**Diseño y simulación de un robot autónomo con IA en Webots**

**DATOS DEL EQUIPO**

* Nombre del proyecto: Pioner3-AT column mover
* Integrantes del equipo:

1. Samuel Arteaga López

2. Lucas Tarazona Comendador

1. **OBJETIVO DEL PROYECTO**

Describid brevemente cuál es la misión de vuestro robot. ¿Qué tarea tiene que cumplir? ¿Por qué es importante o interesante?

Nuestro robot tiene la misión de clasificar las columnas del mapa según si es par o impar. Es interesante que lo haga porque se decidirá si es par o impar en base al número resultante entre la suma de la fila y la columna, es decir, si tenemos una columna en la fila 1 y la columna 1, el resultado es 2, la clasificará como par, en caso contrario, impar

1. **DISEÑO DEL SISTEMA ROBOTIZADO**

* Tipo de robot elegido: Móvil diferencial (Pioneer3-AT)
* Razones de la elección: Robot introductorio en visión artificial, robusto y sencillo de controlar, ideal para pruebas en entornos simulados.
* Descripción del entorno simulado: Entorno matricial sencillo creado con columnas en posiciones fijas, basado en el mapa por defecto de Webots.
* Sensores y actuadores utilizados:
* Cámara RGB
* Sensores de distancia (para evitar colisiones)
* Motores diferenciales (ruedas)

1. **CONTROL Y PROGRAMACIÓN**

* Lenguaje de programación: Python
* Tipo de control usado: Máquina de estados sencilla basada en detección visual.
* Descripción del algoritmo principal:  
  El robot recorre el entorno detectando columnas con la cámara. Al detectar una columna, calcula si la suma de su posición fila + columna es par o impar, y ejecuta la acción correspondiente (por ejemplo, moverse a izquierda si par, derecha si impar).

*Mientras no termine el recorrido:*

*avanzar*

*si detecta columna:*

*capturar imagen*

*determinar posición (fila, columna)*

*calcular suma fila + columna*

*si suma es par:*

*realizar acción "par" (ej: mover columna a izquierda)*

*sino:*

*realizar acción "impar" (ej: mover columna a derecha)*

*evitar obstáculos cercanos*

1. **VISIÓN ARTIFICIAL E IA**

* Tipo de detección usado: Detección por color (para reconocer las columnas)
* Herramienta utilizada: OpenCV
* Comportamiento del robot ante lo detectado:
* Detecta objetos de color específico (columnas).
* Extrae la posición aproximada en el entorno.
* Clasifica como par o impar y realiza movimiento de clasificación.

1. **PRUEBAS Y RESULTADOS**

* Pruebas realizadas:
* Obstáculos estáticos (otras columnas, paredes)
* Objetos móviles (simulación de pequeñas variaciones)
* Variaciones de iluminación (pequeñas para ver sensibilidad del sensor de visión)
* Resultados:
* ¿Qué funcionó bien?  
  La detección de color fue robusta en iluminación normal; el cálculo de posición y clasificación funcionó en la mayoría de los casos.
* ¿Qué problemas tuvisteis?
  + Detección errónea bajo luz muy baja o demasiado alta.
  + A veces interpretaba el suelo como objeto.
* ¿Cómo los solucionasteis?
  + Aplicando filtros de color más precisos en OpenCV.
  + Añadiendo una altura mínima en la detección (descartar objetos demasiado bajos).

1. **EVALUACIÓN DEL PROYECTO**

* Tiempo estimado de desarrollo: Aproximadamente 2 semanas.
* ¿Se cumplió el objetivo? Sí, el robot fue capaz de detectar columnas y clasificarlas correctamente en la mayoría de las pruebas.
* ¿Qué cambiaríais para mejorar el robot?
* Usar detección basada en forma además de color.
* Implementar un algoritmo de localización más preciso para determinar la posición exacta.
* ¿Qué habéis aprendido del proyecto?
* Integrar visión artificial en un entorno simulado.
* Programar máquinas de estado básicas en Python.
* Planificar de manera estructurada un proyecto robótico completo.

1. **EVIDENCIAS VISUALES**

* Capturas de pantalla del simulador:  
  (Añadir capturas del robot moviéndose y detectando columnas)
* Esquemas del entorno o diseño del robot:  
  (Añadir un esquema sencillo del mapa, mostrando columnas y ruta del robot)
* Vídeo o enlace:  
  [Vídeo del proyecto](https://drive.google.com/....) (enlace a vuestra grabación)

1. **ARCHIVOS ADJUNTOS**

☐ Código fuente  
☐ Archivo del mundo Webots (.wbt)  
☐ Controlador del robot (.py / .cpp)  
☐ Presentación final (opcional)