UNIVERSIDAD DE TARAPACÁ
Universidad del Estado

Ingeniería Computación e Informática

PROTOTIPO DE UN VEHÍCULO ROBOT MINERO PARA EL TRASLADO DE MINERALES A GRAN ESCALA

Asignatura: Proyecto I

Carrera: Ingeniería Civil en Computación e Informática

Docente: Baris Klobertanz

Integrantes:

- Ayleen Humire
- Brandon Quispe
- German Castro
- Claudio Pinazo
- Daniela Poma

Diciembre 2025

CONTENIDOS

1. CONTEXTO Y PROBLEMA

Contexto del proyecto

La minería 4.0 busca modernizar los procesos productivos mediante automatización, robótica e inteligencia artificial.
Sin embargo, en la minería subterránea aún existen altos niveles de riesgo para los trabajadores, especialmente en tareas de carga y transporte de material.
Accidentes ocurridos en minas como El Teniente (Chile) y Grasberg (Indonesia) evidencian la necesidad de reducir la exposición humana a zonas peligrosas mediante soluciones tecnológicas.

1. CONTEXTO Y PROBLEMA

Problema identificado

La operación minera presenta riesgos significativos para los trabajadores debido a la presencia humana en zonas de alto peligro.

La falta de soluciones accesibles para simular y probar alternativas robóticas limita el desarrollo de sistemas que reduzcan estos riesgos.

Importancia de resolverlo:

- Mejora de la seguridad laboral
- Reducción de accidentes
- Exploración de soluciones robóticas aplicables a minería real

2. OBJETIVOS DEL PROYECTO

Objetivo general

Diseñar y construir un robot utilizando el kit LEGO Spike Prime que sea capaz de desplazarse, integrando principios de mecánica y programación, con el propósito de desarrollar competencias técnicas en robótica educativa y fomentar el trabajo colaborativo, además de simular la reducción de riesgos en zonas de minería, beneficiando a los trabajadores.

2. OBJETIVOS DEL PROYECTO

Objetivos específicos

- Diseñar la estructura mecánica del robot utilizando el kit LEGO Spike Prime.
- Ensamblar físicamente el robot integrando chasis, ruedas, HUB y motores.
- Programar el sistema de control para permitir el desplazamiento del robot.
- Evaluar el funcionamiento del robot mediante pruebas y ajustes.

3. ORGANIZACIÓN DEL EQUIPO

Primera fase (semanas 1-6):

- Jefe de proyecto: German Castro
- Ensamblador: Claudio Pinazo
- Programador: Brandon Quispe
- Documentador: Ayleen Humire
- Diseñador: Daniela Poma

Segunda fase (semanas 7-14):

