

## World Robot Olympiad 2025 – Future Engineers

### Team IITA InnovaX

#### Síntesis

Teniendo en cuenta que la verdadera ingeniería se refiere no solamente a la resolución de problemas, sino a la documentación y comunicación de ideas, el presente diario tiene como reflejo el desarrollo progresivo que realizamos en este proyecto para resolver el desafío planteado en la categoría Futuros Ingenieros de la WRO.

#### Sobre el equipo

Desde el presente año, el equipo se encuentra formado por Gerardo B. Uriburu Romero y Natanahel Fernández. Hemos comenzado a trabajar desde el mes de Julio y coordinado nuestro trabajo en el "Instituto de Innovación y Tecnología Aplicada (IITA)"

Siendo un equipo conformado solamente por dos personas, la distribución del trabajo se ha intentado que sea lo más equitativa posible, siendo que ambos integrantes no solamente tenemos roles definidos, sino que también colaboramos mutuamente en las actividades que desarrollamos individualmente.

- Gerardo B. Uriburu Romero es el líder del equipo, encargado de la programación y estrategia del robot durante su desarrollo.
- Natanahel Fernández (sublíder del equipo) ha sido el encargado de la programación y configuración de la visión computarizada que se encuentra en el robot.

Con el fin de desarrollar el proyecto de manera coordinada, dedicamos al mismo reuniones cuya duración (durante las primeras semanas) ha sido de dos horas reloj, extendiéndose a medida que se acercaba la competencia Regional y Nacional a un máximo de 8 horas reloj.

Desde un principio éramos conscientes sobre el poco tiempo que teníamos para el desarrollo del robot, sin embargo, nos pareció interesante y entretenido el afrontar este desafío.

El contenido volcado en el presente documento no solamente incluye fechas cronológicas con el trabajo realizado, sino también incluye de manera descriptiva los aspectos que se tomaron en cuenta tanto para la construcción como el desarrollo del software que se utiliza en el autómata.

## PLANTEO DEL DESAFÍO

El desafío *Futuros Ingenieros* de la Olimpiada Mundial de Robótica (WRO 2025) plantea el diseño, construcción y programación de un vehículo autónomo capaz de recorrer un circuito predefinido, enfrentando la presencia de obstáculos y condiciones variables que simulan los retos propios de la ingeniería real en el desarrollo de automóviles inteligentes.

Nuestro objetivo como equipo es desarrollar un robot que combine **precisión mecánica, confiabilidad electrónica y un software de control robusto**, de manera tal que logre un desempeño estable en la pista y demuestre un enfoque sólido en la resolución de problemas de ingeniería.



El proceso de desarrollo adoptado se fundamenta en el **ciclo iterativo de diseño ingenieril**: idear, construir, probar, analizar resultados, retroalimentar y mejorar. Este enfoque garantiza la evolución progresiva del vehículo a través de distintas versiones, permitiendo optimizar tanto el hardware como los algoritmos de navegación.

Más allá del resultado en la competencia, este proyecto se concibe como una **experiencia formativa integral**, que potencia el trabajo en equipo, la creatividad y la aplicación práctica de conceptos de robótica, control y visión artificial, alineándose con los principios fundamentales de la ingeniería moderna.

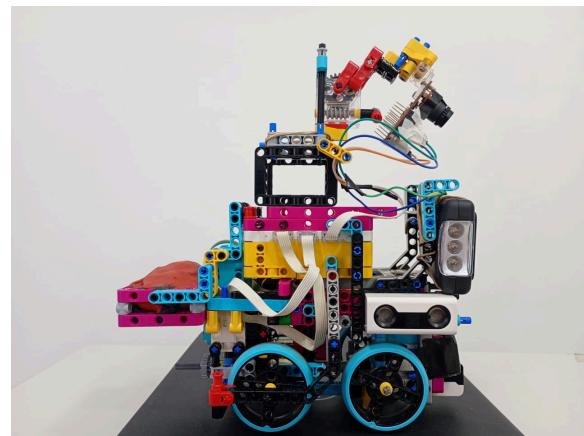
## GESTIÓN DEL MOVIMIENTO

El diseño del vehículo autónomo constituye la base del proyecto y se desarrolló siguiendo un enfoque sistemático propio de la ingeniería. Para ello, se analizaron los requerimientos de la competencia y se descompuso el sistema en tres grandes subsistemas: mecánico, electrónico y de software. Cada uno de ellos fue diseñado y optimizado de manera independiente, pero siempre considerando la necesidad de integración y el desempeño global del robot.

