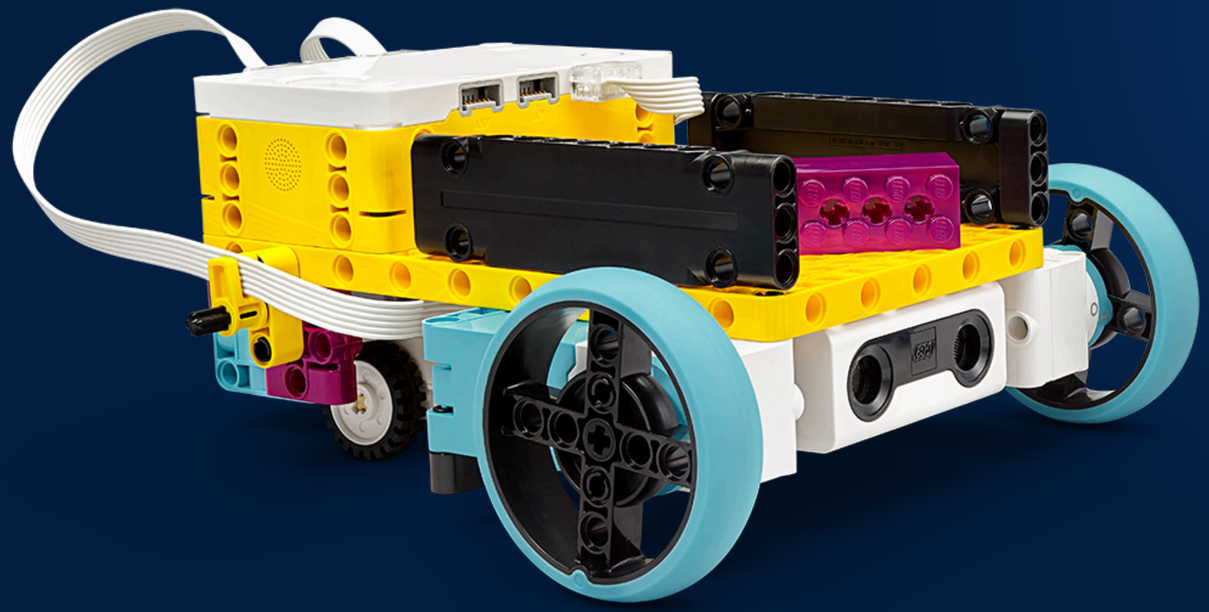
Ingeniería
Computación e Informática

F.L.E.T.E.

Facilitador Logístico Eléctrico Transportador de Equipo

Asignatura: Proyecto I



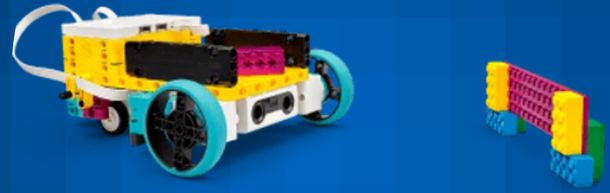
Profesor: Baris Klobertanz Quiroz

Alumnos: Martin Acevedo Tudela
Pablo Andia Lopéz
Felipe Diaz Araos
Edynson Tola Fernandez
José Yampara Yampara

CONTENIDO

1 INTRODUCCIÓN

El proyecto F.L.E.T.E. simula el transporte automatizado de mineral en minería subterránea utilizando el kit LEGO Spike Prime y control remoto, con el objetivo de mejorar la productividad y el rendimiento de los transportes de cargamento pesado en las mineras subterráneas. El proyecto también busca crear un prototipo que optimice y garantice la seguridad del empleador y de los empleadores mientras el vehículo esté en funcionamiento.



2 OBJETIVO GENERAL

El proyecto tiene como objetivo diseñar y programar un prototipo robótico utilizando exclusivamente el hardware del kit LEGO Spike Prime, el cual ejecutará la simulación de transporte automatizado de carga (bloques de lego). Este sistema será operado mediante una interfaz gráfica desarrollada en Python, con la finalidad de demostrar la optimización de procesos logísticos y la eficiencia de la automatización en las mineras subterráneas.