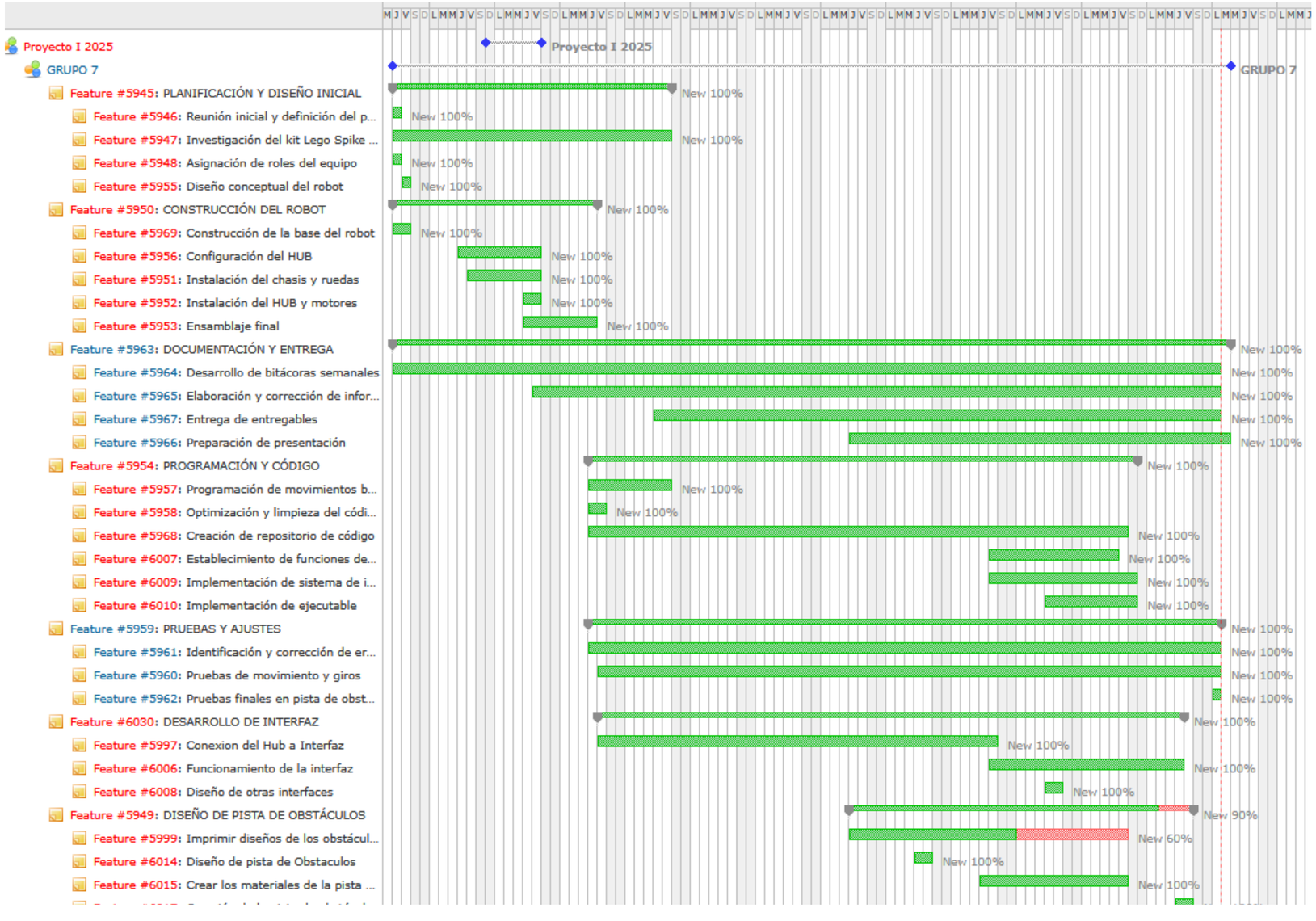
- Jefe de proyecto: Ayleen Humire
- Ensamblador: Daniela Poma
- Programador: Claudio Pinazo
- Documentador: German Castro
- Diseñador: Brandon Quispe

3. ORGANIZACIÓN DEL EQUIPO

Estructura organizacional del proyecto

- **Jefe de proyecto:** supervisa y organiza el progreso del proyecto, verifica el cumplimiento de los plazos y coordina al equipo.
- **Ensamblador:** encargado del montaje y armado del robot, verificando su funcionamiento físico y colaborando con el programador.
- **Programador:** responsable de la codificación del robot y del desarrollo de la interfaz de control.
- **Documentador:** registra el avance del proyecto y redacta los informes.
- **Diseñador:** encargado del diseño visual del proyecto y de las pruebas de campo del vehículo.

4. PLANIFICACIÓN DEL PROYECTO



4. PLANIFICACIÓN DEL PROYECTO

Carta Gantt

La Carta Gantt permitió visualizar la planificación temporal del proyecto, considerando:

- Actividades durante todo el semestre
- Dependencias entre tareas
- Hitos importantes como implementación, pruebas y entrega

Los hitos principales fueron:

- Programación del sistema cliente-servidor
- Desarrollo de la interfaz gráfica
- Construcción de la pista de obstáculos
- Preparación de la presentación final

5. GESTIÓN DE RIESGOS

Los riesgos fueron categorizados según su impacto:

1. Daño catastrófico
2. Daño crítico
3. Daño circunstancial
4. Daño irrelevante
5. Daño recurrente

Riesgo	Nivel de Impacto	Acción Remedial
Retraso en el cumplimiento de tareas.	2	Notificar al equipo sobre el retraso y usar horas extraordinarias acordadas de forma previa con cada miembro. De esta forma se compensará el tiempo perdido y no afectará al progreso.
Fallo en la batería o en la fuente de alimentación del Hub/Spike Prime	2	Verificar el estado de carga de la batería antes de cada sesión. Tener una batería de repuesto o un cargador disponible en el aula/espacio de trabajo.
Pérdida de un archivo clave o corrupción del código fuente	2	Implementar un sistema de control de versiones (ej. Git) para todo el código, permitiendo revertir a versiones anteriores y compartir el código de forma segura entre los miembros.
Ausencia de piezas del Kit LEGO Spike Prime.	1	Solicitar las piezas faltantes al administrador de los materiales o, en caso de no disponibilidad, diseñar y fabricar los repuestos mediante impresión 3D.
Horario insuficiente para el cumplimiento de tareas en conjunto.	1	Reorganizar los horarios de trabajo grupal, establecer reuniones con anticipación y distribuir las tareas individualmente.
Falla crítica del computador principal (El que contiene el código final o el entorno de desarrollo)	1	Realizar copias de seguridad incrementales del código y el entorno de desarrollo en un servicio en la nube (ej. Google Drive, GitHub) o en un disco duro externo al final de cada sesión de trabajo.