## Chasis y Estructura Mecánica

El chasis constituye el esqueleto estructural del vehículo y cumple una doble función: proporcionar rigidez y estabilidad al sistema, y servir como soporte modular para el montaje de los distintos componentes.

El diseño se llevó a cabo utilizando piezas LEGO Spike, seleccionadas por su modularidad, facilidad de ensamblaje y compatibilidad con procesos de prototipado rápido. La estructura fue concebida con un **centro de gravedad bajo**, lo que reduce la probabilidad de vuelco durante maniobras bruscas y mejora la tracción de las ruedas traseras. Asimismo, se cuidó la **distribución de masas**, centralizando el peso en torno al bloque de baterías para optimizar la estabilidad dinámica.



Vista lateral del robot

El montaje de los motores y sensores se planificó desde el inicio, destinando espacios específicos que garantizan su correcta fijación y la posibilidad de ajustes en futuras iteraciones. En particular, la cámara de visión artificial fue instalada en un soporte frontal con inclinación ajustable, lo que permite modificar su campo de visión de acuerdo con los requerimientos de la pista y las condiciones de iluminación.

## Sistema de Tracción y Dirección

El sistema de movimiento del vehículo está conformado por dos subsistemas claramente diferenciados:

- **Propulsión:** a cargo de un motor de tracción ubicado en la parte trasera, el cual transmite la potencia a las ruedas mediante un sistema diferencial mecánico. Esta disposición replica la configuración de tracción trasera utilizada en automóviles reales, ofreciendo un comportamiento más realista y estable. El

motor fue calibrado para proporcionar aceleraciones progresivas, reduciendo el riesgo de pérdida de adherencia en superficies lisas.

- **Dirección:** implementada mediante un motor adicional instalado en la parte frontal, que controla el ángulo de las ruedas delanteras. Se adoptó un mecanismo de dirección tipo *car steering*, que permite realizar giros de forma progresiva y precisa. Esta elección responde a la necesidad de aproximar el comportamiento del robot al de un vehículo convencional, lo que a su vez contribuye a desarrollar estrategias de control más aplicables a la ingeniería automotriz real.

El diseño de la tracción del vehículo ha requerido especial atención, no solamente para cumplimentar con las reglas generales de la categoría, sino por el hecho de que se buscó mitigar el deslizamiento de las ruedas (patinaje) tanto durante un movimiento recto como durante curvas y maniobras.

Al mismo tiempo, la dirección ha sido desarrollada de tal manera que el robot pueda garantizar que la respuesta de las señales de control fuera inmediata y estable.

## GESTIÓN DE LA POTENCIA Y SENTIDOS

Para gestionar el suministro de energía, hemos decidido utilizar el Lego Spike Hub, que nos permite no solamente otorgar la suficiente potencia a los sensores que implementamos en la propia arquitectura del robot, sino que también nos otorga la posibilidad de programar dichos componentes de manera nativa con el Firmware de PyBricks compatible con Lego Spike Prime.

La percepción del entorno constituye un aspecto crítico en el desempeño de vehículos autónomos. Para ello, el diseño del vehículo contempla la integración de diversos sensores que trabajan de manera complementaria, asegurando una correcta interpretación de la pista y una respuesta confiable en tiempo real.

- **Sensores ultrasónicos:** dos ubicados en la parte delantera del vehículo de forma lateral, permiten la detección de obstáculos a través de la emisión y

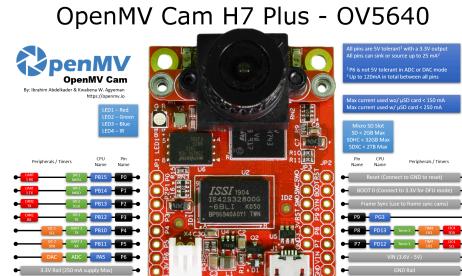
recepción de ondas acústicas. Su disposición perpendicular respecto al eje central del robot posibilita estimar distancias con un margen de seguridad adecuado, lo que resulta fundamental para la ejecución de maniobras de evasión y el mantenimiento de trayectorias seguras.

