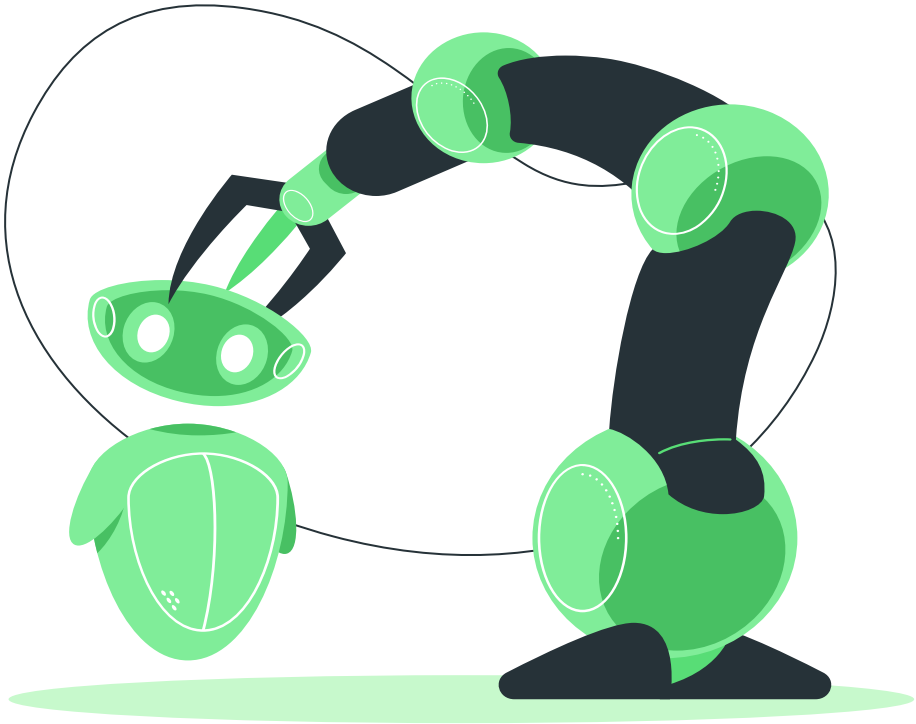
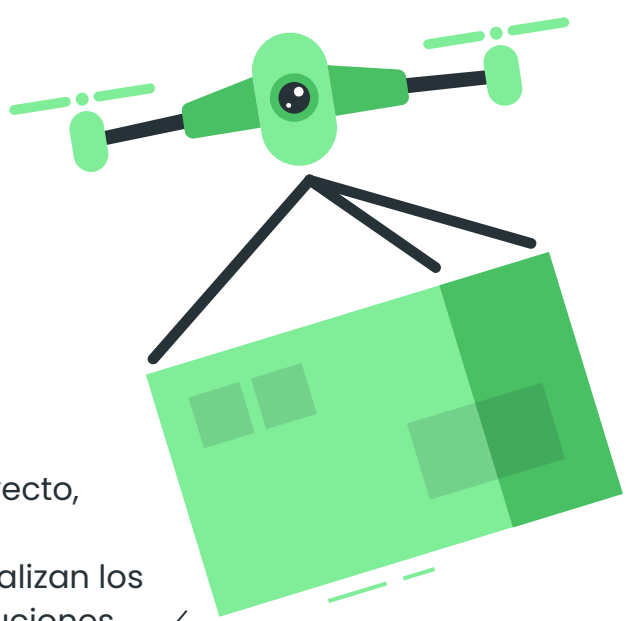
Proyecto McQueen



Integrantes: Milton Porlles
Renato chacon
Bruno rojas

Tabla de contenidos

Introducción



En esta presentación se presenta el avance del proyecto, mostrando las principales decisiones de diseño y el funcionamiento general del sistema. Además, se analizan los desafíos encontrados durante el desarrollo y las soluciones aplicadas para lograr un sistema funcional y eficiente.

Requerimientos funcionales

Actores del sistema

Usuario: Mineros encargados del transporte de materiales

Cliente: dueño de la empresa minera

RF1. Entrega del paquete

- EL robot debe llegar a destino sin perder la carga
- La carga debe llegar sin daños

RF2. Cualidades de carga

- El robot debe tener una zona de carga con capacidad de 96 cm3 como mínimo
- El robot debe poder moverse. con 50 gramos de carga de 24 gramos.