6. FUNDAMENTOS DE LOS MOVIMIENTOS

¿Qué aceleración necesita el robot para recorrer 10 metros en 5 segundos?

Principio físico aplicado: Movimiento Rectilíneo Uniformemente Acelerado (MRUA)

1. **Fórmula:**

$$d = v_0 t + \frac{1}{2} a t^2$$

2. **Valores iniciales:**

- Distancia (d) = 10 metros
- Velocidad inicial (v_0) = 0 m/s
- Tiempo (t) = 5 segundos

3. **Cálculo:**

$a = 2d/t^2$

$a = 2(10\ m)/(5\ s)^2$

$a = 20/25$

$a = 0.8\ m/s^2$

4. **Valor resultante:**

Se requiere una aceleración de 0.8 m/s

Aplicación al diseño:

- Distribución optimizada del centro de masa.
- Reducción de masa total eliminando piezas innecesarias.
- Selección de diámetro de rueda que equilibra velocidad y torque.

7. REQUERIMIENTOS DEL SISTEMA

Usuario y Cliente

Usuario: Operadores de maquinaria minera y personal técnico encargado del transporte de minerales en zonas de alto riesgo.

Cliente: Empresas mineras y departamentos de seguridad industrial que buscan reducir la exposición humana mediante automatización y control remoto.

Requerimientos Funcionales

RF1 - Movimiento multidireccional: El robot se mueve en 4 direcciones (adelante, atrás, izquierda, derecha).
RF2 - Control de potencia: La GUI permite determinar la fuerza de los motores.
RF3 - Interfaz gráfica: El robot proporciona una GUI para control del usuario.
RF4 - Acceso mediante ejecutable: La GUI se accede desde un archivo .exe.
RF5 - Recepción de órdenes: El robot recibe y ejecuta órdenes desde la GUI.
RF6 - Parada de emergencia: Detención inmediata ante situaciones de peligro.

7. REQUERIMIENTOS DEL SISTEMA

Requerimientos no Funcionales

RNF1 - Disponibilidad: Funcionamiento continuo durante al menos 2 horas sin interrupciones.
RNF2 - Robustez: Manejo correcto de fallos sin afectar el rendimiento. Recuperación del control en menos de 10 segundos.
RNF3 - Rendimiento: Respuesta rápida a órdenes con interacción fluida y sin demoras.
RNF4 - Usabilidad: Interfaz intuitiva. Usuario nuevo puede ejecutar funciones básicas en menos de 5 minutos.
RNF5 - Compatibilidad: Sistema ampliable para integrar módulos adicionales sin degradar más del 10% el rendimiento.
RNF6 - Formato ejecutable: GUI ejecutable mediante archivo .exe sin configuraciones complejas.

8. ARQUITECTURA DEL SISTEMA

Modelo Cliente-Servidor

Componentes principales:

Cliente (Interfaz Gráfica)

- Tecnología: Python con Custom Tkinter
- Funciones:
  - Renderizar interfaz visual
  - Capturar eventos de teclado (WASD) y clics
  - Gestionar control de potencia
  - Establecer conexión con el Hub
  - Enviar comandos continuos
  - Implementar parada de emergencia