3 OBJETIVOS ESPECÍFICOS

- Estudiar los componentes del set Lego Spike Prime.
- Proponer y seleccionar el mejor prototipo para el proyecto.
- Construir el prototipo seleccionado.
- Investigar y analizar las librerías disponibles en Python, evaluando la compatibilidad con el set LEGO SPIKE Prime, para asi poder tener un control completo del robot.
- Desarrollar las funciones del robot (avance, retroceso, giro y frenado), teniendo como base algoritmos eficientes y funciones que optimicen el control de los movimientos básicos del robot.
- Diseñar e implementar una interfaz gráfica en Python, mediante la librería Tkinter, que permita recibir y gestionar las instrucciones destinadas al robot.

4 ESTRUCTURA ORGANIZACIONAL

JEFE DE PROYECTO

Representante del equipo, supervisa, organiza el progreso del proyecto y gestiona las reuniones grupales (dailys).

ENSAMBLADOR

Encargado del montaje, armado del robot y del conteo de piezas del set Lego Spike Prime.

PROGRAMADOR

Encargado del área de la codificación y funcionamiento del robot.

DOCUMENTADOR

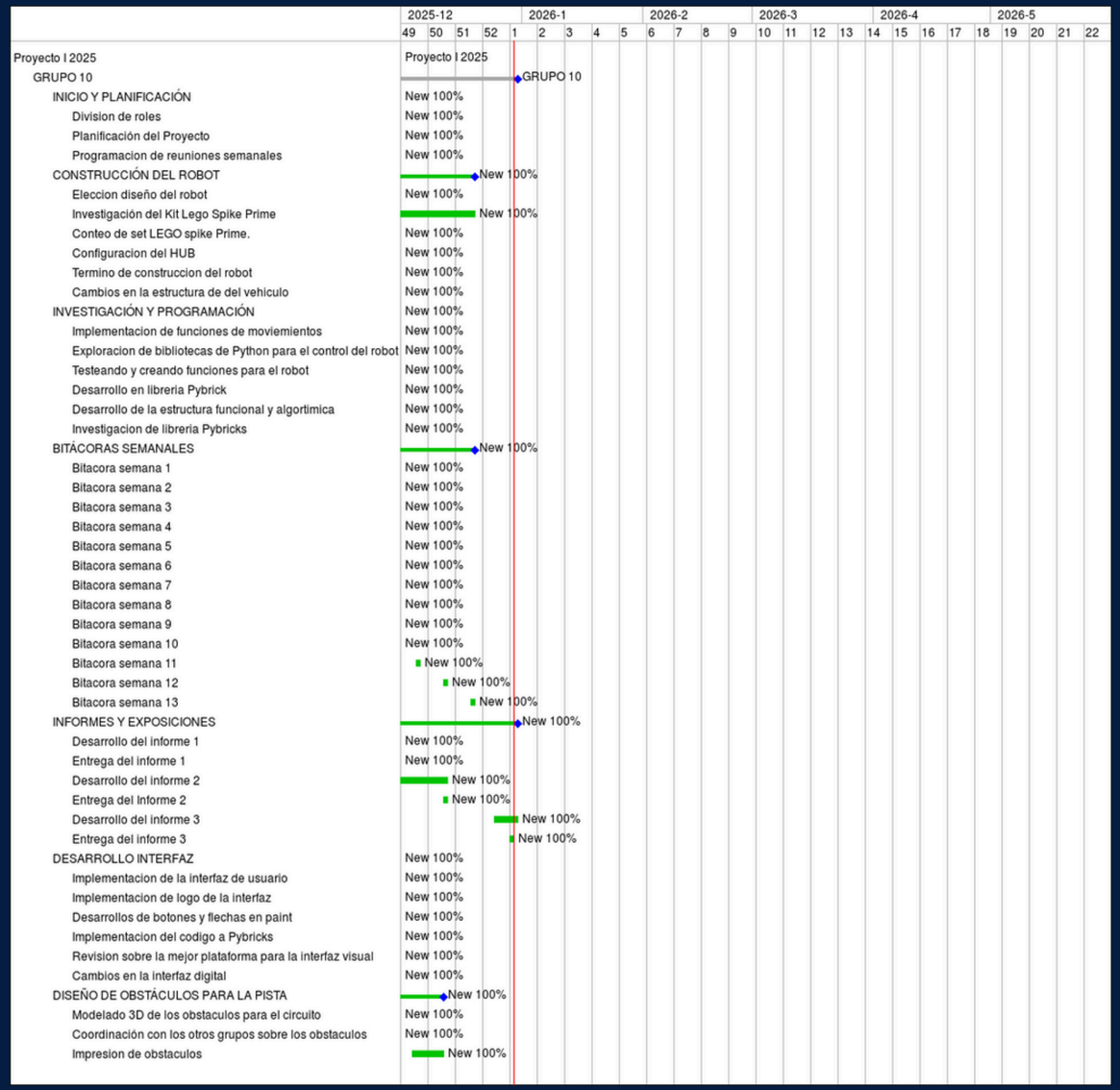
Encargado de realizar las bitácoras semanales, llevar una retroalimentación sobre las dificultades y actividades a realizar cada semana. También es el encargado de realizar el informe.