- **Cámara OpenMV H7 Plus:** constituye el núcleo del sistema de visión artificial, siendo programada para identificar colores específicos en el entorno (naranja y violeta) que actúan como indicadores de maniobras de giro. Su montaje ajustable en altura y ángulo garantiza un campo de visión adaptable a diferentes escenarios y condiciones de iluminación, aumentando la robustez del sistema frente a variaciones externas.
- **Sensor de color (LEGO Spike):** incorporado en la parte inferior del chasis, su función principal es el reconocimiento del color del suelo. Este sensor complementa la visión artificial al proporcionar información crítica en segmentos de pista donde es necesario detectar líneas o áreas específicas, asegurando que el vehículo identifique correctamente marcadores ubicados en la superficie. Su uso también permite verificar cambios de contraste en el terreno, brindando una segunda capa de redundancia para la toma de decisiones.

Para la conexión de la OpenMV Cam H7 Plus y el Lego Spike Hub, el principal problema se manifestó a la hora de realizar tal interconexión dado a la incompatibilidad de tales plataformas.

Para solucionar esto anterior, fué necesario modificar un cable conector del kit Lego Spike Prime y realizar las conexiones individuales para cada cable de ambas plataformas.

Una vez realizado lo planteado anteriormente, se han implementado bibliotecas como PUP Remote y lpf2 para realizar la comunicación entre estos componentes cruciales, desembocando en el reconocimiento de objetos con la cámara e interpretación de los mismos a través del hub.

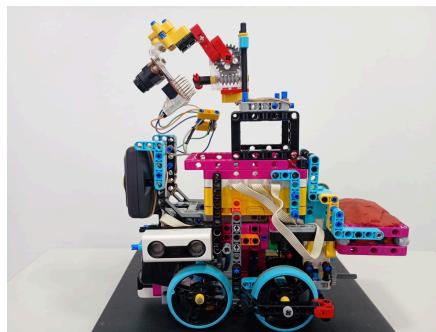


Además de esto, se planteó en su momento implementar un tercer sensor de distancia para mayor precisión, sin embargo, esto no se pudo concretar dada la escasez de puertos disponibles como se puede observar en el diagrama eléctrico del vehículo.

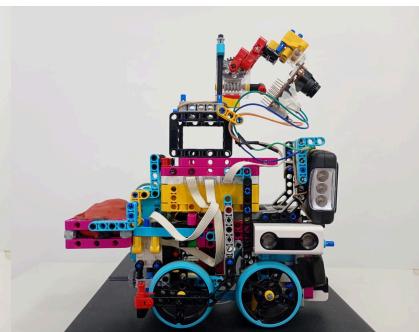
La combinación de estos dispositivos otorga redundancia y confiabilidad al subsistema de percepción. Mientras los sensores ultrasónicos aseguran la detección de obstáculos físicos, la cámara OpenMV interpreta estímulos visuales a la distancia y el sensor de color aporta precisión en la detección de marcadores directamente en el suelo. Esta sinergia garantiza una mayor robustez en la navegación autónoma del vehículo.

## FOTOS DEL ROBOT

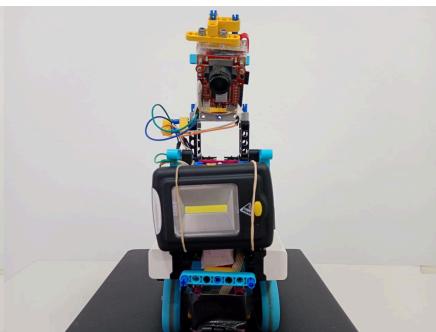
A continuación se presentan las fotografías del robot desde distintos lados permitiendo visualizar su construcción, sensores que permiten su movilidad por el entorno y distribución de peso.