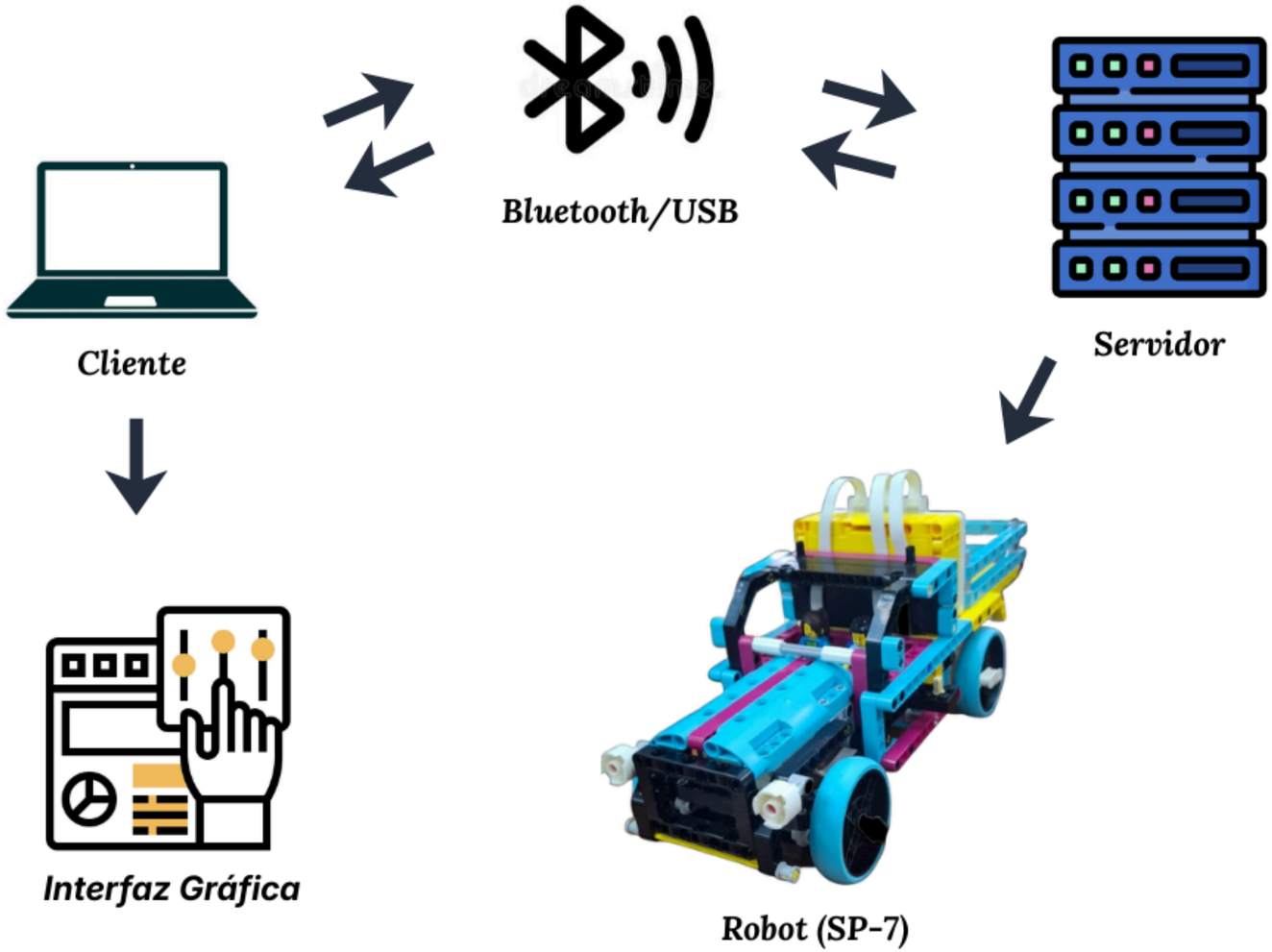
Servidor (Hub LEGO Spike Prime)

- Tecnología: Micro-Python
- Funciones:
  - Escuchar conexiones del cliente
  - Controlar servomotores
  - Ajustar potencia según parámetros
  - Ejecutar acciones con baja latencia

Canal de Comunicación

- Protocolo: sobre Bluetooth/ USB
- Conexión persistente con flujo continuo
- Latencia menor a 1 segundo