DISEÑADOR

Encargado de la creación de la interfaz (GUI), presentaciones y logos o imágenes necesarias para el correcto desarrollo del proyecto.

5

CARTA GANTT



6 GESTIÓN DE RIESGOS

Daño momentáneo : Riesgo menor que no para el desarrollo del proyecto. Generalmente tiene solución inmediata. (1)
Daño menor : Riesgo de poca importancia pero que es reiterativo. Puede solucionarse en cualquier momento.(2)
Daño relevante : Riesgo que retrasa el correcto desarrollo del proyecto, se debe resolver a la mayor brevedad posible.(3)
Daño Crítico : Riesgos que deben solucionarse de forma inmediata, de lo contrario puede provocar la detención del proyecto.(4)

6

GESTIÓN DE RIESGOS

Riesgo	Nivel de Gravedad	Acción Remedial
Personal faltando al horario asignado de trabajo	4	Preguntar la causa de su ausencia, para poder gestionar otra reunión en horarios disponibles
Falla de registro en la plataforma redmine	4	Comunicar al administrador de la página para encontrar una solución.
Recibir equipo defectuoso	4	Conseguir un reemplazo del equipo con el encargado de las piezas
Horario insuficiente para el cumplimiento de tareas en conjunto	3	Coordinación de una reunión fuera de clases, en el mejor horario posible para cada integrante
Desempeño del robot no es eficiente	2	Utilizar conocimientos propios para reconocer las fallas o de última instancia buscar información en línea
Incomprensión de fallo con bibliotecas	2	Buscar ejemplos en medios oficiales para solucionar el problema
Error en la codificación	2	Investigar el origen del error e intentar resolver el problema buscando información en internet
Atraso en el cumplimiento de tareas	2	Priorizar tareas más importantes para agilizar la productividad del proyecto. Agendar reuniones en horarios libres si es necesario
Ausencia de piezas	1	Verificar si fue ausencia de fábrica o error de algún integrante, y buscar la pieza perdida
Dificultades con la conexión wifi	1	Usar cable ethernet o compartir red por datos móviles

7

PROBLEMAS ENCONTRADOS Y SOLUCIONADOS

Frecuencia	Riesgos	Acción remedial
20	Error en la codificación	Investigar el origen del error e intentar resolver el problema buscando información en internet
15	Dificultades con la conexión wifi	Usar cable ethernet o compartir red por datos móviles
8	Personal faltando al horario asignado de trabajo	Preguntar la causa de su ausencia, para poder gestionar otra reunión en horarios disponibles
7	Incomprensión de fallo con bibliotecas	Buscar ejemplos en medios oficiales para solucionar el problema
3	Atraso en el cumplimiento de tareas	Priorizar tareas más importantes para agilizar la productividad del proyecto. Agendar reuniones en horarios libres si es necesario
1	Vehículo se mueve con dificultad o es incapaz de moverse libremente	Probar a intercambiar dentro de lo posible piezas que generan el problema