Vista lateral izquierda



Vista lateral derecha



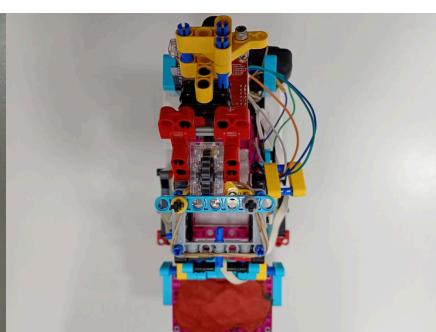
Vista frontal



Vista trasera



Vista inferior



Vista superior

## FOTO DEL EQUIPO



## CRONOLOGÍA DE ACTIVIDADES

A continuación, se presenta semana a semana las actividades realizadas, problemas encontrados, discusiones y tareas a realizar. Al mismo tiempo, se puede encontrar una copia del presente documento (al igual que material necesario para complementar la documentación) en nuestro repositorio de GitHub: <https://github.com/IITA-Proyectos/WRO2025-IITA-SALTA-FE>

La estructura utilizada para representar de la mejor manera posible la cronología de las actividades será la siguiente:

- Fecha del encuentro
- Objetivos del día
- Tareas realizadas
- Problemas y soluciones
- Planificaciones tentativas

## **FECHA DEL ENCUENTRO: 07-07**

### **Objetivos del día:**

- Iniciar el trabajo práctico con el robot.
- Ensamblar un primer prototipo del sistema de tracción con diferencial utilizando piezas LEGO.
- Familiarizarnos con la estructura y funcionamiento general del vehículo.

### **Tareas realizadas:**

- Construcción del mecanismo de tracción trasera con diferencial mecánico.
- Montaje básico del chasis con piezas LEGO Spike.
- Exploración y análisis de la distribución de piezas y posibles configuraciones estructurales.
- Trabajo inicial de integración equipo–robot, con el fin de comprender mejor las posibilidades y limitaciones del sistema.

### **Problemas y soluciones:**

- No se presentaron problemas técnicos relevantes, aunque el armado requirió varias horas de pruebas para encontrar la configuración más estable del diferencial.
- La mayor dificultad fue la curva de aprendizaje inicial: reconocer cómo adaptar las piezas LEGO a un diseño funcional de vehículo autónomo.

### **Planificación / Tentativa:**

- Avanzar en la definición de la dirección del vehículo (car steering).
- Revisar la estabilidad del chasis y preparar la base para incorporar sensores en los siguientes encuentros.

## FECHA DEL ENCUENTRO: 14-07

### Objetivos del día:

- Definir la estrategia de navegación para el **Desafío 1** antes de comenzar a programar.
- Analizar las posibles variantes del recorrido y sus complicaciones.
- Dibujar la pista y estudiar los escenarios que podrían presentarse (sentido de giro, mantenimiento de la ruta, regreso al punto de partida, etc.).

### Tareas realizadas:

- En papel, esbozamos la pista del desafío 1 con sus secciones curvas y rectas.
- Enumeramos escenarios posibles:
  - Giro en sentido horario o antihorario
  - Mantenerse centrado en el carril durante curvas y rectas
  - Volver al punto de partida o sección inicial
    - Riesgos de salirse de pista en esquinas o perder la referencia visual
- Discutimos y evaluamos las complicaciones en cada escenario:
  - Qué sucede si el robot se desorienta en curvas profundas
  - Cómo reaccionar ante bordes o cambios de pendiente
  - Cómo asegurarse de que el robot "salga" de la sección de inicio correctamente
- Decidimos criterios para la estrategia base del algoritmo: flexibilidad al cambio de orientación, tolerancias para corregir desvíos, manejo seguro de curvas.

### Problemas y soluciones:

- Problema: demasiadas variables posibles — muchos escenarios a tener en cuenta.

Solución: priorizamos los más probable(s) o “peor caso” para codificar primero, dejar algunos casos secundarios para iterar luego.

- Problema: incertidumbre sobre cuáles condicionantes impuestos por las reglas podrían aparecer (sentido de giro aleatorio, obstáculos, posición de señales).