RF3. Movilidad

- El robot debe poder rodear obstáculos en su camino
- El robot debe poder pasar sobre rampas con una inclinación de 4° aprox.
- El robot debe poder desplazarse de un punto X a un punto Y

Alcance incluido

- El robot va a recorrer una pista definida, no una pista libre
- Tendrá obstáculos para dificultar su paso
- Solo hara una navegación de un punto x a un punto y y vuelta a su base
- Incluye un controlador simple

Fuera del alcance

- El robot no va a operar en un terreno real minero solamente en una pista de pruebas.
- No se van a cargar múltiples paquetes, solo un paquete.
- No va a hacer su recorrido en ambientes con polvo o condiciones más extremas, solo tendrá su obstáculo.

Requerimientos no funcionales

Atributo	Requerimiento	Métrica
Robustez	El robot debe soportar el entorno simulado (la pista) manteniendo su estructura	el robot debe operar sin daños estructurales de sobrecarga durante al menos 5 minutos
Rendimiento	la máquina debe ser eficiente en el transporte de materiales	La velocidad del dispositivo debe bastar para atravesar el recorrido en menos de 2 minutos
Usabilidad	La interfaz gráfica debe ser autoexplicativa	Un usuario no entrenado debe lograr realizar un movimiento con el robot en menos de 10 segundos
Disponibilidad	La conexión debe ser estable mientras se use el robot	La conexión bluetooth debe mantenerse por el 98% del tiempo de uso

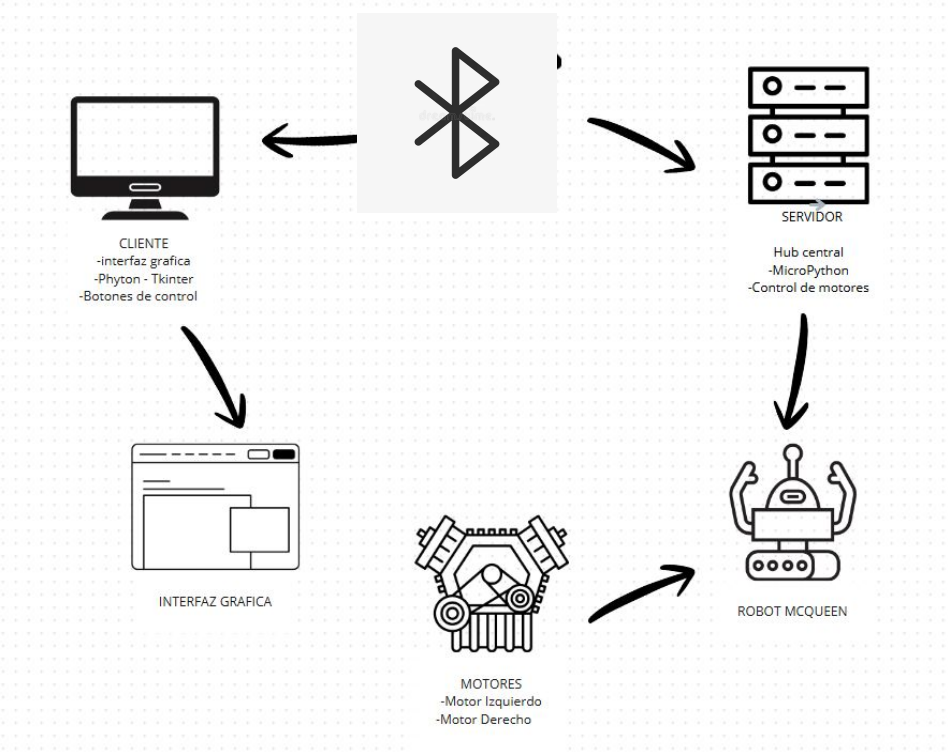
Arquitectura del software

Cliente (Interfaz de Control): Gestiona la GUI, captura las órdenes del operador y emite las peticiones de movimiento.

Canal (Bluetooth/BLE): Establece el enlace inalámbrico para el intercambio de datos en tiempo real entre ambos puntos.