8

FUNDAMENTO DE LOS MOVIMIENTOS

Giro de las ruedas según diferentes velocidades

Datos iniciales

Radio: 2 cm = 0,02 m
Diámetro: 4 cm = 0,04m
Circunferencia de la rueda = 0,1257m
velocidad lenta = 400 °/s
velocidad normal = 800 °/s
velocidad Sport = 1500 °/s

$$C = 2\pi r$$

$$\omega = \frac{\omega°/s}{} \cdot \frac{\pi}{180}$$

$$v = \omega\, r$$

$$f = \frac{\omega°/s}{360}$$

$$RPM = 60\, f$$

$$T = \frac{1}{f}$$

1ª Lenta – 400 °/s
Maniobras de precisión y estacionamiento
Velocidad angular: 6,98 rad/s
Velocidad lineal: 0,14 m/s (≈ 14,0 cm/s)
Frecuencia de giro: 1,11 rev/s
Velocidad de rotación: 66,7 RPM
Periodo de una vuelta: 0,90 s

2ª Normal – 800 °/s
Velocidad crucero por defecto
Velocidad angular: 13,96 rad/s
Velocidad lineal: 0,28 m/s (≈ 27,9 cm/s)
Frecuencia de giro: 2,22 rev/s
Velocidad de rotación: 133,3 RPM
Periodo de una vuelta: 0,45 s

3ª Sport – 1500 °/s
Desplazamientos rápidos en tramos rectos
Velocidad angular: 26,18 rad/s
Velocidad lineal: 0,52 m/s (≈ 52,4 cm/s)
Frecuencia de giro: 4,17 rev/s
Velocidad de rotación: 250 RPM
Periodo de una vuelta: 0,24 s

8 FUNDAMENTO DE LOS MOVIMIENTOS

Velocidad y aceleración del robot

1ª Marcha – LENTA (400 °/s)
Maniobras de precisión y estacionamiento
Velocidad
Velocidad angular: 6,98 rad/s
Velocidad lineal: 0,14 m/s (14 cm/s)
Aceleración
Aceleración angular: 13,96 rad/s²
Aceleración lineal: 0,28 m/s²

2ª Marcha – NORMAL (800 °/s)
Velocidad crucero
Velocidad
Velocidad angular: 13,96 rad/s
Velocidad lineal: 0,28 m/s (27,9 cm/s)
Aceleración
Aceleración angular: 27,9 rad/s²
Aceleración lineal: 0,56 m/s²

3ª Marcha – SPORT (1500 °/s)
Desplazamientos rápidos en tramos rectos
Velocidad
Velocidad angular: 26,18 rad/s
Velocidad lineal: 0,52 m/s (52,4 cm/s)
Aceleración
Aceleración angular: 52,4 rad/s²
Aceleración lineal: 1,05 m/s²

Marcha	Velocidad (m/s)	Aceleración(m/s²)
Lenta	0,14	0,28
Normal	0,28	0,56
Sport	0,52	1,05

$$v = \omega \cdot r$$

$$a = \frac{\omega}{t}$$

REQUERIMIENTOS FUNCIONALES

RF1 Movilidad: El robot debe poder moverse en todas las direcciones y detenerse.

RF2 Transporte: Se transportan objetos en su contenedor.

RF3 Ejecución de órdenes: El robot debe ser capaz de responder las órdenes ingresadas por el usuario.

RF4 Indicador de estado de conexión: Se debe indicar en la GUI el estado de conexión del robot.

RF5 Velocidad: Se debe poder modificar la marcha del motor e indicar esta.