Solución: consultar las reglas generales de *Futuros Ingenieros* para ver esas variantes. (Por ejemplo: el circuito puede cambiar, sentido de conducción puede variar)

### **Planificación / Tentativa:**

- En el próximo encuentro, comenzar a modelar el algoritmo (flujo de control) para navegar la pista en los escenarios más críticos.
- Simular mentalmente o con esquemas el paso por curvas, correcciones y retorno.
- Decidir la estructura básica del código (estado actual del recorrido, manejo de errores, corrección de rumbo).
- Releer las reglas oficiales del desafío 1 para asegurarnos de que no estamos pasando por alto condiciones especiales (como sentido aleatorio de marcha)

### **FECHA DEL ENCUENTRO: 21-07**

### **Objetivos del día:**

- Elaborar el primer pseudocódigo para modelar la lógica del recorrido en el Desafío 1.
- Analizar estructuras y enfoques utilizados por equipos ganadores de la WRO Future Engineers del año pasado.

- Identificar posibles problemas al momento de implementar el control de curvas y detección de líneas.

### Tareas realizadas:

- Redactamos un **pseudocódigo base** para el manejo de curvas, considerando los sensores disponibles (color y ultrasónicos) y la necesidad de alternar entre estados (recta, inicio de curva, en curva, fin de curva).
- Estudiamos videos/documentación de equipos finalistas y campeones de la edición 2024, tomando nota de:
  - Uso de estructuras livianas y chasis compactos.
  - Posicionamiento de sensores para mejorar la precisión en curvas.
  - Métodos de corrección en tiempo real en curvas largas.
- Ajustamos el pseudocódigo para incluir decisiones condicionales claras (inicio de curva → dirección → mantenimiento → fin de curva).

### Problemas y soluciones:

- Problema: dificultad para estimar el ángulo ideal de dirección y cómo traducirlo a grados en el motor de dirección.  
 Solución: dejar el cálculo parametrizado en el pseudocódigo (ejemplo: 21.8° dirección → 65.4° motor B), pendiente de calibración experimental.
- Problema: alta complejidad en los casos de curvas dobles o consecutivas.  
 Solución: dividir el flujo en **estados de navegación** y no intentar resolver todo de forma lineal.
- Problema: diferencias en los diseños de los equipos del año anterior generaban dudas sobre qué camino tomar.  
 Solución: identificar patrones comunes (dirección tipo car steering, cámara ajustable, redundancia en sensores) para guiar nuestras

decisiones sin copiar diseños.

### **Planificación / Tentativa:**

- En el siguiente encuentro, comenzar a traducir el pseudocódigo a código real en Python/Pybricks, probando primero la detección de líneas (violeta/naranja).
- Diseñar pruebas controladas en pista para medir cuánto debe girar exactamente el motor de dirección.
- Continuar documentando los problemas encontrados para ir depurando el algoritmo en cada iteración.

### **FECHA DEL ENCUENTRO: 28-07**

### **Objetivos del día:**

- Continuar afinando la estrategia de recorrido.
- Avanzar en la transición del pseudocódigo hacia código real.

### **Tareas realizadas:**

- Discusión sobre cómo estructurar el programa en módulos (detección de color, manejo de curvas, avance recto).
- Ensayos preliminares de detección de líneas con el sensor de color LEGO.

### **Problemas y soluciones:**

- Problema: dudas sobre cómo integrar el pseudocódigo a Pybricks de manera clara.  
Solución: decidir iniciar con un bloque simple de detección de líneas antes de pasar a las curvas.

### **Planificación / Tentativa:**

- Probar pequeños fragmentos de código en pista en la próxima reunión.

### **FECHA DEL ENCUENTRO: 02-08**

#### **Objetivos del día:**

- Ensayar las primeras pruebas en pista con detección de líneas.

#### **Tareas realizadas:**

- Programación inicial en Pybricks para que el robot reconozca la línea violeta y naranja.
- Observación del comportamiento del sensor en distintas condiciones de luz.

#### **Problemas y soluciones:**

- Variación en la lectura del sensor según iluminación.  
Solución: plantear la idea de calibrar umbrales dinámicos o usar la cámara como apoyo.

#### **Planificación / Tentativa:**

- Implementar pequeños ajustes en los umbrales del sensor en el próximo encuentro.