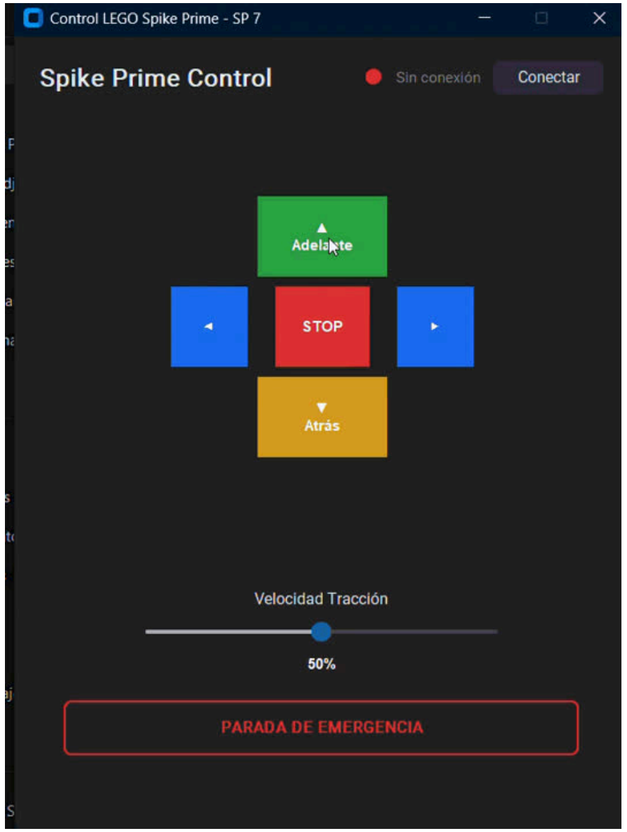
8. ARQUITECTURA DEL SISTEMA



9. IMPLEMENTACIÓN

Flujo de ejecución: Movimiento hacia adelante

1. Usuario presiona tecla "W" o el botón "Adelante" en la GUI




```
self.btn_up = tk.Button(dpad_container, text="▲\nAdelante", bg=self.color_green,
                        activebackground="#268c3b", width=14, height=4, **btn_style)
self.btn_up.grid(row=0, column=1, pady=10)
self.btn_up.bind("<ButtonPress-1>", lambda e: self.cmd_move("F"))
self.btn_up.bind("<ButtonRelease-1>", lambda e: self.cmd_stop_traction())
```

```
if key == "w":
    self.cmd_move("F")
    self.btn_up.configure(bg=self.btn_up.cget('activebackground'), relief='sunken')
```

- Es la parte visual del sistema.
- Permite al usuario interactuar con el robot mediante botones, sliders u otros controles.
- No controla directamente el hardware; solo genera acciones (ej.: avanzar, girar, detener).
- Envía estas acciones al cliente.

9. IMPLEMENTACIÓN

Flujo de ejecución: Movimiento hacia adelante

2. Cliente (interfaz.py)

```
def cmd_steer(self, action):
    if self.worker.running.is_set():
        self.worker.send_packet(action)

Windsurf: Refactor | Explain | Generate Docstring | ×
def cmd_move(self, direction):
    pct = self.slider.get()
    speed = int(pct * 10)
    cmd = f"{direction}{speed}"
    if self.worker.running.is_set():
        self.worker.send_packet(cmd)

Windsurf: Refactor | Explain | Generate Docstring | ×
def cmd_stop_traction(self, event=None):
    if self.worker.running.is_set():
        self.worker.send_packet("S")
```

- Recibe las órdenes desde la interfaz gráfica.
- Traduce estas órdenes a comandos comprensibles para el robot.
- Gestiona la comunicación con el servidor.

3.Bluethooh (Conexion.py)

```
async def _runner(self):
    temp_path = None
    try:
        self.log("Buscando hub 'SP-7'...")
        device = await find_device("SP-7")
        if not device:
            self.log("No se encontró hub.")
            return
```

```
def start(self):
    if not self.thread.is_alive():
        self.thread.start()

def stop(self):
    if self.loop.is_running():
        for task in asyncio.all_tasks(self.loop):
            task.cancel()
        self.loop.call_soon_threadsafe(self.loop.stop)

def send_packet(self, text_cmd: str):
    if self.loop.is_running():
        self.loop.call_soon_threadsafe(self.queue.put_nowait, text_cmd)
```

9. IMPLEMENTACIÓN

Flujo de ejecución: Movimiento hacia adelante

4. Servidor (controlmotores.py):

```
# -- CONFIGURACIÓN --
hub = PrimeHub()
hub.light.on(Color.ORANGE)

motor_der = None
motor_izq = None
motor_dir = None # Motor de dirección (Puerto C)

# Inicialización de motores de tracción
try:
    motor_der = Motor(Port.A)  # Rueda derecha
except Exception: pass

try:
    motor_izq = Motor(Port.E)  # Rueda izquierda
except Exception: pass

# Inicialización del motor de dirección
try:
    motor_dir = Motor(Port.C)
    motor_dir.reset_angle(0)
except Exception: pass
```

```
elif action == 'F' or action == 'B':
    try:
        val_part = cmd[1:]
        if val_part == '': val_part = '0'
        speed = int(val_part)

        if action == 'B':
            speed = -speed

        if motor_der: motor_der.run(speed)
        if motor_izq: motor_izq.run(-speed)
        hub.light.on(Color.BLUE)
    except ValueError:
        pass
```