10 REQUERIMIENTOS NO FUNCIONALES

RNF1 Disponibilidad: El robot debe poseer la facultad de funcionar por a lo menos 1 hora continua.

RNF2 Robustez: El sistema debe estar programado para gestionar de forma efectiva el 90% de los errores y seguir funcionando.

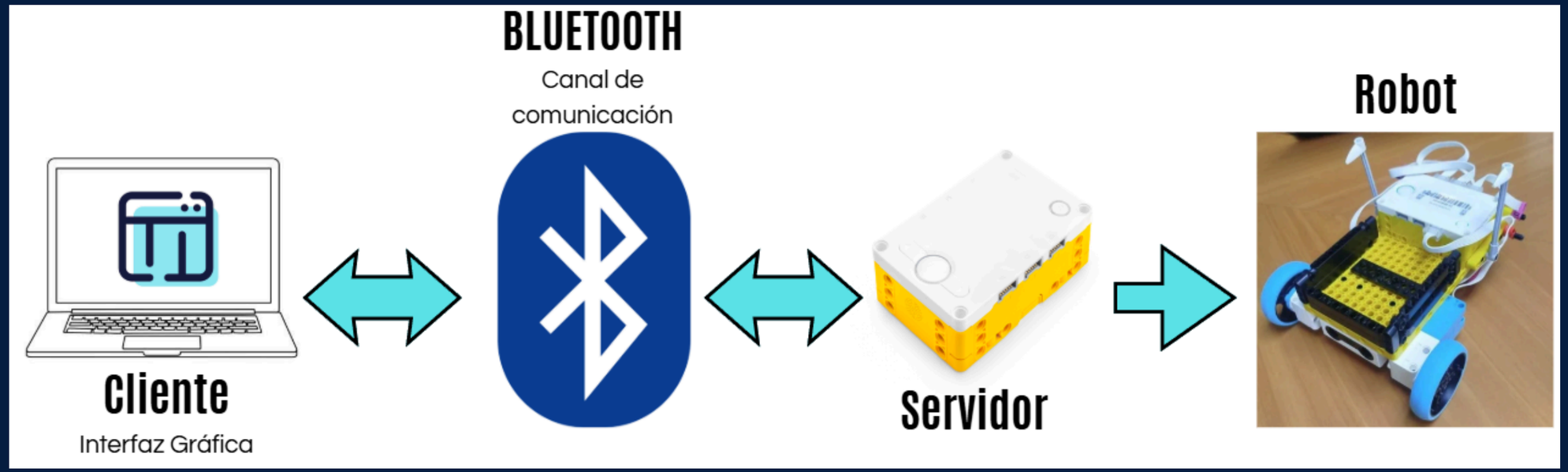
RNF3 Rendimiento: El sistema debe ser capaz de cumplir las ordenes del usuario con una latencia de no más de 1 segundo.

RNF4 Usabilidad: La GUI debe ser intuitiva y fácil de usar, para que el 100% de los usuarios pueda realizar una tarea designada tras una breve inducción.

RNF5 Componentes: Los componentes del robot deben estar en buena condición y completamente funcionales para efectuar acciones con normalidad el 99.9% del tiempo.

ARQUITECTURA

La arquitectura está basada en el modelo cliente-servidor.



12

IMPLEMENTACIÓN: ARQUITECTURA Y FUNDAMENTOS

OBJETIVO FÍSICO

Alcanzar una velocidad mínima de 0,5 m/s (10 metros en 20 segundos) para simular eficiencia en faena.

MODELO CLIENTE-SERVIDOR

Ejecución remota donde el PC (Cliente) procesa la lógica pesada y el Hub (Servidor) ejecuta las órdenes físicas.

COMUNICACIÓN

Transmisión de instrucciones serializadas vía Bluetooth Low Energy (BLE).