## **FECHA DEL ENCUENTRO: 09-08**

### **Objetivos del día:**

- Mejorar la detección de colores y comenzar a implementar las primeras rutinas de curva.

### **Tareas realizadas:**

- Ajustes sobre los umbrales de detección del sensor de color.
- Ensayo del inicio de curva al detectar violeta.

### **Problemas y soluciones:**

- El robot no siempre reaccionaba al mismo punto de la pista.  
Solución: decidir que se necesitaría implementar un sistema más robusto (PID para curvas).

### **Planificación / Tentativa:**

- Introducir control PID básico para el próximo encuentro.

## **FECHA DEL ENCUENTRO: 16-08**

### **Objetivos del día:**

- Implementar y ajustar el control PID para el manejo de giros.

### **Tareas realizadas:**

- Pruebas de giros utilizando PID en la dirección. Trabajo sobre el cálculo de ángulos de giro (target angle).
- Ajustes de los signos en las ecuaciones para que el robot gire correctamente en ambos sentidos.

### **Problemas y soluciones:**

- Problema: dificultad para resetear el PID en cada secuencia, lo que generaba errores acumulados.  
Solución: identificar la necesidad de inicializar variables clave al inicio de cada curva.
- Problema: confusión sobre los signos positivos/negativos en los cálculos de ángulo.  
Solución: ensayo y error con distintos valores hasta encontrar coherencia en el comportamiento.

### **Planificación / Tentativa:**

- Definir valores iniciales de PID más estables.
- Documentar claramente cómo y cuándo se debe resetear el PID en cada secuencia.

- Probar nuevamente en pista en la próxima sesión con estas correcciones.

## **FECHA DEL ENCUENTRO: 06-09-2025**

### **Ajustes mecánicos + Calibración de umbrales (color rojo)**

#### **1. Ajustes mecánicos**

- Se agregó **peso en la parte trasera** del robot para compensar el exceso de peso de la linterna en el frente.
- Problema detectado: el peso trasero provocaba que se **atasque la parte de la tracción**.
- Solución: ajustes en el **eje de tracción**, revisión de **engranajes** y del **diferencial**, logrando mayor fluidez en la conducción.

#### **2. Cámara**

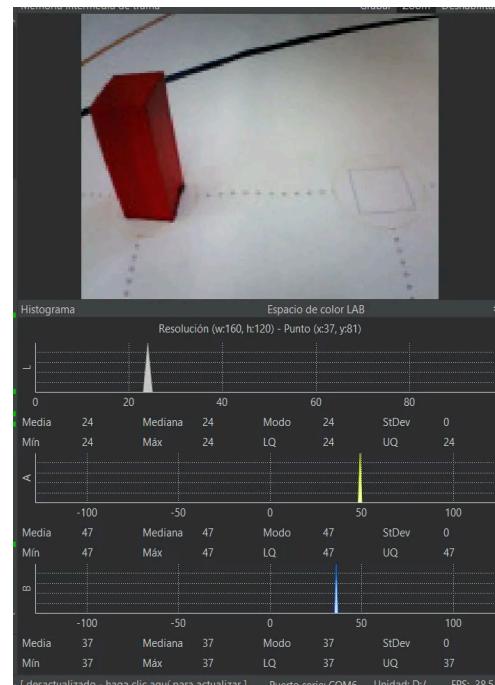
- Se ajustó la **nitidez** de la lente y se limpió para mejorar la detección.

#### **3. Calibración de umbrales (color rojo)**

Se trabajó sección por sección (A, B, C, D) creando **umbrales más amplios** para cubrir diferentes tonalidades de rojo en condiciones de luz variables.

Estrategia: ubicar el obstáculo en las **6 posiciones posibles dentro de cada sección** y validar detección.

Resultado: funcionó en todas las pruebas excepto en la **sección D**, donde con persianas abiertas hubo fallos; con persianas cerradas, la detección fue correcta.



## **FECHA DEL ENCUENTRO: 08-09-2025**

### **Detección de pilares rojos + Investigación sobre modelo LAB**

#### **1. Programa en OpenMV**

- Se desarrolló un programa que mejora la **detección de los pilares rojos**, no solo por color, sino también a partir de la **delimitación de su forma geométrica**.
- Esto permite una mayor robustez en escenarios con cambios de iluminación o posibles interferencias de otros colores.