9. IMPLEMENTACIÓN

Flujo de ejecución: Movimiento hacia adelante

5. Robot

```
hub = PrimeHub()

motor_der = Motor(Port.A)    # Rueda derecha
motor_izq = Motor(Port.E)    # Rueda izquierda
motor_dir = Motor(Port.C)    # Dirección

# Variables iniciales
Speed = 0
giro = 0
      You, last month • actualizacion del uuid …
# BUCLE PRINCIPAL DE COMANDOS

while True:

    cmd = sys.stdin.readline()

    if not cmd:
        wait(10)
        continue
    cmd = cmd.strip()
```

10. PRUEBA FUNCIONAL

Descripción de la prueba mínima funcional

Circuito de prueba:

- Matriz de 6x5 sobre alfombrilla proporcionada.
- Punto de entrada y salida definidos.
- Obstáculos: rampas, cilindros, conos y objetos implementados por los integrantes.

Criterios de éxito:

- Completar el recorrido desde inicio hasta término
- Superar todos los obstáculos sin perder la carga
- Mantenerse dentro de los límites de la pista
- Responder correctamente a todos los comandos GUI
- Conexión estable durante mínimo 5 minutos
- Parada de emergencia funcional

10. PRUEBA FUNCIONAL

Resultados de la prueba

Requerimientos validados:

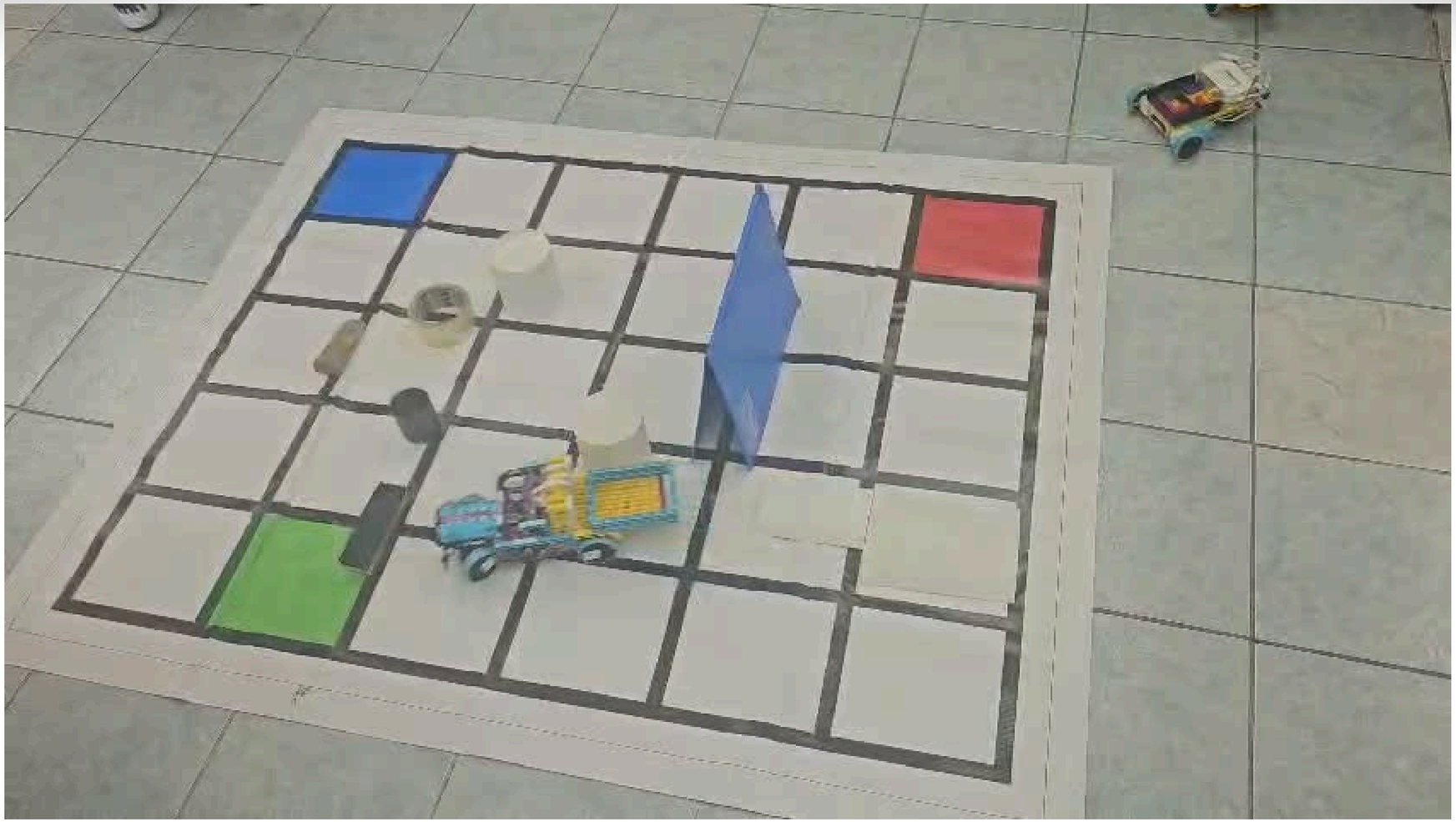
- **RF1:** Movimiento multidireccional funcional.
- **RF2:** Control de potencia mediante slider.
- **RF3 y RF5:** Control mediante GUI operativo.
- **RF6:** Parada de emergencia inmediata.
- **RNF3:** Latencia menor a 1 segundo.
- **RNF4:** Interfaz intuitiva y fácil de usar.