LÓGICA DEL CLIENTE: CONEXIÓN ASÍNCRONA

- El ordenador ejecuta la interfaz en Python.
- **Multithreading**: Se implementó un hilo secundario (WorkerBluetooth) para gestionar la conexión sin congelar la interfaz visual.
- Cola de Datos: Uso de asyncio.Queue para el envío seguro de paquetes.

```python
class WorkerBluetooth(threading.Thread):
    def __init__(self, log_queue, status_callback):
        super().__init__(daemon=True) # Hilo independiente
        self.loop = asyncio.new_event_loop()
        self.queue = asyncio.Queue()  # Cola de mensajes

    async def _main_logic(self):
        # ... (código de conexión) ...
        while True:
            cmd = await self.queue.get() # Espera órdenes
            if self.connected:
                await self.hub.write(cmd.encode()) # Envía al robot
```

LÓGICA DEL SERVIDOR: ESCUCHA ACTIVA

- **Inyección de Código**: Al conectar, el sistema transfiere el script CODIGO_ROBOT a la memoria RAM del Hub.
- **Procesamiento en Tiempo Real**: El robot ejecuta un bucle infinito con lectura no bloqueante (uselect).
- **Control de Puertos**: Gestión exclusiva de motores A, E (Tracción) y C (Dirección).

```python
while True:
    # Lectura sin bloqueo para alta velocidad
    ready = uselect.select([sys.stdin], [], [], 0)[0]
    if ready:
        char = sys.stdin.read(1)
        if char == ';':
            # Decodificación de comandos
            if action == 'F': motorA.run(speed) # Forward
            elif action == 'L': motor_dir.run_target(...) # Left
    wait(5) # Ciclo de 5ms
```

INTERFAZ DE CONTROL (GUI)

- Desarrollada con librería **Tkinter**
- **Panel Superior:** Indicador de estado (Offline/Online) y gestión de conexión segura.
- **Control Híbrido:** Permite operación táctil (botones en pantalla) o física (teclado WASD).
- **Feedback Visual:** Los botones se iluminan al recibir acciones para confirmar la orden

GESTIÓN DINÁMICA DE VELOCIDAD

Implementación de **3 Marchas Virtuales** seleccionables por software:

- **1 Lenta (400°/s):** Precisión.
- **2 Normal (800°/s):** Crucero.
- **3 Sport (1500°/s):** Velocidad.

Al cambiar de marcha (Shift/Ctrl), el sistema actualiza la variable global y reconfigura los comandos de movimiento al instante.

```
1  self.velocidades_list = [400, 800, 1500]
2
3  def actualizar_velocidad(self):
4      velocidad = self.velocidades_list[self.marcha_index]
5      self.velocidad_actual = velocidad
6
7      # Actualización dinámica de comandos
8      self.btn_avanzar.on_press_cmd = f"F{velocidad};"
9      self.btn_retro.on_press_cmd = f"B{velocidad};"
10     # ...
```

OPTIMIZACIÓN DE LATENCIA Y ESTADO

- **Problema:** El envío constante de teclas (ej: mantener 'W') satura el Bluetooth.
- **Solución:** Implementación de un Evaluador de Estado.
- **Filtro Anti-Spam:** El código compara la orden actual con la anterior (last_traccion). Si son iguales, no envía nada. Solo transmite cuando hay un cambio de estado real.

```
# 3. ENVIAR SOLO SI CAMBIÓ EL ESTADO (Anti-Spam)
if cmd_traccion != self.last_traccion:
    self.enviar(cmd_traccion)
    self.last_traccion = cmd_traccion

if cmd_direccion != self.last_direccion:
    self.enviar(cmd_direccion)
    self.last_direccion = cmd_direccion
```