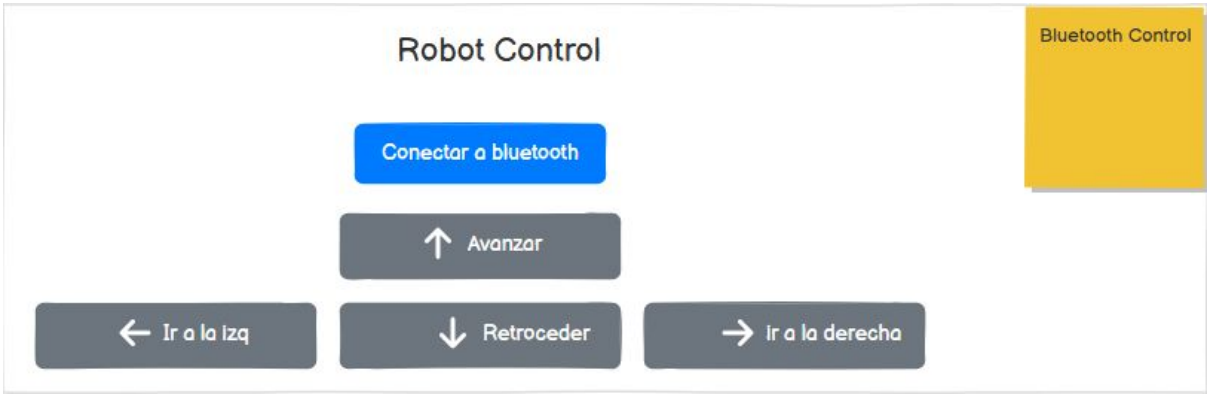
Servidor (Hub LEGO Spike): Procesa las instrucciones recibidas, ejecuta la lógica de control y coordina los actuadores.

Robot McQueen: Entidad física que integra motores para el desplazamiento y sensores para la detección de obstáculos.



Diseño inicial de la interfaz (GUI)

Nos permite conectar el robot vía bluetooth y controlarlo mediante botones direccionales de forma simple



Fundamento del movimiento

Para garantizar que el diseño es viable, debemos confirmar que el motor puede generar una **Fuerza Neta Positiva** que acelere el robot de 0 a su velocidad de crucero en un tiempo razonable. Para esto utilizaremos los datos de Lego.

Peso del vehículo = 0.8kg.

Diametro de Rueda = 0.028m.

Torque máximo de un motor = 0.25Nm.

Torque máximo de McQueen = 0.50Nm (dos motores).

Con la siguiente fórmula calculamos la fuerza de propulsión:

$$F_{\text{prop}} = \frac{\tau_{\text{total}}}{r_{\text{rueda}}} = \frac{0.50 \text{ N} \cdot \text{m}}{0.028 \text{ m}} \approx \mathbf{17.86} \text{ Newtons}$$

- **Aceleración Máxima** (a): Aplicando la Segunda Ley de Newton ($\sum F = m \cdot a$):

$$a = \frac{\sum F}{m} = \frac{17.06 \text{ N}}{0.80 \text{ kg}} = \mathbf{21.325} \text{ m/s}^2$$

Descripción del sistema

Se utilizó un repositorio en github donde se almacena toda la información de los cambios

Cliente:

Podemos ver la función encargada de la coneccion BLE

```python
async def _connect_async(self):
    """Lógica de conexión asíncrona (Bluetooth BLE)."""
    try:
        print(f"Buscando dispositivo: {self.DEVICE_NAME}...")

        dispositivo_ble = await find_device(self.DEVICE_NAME)

        print("Dispositivo encontrado. Conectando...")
        self.hub = PybricksHub(dispositivo_ble)
        await self.hub.connect()

        self.connected = True
        print("Conexión BLE exitosa.")
        return True

    except Exception as e:
        self.connected = False
        self.hub = None
        print(f"Error de conexión BLE: {e}")
        return False
```

Aqui la funcion encargada del lado del cliente que se encarga de enviar comandos al servidor

```python
async def _send_command_async(self, command: str):
    # (Esta función se mantiene igual a la versión USB/Serial)
    if not self.connected:
        return "Error: Desconectado"

    try:
        command_executable = f"ejecutar_comando('{command}')\n"
        await self.hub.write_output(command_executable.encode('utf-8'))
        return "Comando enviado"

    except Exception as e:
        self.connected = False
        self.hub = None
        return f"Error de envío: {e}"
```

Descripción del sistema

Servidor:

La función principal del lado del servidor que recibe comando y ejecuta

```python
def ejecutar_comando(comando: str):
    """Analiza y ejecuta el comando recibido por el Cliente."""
    hub.light.on(Color.BLUE)

    try:
        accion, valor_str = comando.split(':')
        valor = float(valor_str)

        if accion == "FWD":
            motor_izq.run_angle(VELOCIDAD_MOVIMIENTO, valor, wait=False)
            motor_der.run_angle(VELOCIDAD_MOVIMIENTO, valor, wait=True)
            hub.light.on(Color.GREEN)

        elif accion == "TURN":
            motor_izq.run_angle(VELOCIDAD_GIRO, valor, wait=False)
            motor_der.run_angle(VELOCIDAD_GIRO, -valor, wait=True)
            hub.light.on(Color.CYAN)

        elif accion == "STOP":
            motor_izq.stop()
            motor_der.stop()
            hub.light.on(Color.WHITE)

        else:
            hub.light.on(Color.RED)

    except Exception as e:
        print(f"Error de comando: {e}")
        hub.light.on(Color.RED)
```

Descripción del sistema

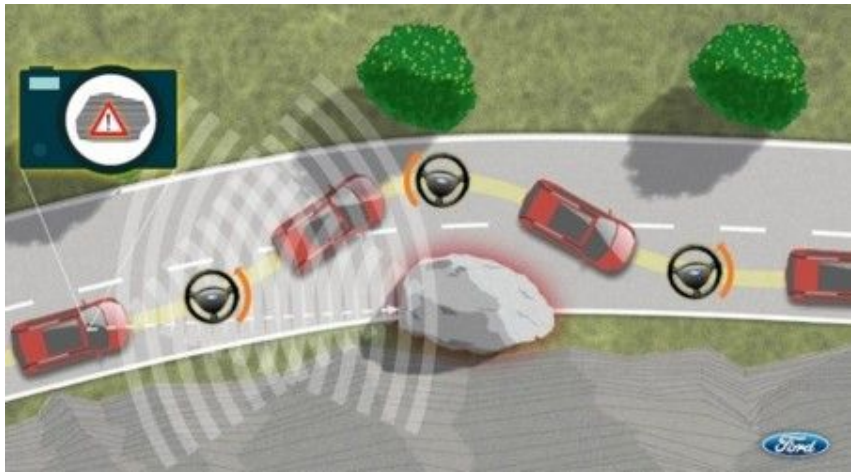
GUI:
Función en la que GUI maneja para conectarse al servidor

```
def handle_connect_button():
    """Llama al método de conexión del Cliente y actualiza la GUI."""
    status_label.config(text=f"Conectando a {PUERTO_HUB}...", fg="orange")
    root.update()

    # Llamada al método sincrónico del Cliente
    success = client.connect()

    if success:
        status_label.config(text=f"¡Conectado por USB a {PUERTO_HUB}!", fg="green")
    else:
        status_label.config(text="Desconectado", fg="red")
        messagebox.showerror("Conexión Fallida", "No se pudo conectar. Verifique el puerto y el robot.")
```

Interfaz utilizada para el movimiento del robot

Estado actual del proyecto

El proyecto se encuentra en un alto nivel de avance, con cerca del 90% ya que se aprendió sobre lego spike y sus componentes, se construyó un vehículo que tiene el potencial para transportar materiales, se ha codificado una secuencia que permite el desplazamiento del vehículo y la codificación de una interfaz gráfica está en su fase final. Por lo que se han realizado 4 de los 5 objetivos del proyecto



Problemas encontrados y solucionados

Durante el desarrollo del proyecto se presentaron dificultades por la falta de integrantes en el grupo, lo que aumentó la carga de trabajo. Para solucionar esto, se optó por doblar esfuerzos y coordinar el trabajo mediante la plataforma Discord.

Además, surgieron problemas en la programación del robot, los cuales aún se están resolviendo. Como solución, se decidió combinar la programación de LEGO Spike con Python, alternativa que ha mostrado resultados positivos y se espera completar en el corto plazo.

Conclusión

El proyecto McQueen representó una valiosa experiencia de aprendizaje, donde el equipo logró superar dificultades mediante el trabajo colaborativo. Durante esta etapa se realizaron mejoras en la presentación, como la reorganización de los requerimientos funcionales y no funcionales, la corrección de la arquitectura del sistema al especificar el uso de Bluetooth y una mejor explicación de la implementación de la GUI. Finalmente, el proyecto se encuentra casi finalizado y con proyección a futuras mejoras en rendimiento y diseño.


¡muuuchas
gracias!