Observaciones:

- Robot superó rampas, cilindros y conos exitosamente.
- Salidas menores de los límites en curvas cerradas.
- Comunicación estable y responsiva.
- Sistema cliente-servidor funcional.

10. DEMOSTRACIÓN EN VIDEO

Vista general:

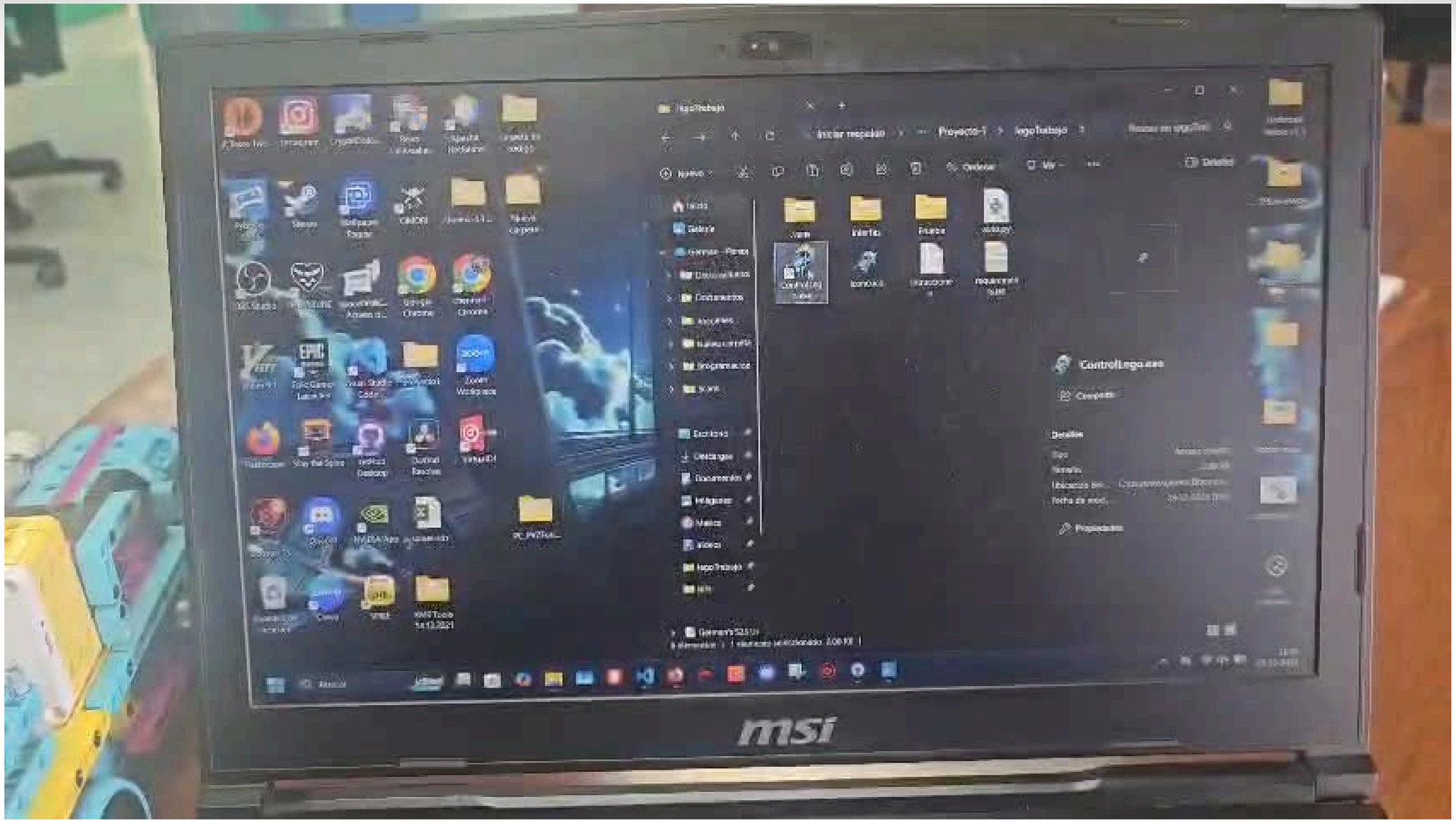


10. DEMOSTRACIÓN EN VIDEO

POV del robot:

10. DEMOSTRACIÓN EN VIDEO

Registro de la interfaz gráfica:



11. MANUAL DE USUARIO

Repositorio GitHub

URL del proyecto:
https://github.com/ChermanWest/Proyecto-1

Aspectos relevantes del manual

1. Requisitos del sistema:

- Kit LEGO SPIKE Prime con motores conectados.
- Python 3.x instalado.
- Conexión Bluetooth o USB entre Hub y computador.

11. MANUAL DE USUARIO

2. Instalación y ejecución:

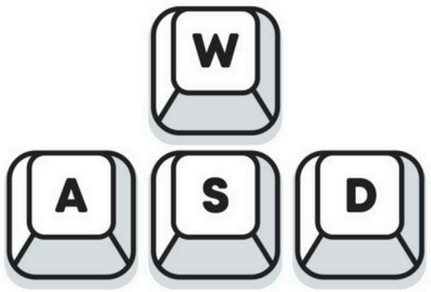
2.1 Clonar repositorio
git clone https://github.com/ChermanWest/Proyecto-1.git

2.2 Ejecutar
ControlLego.exe
...Proyecto-1/legoTrabajo/controlLego.exe

11. MANUAL DE USUARIO

3. Controles de la interfaz:

- W: Avanzar
- S: Retroceder
- A: Girar izquierda
- D: Girar derecha
- Slider: Ajusta velocidad/potencia
- Botón emergencia:
  Detiene inmediatamente el robot



4. Solución de problemas:

- Verificar que el HUB SPIKE Prime esté encendido
- Confirmar conexión correcta de motores
- Revisar conexión Bluetooth/USB estable

12. CONCLUSIONES

Competencias desarrolladas

Habilidades técnicas:

- Programación Python y Micro-Python para dispositivos físicos.
- Comunicación Bluetooth Low Energy (BLE).
- Desarrollo de interfaces gráficas con Custom Tkinter.
- Control de versiones con Git y GitHub.
- Manipulacion de sets de Lego Spike Prime (HUB, motores, bloques).

Habilidades de gestión:

