); motor_emp.run_angle(1000,-180)",
66      |          |      "emp_up": "motor_emp.run_angle(1000,180)",
67      |          |      "emp_down": "motor_emp.run_angle(1000,-180)",
68      |          |      "pos_right": "motor_pos.run_target(300,35)",
69      |          |      "pos_left": "motor_pos.run_target(300,-35)",
70      |      }
```



# Prueba mínima funcional

El robot Lego debe ser capaz de identificar y almacenar correctamente cada material al compartimiento correspondiente de ese material.





# Manual de Usuario

A continuación les mostraremos nuestro manual de usuario realizado en el README.md, que se encuentra en nuestro repositorio de Github.

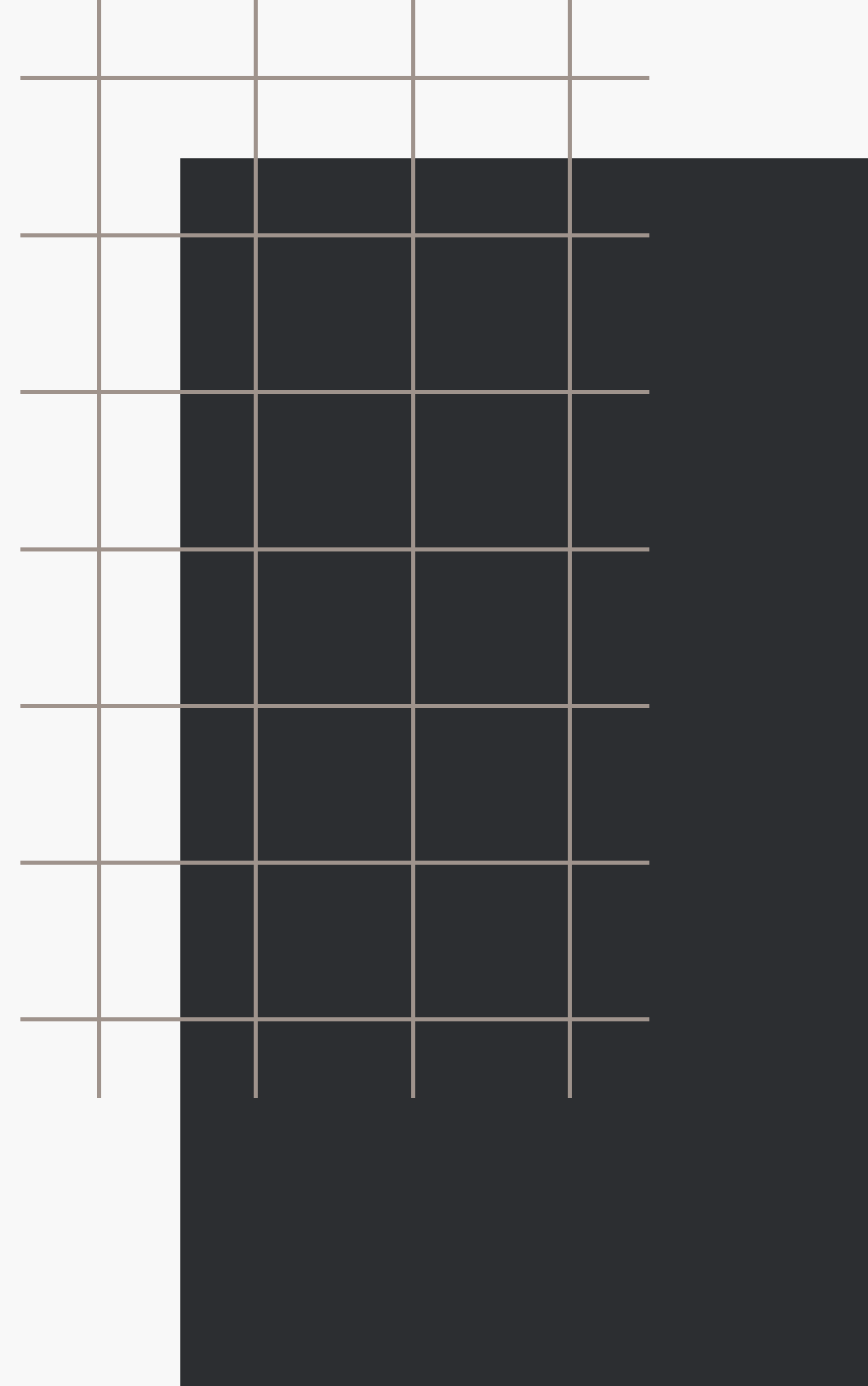
Link de GitHub:

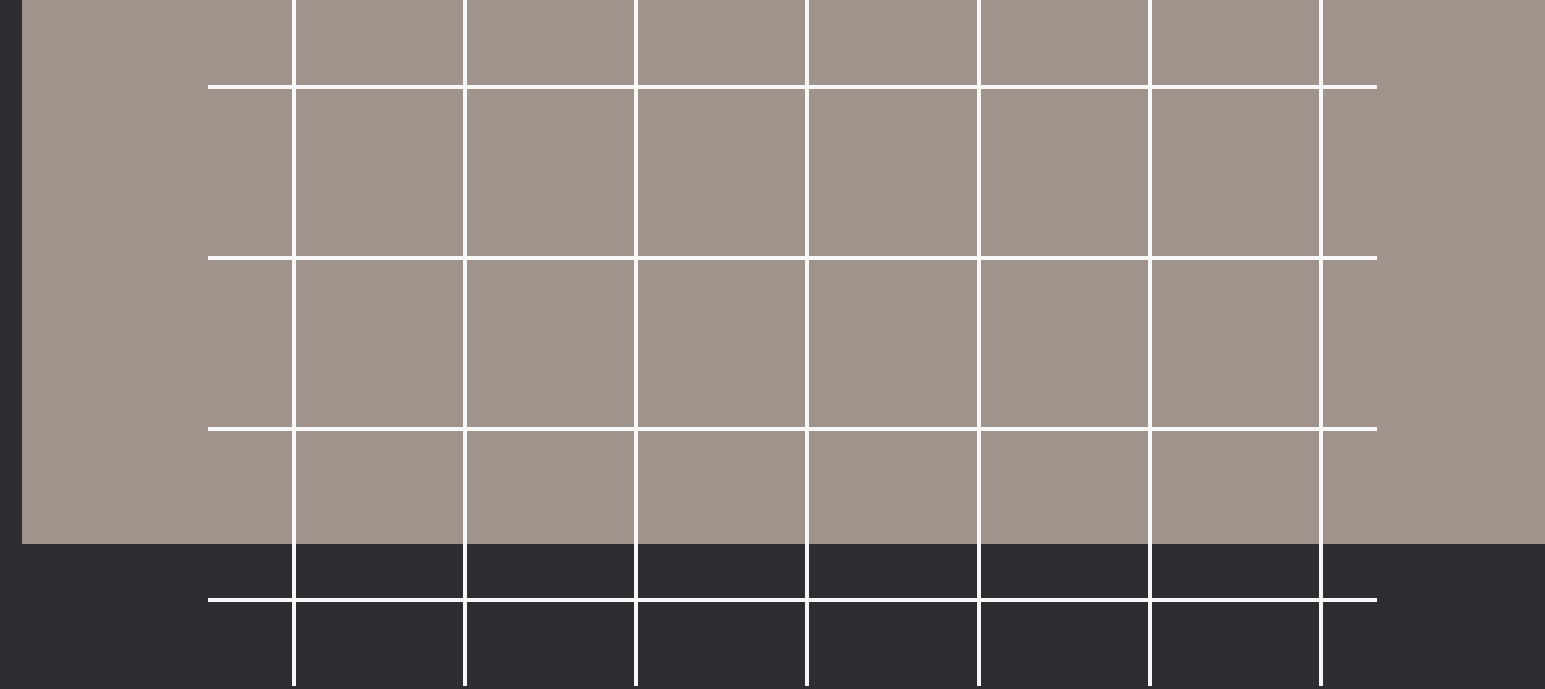
<https://github.com/saoudahmedlanchipa-code/LegoSpikePrimeProyecto1Sorting>





# CONCLUSIÓN





**MUCHAS GRACIAS**