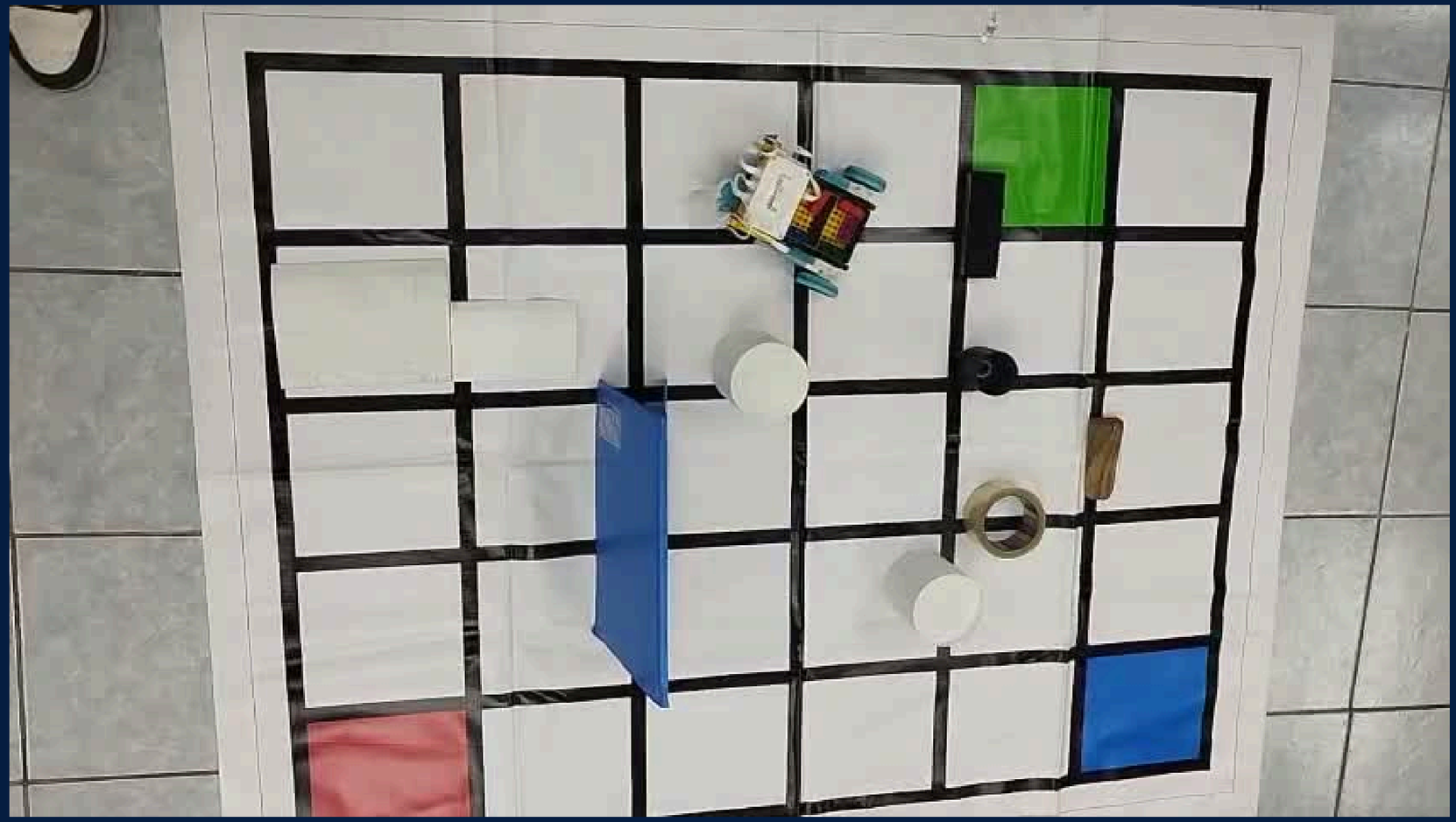
FUNCIONES AVANZADAS Y REGISTRO

- Accesos Rápidos: Macros que combinan tracción y dirección en un solo paquete de datos (ej: "L;B800;" para curva atrás izquierda).
- Consola (Log): Sistema de monitoreo en tiempo real que permite visualizar la serialización de comandos y depurar la conexión.

```
# Macros de Maniobras Complejas
self.btn_za.on_press_cmd = f"L;B{velocidad};" # Curva Atrás Izq
self.btn_ca.on_press_cmd = f"R;B{velocidad};" # Curva Atrás Der
```

13

PRUEBA MINIMA FUNCIONAL (DEMO)



14 DESPLIEGUE DEL SISTEMA

- **Repositorio**: El código fuente se encuentra alojado y versionado en GitHub para facilitar la colaboración y mantenimiento.