- Trabajo colaborativo con rotación de roles.
- Gestión de proyectos mediante Redmine y Carta Gantt.
- Resolución de problemas bajo restricciones de tiempo.
- Documentación técnica profesional.
- Trabajo en equipo tanto virtual como presencialmente.

12. CONCLUSIONES

Desafíos superados

Principales obstáculos resueltos:

- Coordinación de horarios limitados
  - Solución: Comunicación digital.
- Complejidad en funciones avanzadas
  - Solución: Descomposición modular del código.
- Falta de algunos obstaculos para la pista
  - Solución: Improvisar obstáculos que simulen nuestros requisitos de pruebas
- Desorganización de roles.
  - Solución: Emitir tareas específicas, para cada miembro del equipo.

Mejoras futuras identificadas:

- Ajuste fino de parámetros de giro para curvas cerradas.
- Pruebas con carga simulada de material mineral
- Implementar un cambio de velocidad más dinámico
- Implementación de un contrapeso para cargas mas pesadas

12. CONCLUSIONES

Reflexión final

Este proyecto demostró que el control remoto manual es una solución viable para reducir riesgos en operaciones mineras, manteniendo al operador alejado de zonas peligrosas mientras mantiene control total del vehículo.

Más allá de los resultados técnicos, el mayor aprendizaje fue comprender que el desarrollo de soluciones de ingeniería requiere tanto dominio técnico como habilidades de gestión, comunicación y trabajo en equipo.

El prototipo representa un primer paso significativo hacia la implementación de sistemas de control remoto en la industria minera, validando que la Minería 4.0 puede contribuir efectivamente a la seguridad de los trabajadores.

MUCHAS GRACIAS