#### **2. Investigación sobre valores LAB**

- **L** → Luminosidad del ambiente en la imagen.
- **A** → Canal rojo/verde (verde negativo, rojo positivo).
- **B** → Canal azul/amarillo.
- Se concluyó que este modelo facilita una calibración más precisa al separar la luz (L) de la información de color (A y B).

#### **3. Próximos pasos y mejoras pendientes**

- Incorporar la **comunicación con PUP Remote** para enviar datos al Spike.
- Ajustar los umbrales de detección para el **color verde**.
- Configurar la **exposición de la cámara** para mantener una luminosidad estándar y evitar variaciones excesivas en entornos cambiante.

## **FECHA DEL ENCUENTRO: 13-09-2025**

### **Objetivos del día:**

- Validar la **detección del color rojo** en OpenMV.
- Asegurar la **conexión confiable** entre OpenMV y Spike Prime (Pybricks).
- Lograr identificar correctamente el **centro geométrico del objetivo** para control preciso del robot.

### **Tareas realizadas:**

#### **1. Pruebas de conexión OpenMV ↔ Pybricks**

Se realizaron test de envío de datos entre OpenMV y Spike Prime.

Problema detectado:

```
PUPRemote Error: Nothing connected or no script running on
remote
```

→ Solución: asegurar que el USB esté desconectado y que el **Puerto E** esté correctamente conectado a la cámara OpenMV H7 Plus.

→ Verificar que los tres scripts necesarios estén en la cámara:

`main.py` , [`pupremote.py`](#) , `lpf2.py`

→ Las últimas dos líneas de envío de datos en OpenMV deben estar des-comentadas:

```
# Enviar datos al hub

pr.update_channel('cam',    id_blob_elegido,    x_blob_elegido,
y_blob_elegido, area_blob_elegido)
```

`pr.process()`

## 2. OpenMV – Detección de pilares

- Se definieron **umbrales CIELab** para distintos tonos de rojo:
- Rojo oscuro
- Rojo brillante
- Rojo muy oscuro
  - Se utiliza la función `find_blobs()` para identificar **todos los blobs** que coincidan con los umbrales.
  - Se selecciona **el blob más grande** y se envían los datos al Hub:

`id_blob` → identificador del color, `x_blob`, `y_blob` → coordenadas del centro, `area_blob` → número de píxeles (estimación de distancia)

-Debug visual: rectángulo delimitador, cruz en el centro y nombre del color.

## 3. Spike Prime – Recepción y reacción

- Recepción de datos vía `pr.call('cam')`.
- Lógica de control basada en:
  - **Área del blob (`area_blob`)** → distancia al pilar
    - Grande → frenar o reducir velocidad
    - Pequeño → avanzar rápido
  - **Posición X (`x_blob`)** → orientación del robot
    - Menor al centro → girar a la derecha
    - Mayor al centro → girar a la izquierda
    - Cercano al centro → mantener recto

## Resultados logrados:

- Detección correcta del **color rojo**.
- Identificación del **centro geométrico** y estimación de distancia basada en **area\_blob**.
- Control inicial de orientación y tracción funcionando correctamente.
- Pendiente: pruebas con **diferentes tonalidades de iluminación** para validar robustez.

## FECHA DEL ENCUENTRO: 14-09-2025

### Calibración de cámara, iluminación y geometría – Preparación para segundo desafío

#### Tareas realizadas:

- **Calibración de la cámara OpenMV**
  - Ajustamos el **valor de exposición** manualmente para controlar la luminosidad, evitando que la cámara haga que el escenario se vea demasiado oscuro o demasiado claro automáticamente.
  - Se realizaron pruebas con obstáculos en el escenario para validar la calibración en condiciones reales.
- **Pruebas de conexión OpenMV ↔ Spike Prime**
  - Verificamos que los datos de los pilares (centroide y color) llegaran correctamente al Spike.
  - Se tuvieron en cuenta las distancias y los colores de los obstáculos para asegurar que la comunicación fuera precisa.
- **Detección y geometría de los pilares**
  - La cámara detecta correctamente los pilares (obstáculos) y calcula su **centroide**.
  - Se dibuja un **rectángulo en OpenMV** para guiar visualmente la detección y validar su precisión.