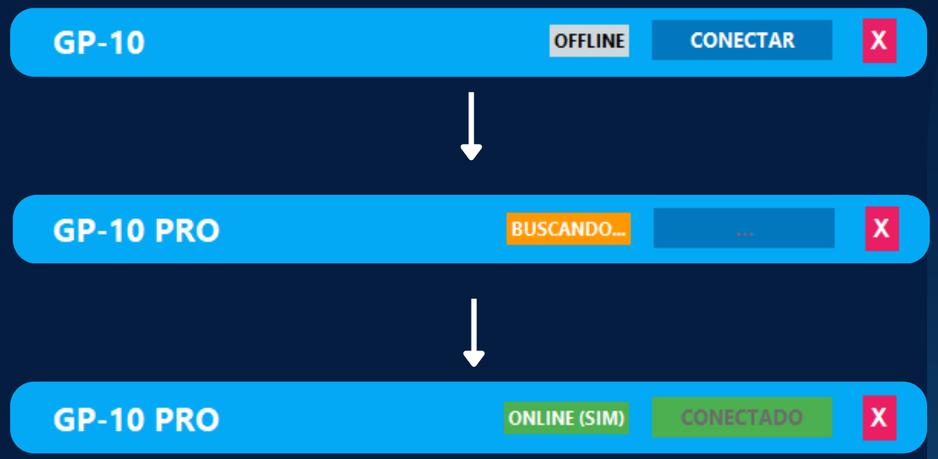
- **Clonación del Proyecto**:
  - Comando: git clone git clone https://github.com/Pabloyera/ProyectoFLETE.git

- **Gestión de Dependencias**:
  - Instalación de librerías: pip install bleak pybricksdev tk

- **Ejecución**:
  - Inicio del cliente: python Robot.py

PROCEDIMIENTO DE CONEXIÓN SEGURA

- **Iniciar:** Presionar el botón azul "CONECTAR".
- **Verificación:** Observar el indicador de estado.
  - Naranja: Buscando dispositivo.
  - Amarillo: Inyectando código al robot.
  - Verde (ONLINE): Listo para operar.
- **Confirmación:** Esperar el mensaje en consola "Conexión lista".



CONTROLES DE OPERACIÓN

- **Mapa de Teclas:**
  - Movimiento: W (Adelante), S (Atrás), A/D (Dirección).
  - Acción: Espacio (Bocina), Q/E (Curvas Rápidas).
  - Caja de Cambios (Gestión de Potencia):
  - Shift ⬆️: Subir Marcha (Más velocidad).
  - Ctrl ⬇️: Bajar Marcha (Más precisión).

- **Seguridad:** Al soltar cualquier tecla, el sistema activa el frenado automático (Auto-Stop).

15

CONCLUSIÓN

El proyecto permitió comprender de manera practica cómo la automatización y el control remoto pueden mejorar significativamente a mejorar la seguridad en entornos laborales. Logrando mejorar la productividad y eficiencia del transporte de cargas.

Mediante el uso del LEGO Spike Prime y una interfaz desarrollada con Python y Tkinter, logramos simular un sistema de transporte que minimice la intervención humana en zonas peligrosas, y así prevenir accidentes en situaciones laborales, ya sea en obras u otro tipo de trabajo relacionado con la construcción.

Como mejoras futuras, se considera la incorporación de sensores adicionales para aumentar el nivel de automatización y seguridad del sistema, permitiendo la detección de obstáculos y un control más preciso del movimiento. Además, se plantea mejorar la interfaz gráfica mediante nuevas opciones de configuración y visualización de datos en tiempo real, con el objetivo de optimizar la supervisión y el control del robot.