Resultado: gran paso en la robustez del sistema de detección.

- **Análisis del segundo desafío**

- Se descompuso el problema en **módulos**:
  - Detectar color del pilar (rojo → derecha, verde → izquierda).
  - Volver al centro.
  - Continuar detectando y realizar curvas según corresponda.
- Se centraron esfuerzos en que el robot **avance recto con PID**, encontrando complicaciones al inicio.
- Solución encontrada: aplicación de una **fórmula matemática** basada en documentación y repositorios revisados.

### Ejemplo práctico:

Si quisieras calcular Kp matemáticamente para tu robot:

```
python

# Datos de tu robot (necesitas medirlos)
masa_robot = 1.5 # kg
distancia_ejes = 0.12 # metros
velocidad_crucero = 100 # unidades del motor

# Fórmula simplificada
kp_calculado = (velocidad_crucero / 100) * (masa_robot / distancia_ejes)
kd_calculado = kp_calculado * 0.25 # Regla empírica
```

Gracias a esta fórmula, el robot ahora **mantiene trayectoria recta de manera estable**.

### Resultados logrados:

- Detección confiable de pilares y cálculo correcto del centroide.
- Comunicación estable con Spike Prime.

- Robot avanza recto de manera consistente usando PID, base para las curvas posteriores del segundo desafío.

**ÚLTIMAS FECHAS DE ENCUENTRO: 20-09-2025 → 23-09-2025**

### **Optimización final, ajustes mecánicos y segundo desafío**

#### **Objetivos generales:**

- Consolidar la **detección de colores** usando una sola configuración de exposición.
- Mejorar la **distribución de peso y montaje de la cámara/linterna**.
- Refinar el **movimiento y control del robot** para cumplir con el segundo desafío (detección y esquive de obstáculos).

#### **Tareas realizadas:**

##### **1. Optimización de detección de color**

- Gracias al ajuste manual de **exposición de la cámara**, ya no se necesitaban múltiples umbrales.
- Esto permitió mantener una detección consistente de rojo y verde sin depender del entorno de luz.

##### **2. Ajustes mecánicos y distribución de peso**

- Movimos la **cámara ligeramente hacia atrás** para aumentar el rango de visión.
- La **linterna** se dejó en la parte delantera para asegurar una iluminación óptima de los obstáculos.

- Se solucionó un problema con el **eje de dirección**, evitando que hiciera fuerza sobre el mecanismo y distribuyendo la carga a una pieza adicional para mayor estabilidad.

### 3. Segundo desafío – Detección y esquive de obstáculos

- Implementación del "**giro simple**":
  - Si el pilar es rojo → giro pequeño hacia la derecha.
  - Solo trabajado inicialmente con los obstáculos del medio.
- Se probó el **giro brusco**, aunque aún no es totalmente confiable.
- Se aseguró que el **movimiento simple** evite colisiones y que el robot retroceda cuando sea necesario.

### 4. Mantenimiento del centro del carril

- Trabajo intenso para que el robot **mantenga siempre la trayectoria central**, corrigiendo desviaciones automáticas.
- Este fue uno de los aspectos más complejos y se pulió durante los cuatro días.

### Resultados logrados:

- Detección confiable de colores sin necesidad de múltiples umbrales.
- Mejor estabilidad mecánica y distribución de peso optimizada.
- Giro simple funcional para esquivar obstáculos.
- Robot mantiene centrado el carril de manera consistente, base para maniobras más complejas.

**Tiempo invertido:** jornadas de **8-9 horas diarias**, enfocadas en pruebas, ajustes y refinamiento de algoritmos.

### PERFORMANCE VIDEOS

- **Desafío Abierto:** [https://youtu.be/\\_RtZ766-rAc](https://youtu.be/_RtZ766-rAc)
- **Desafío de Obstáculos:** [https://youtu.be/Cv\\_SwLSWwAo](https://youtu.be/Cv_SwLSWwAo)