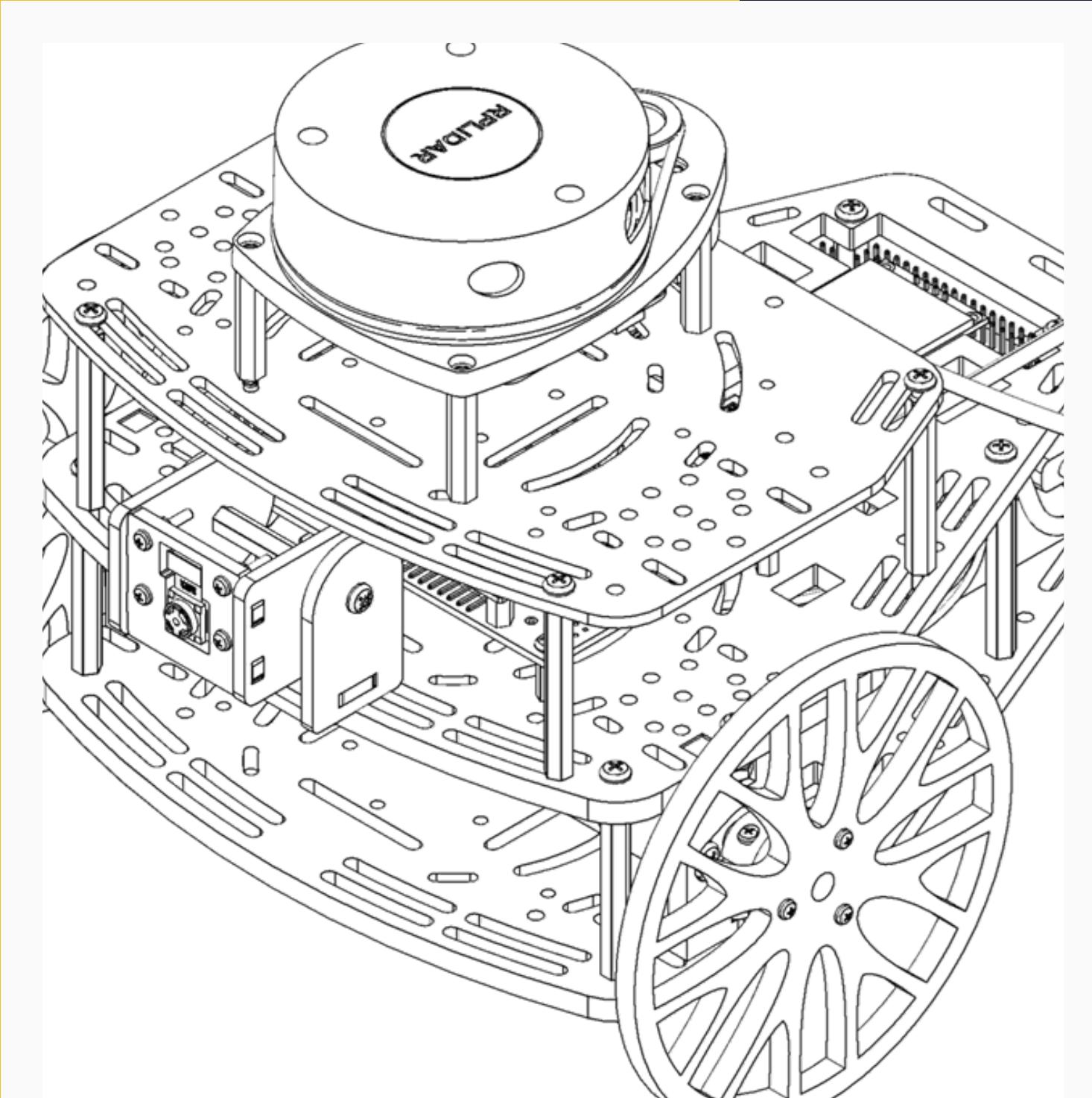




# ACTIVIDAD RETO VISIÓN POR COMPUTADORA CON ROS

Daniel Castillo López A01737357  
Emmanuel Lechuga Arreola A01736241  
Paola Rojas Domínguez A01737136



# ... **RETO**

Extender el sistema de navegación autónoma para que el robot móvil Puzzlebot sea capaz de detectar, mediante su cámara, el color de un semáforo (rojo, amarillo o verde) y tomar decisiones dinámicas en tiempo real. Para lograrlo, se debe integrar visión por computadora con control de lazo cerrado, implementar una capa de toma de decisiones robusta y analizar el rendimiento del sistema con métricas cuantitativas y gráficas.

# OBTETIVOS

## General

Diseñar e implementar un sistema de navegación autónoma que combine visión artificial y control de lazo cerrado, permitiendo al Puzzlebot tomar decisiones de movimiento en función del color de un semáforo detectado por su cámara.

## Particulares



Desarrollar un algoritmo de visión por computadora capaz de detectar los colores rojo, amarillo y verde en un semáforo con precisión y robustez.



Implementar una lógica de decisión que modifique el comportamiento del robot según el color detectado.



Integrar el sistema de visión y decisión con el controlador de navegación.



Ajustar y validar el comportamiento del controlador para que sea robusto frente a perturbaciones, ruido y no linealidades.



# METODOLOGÍA

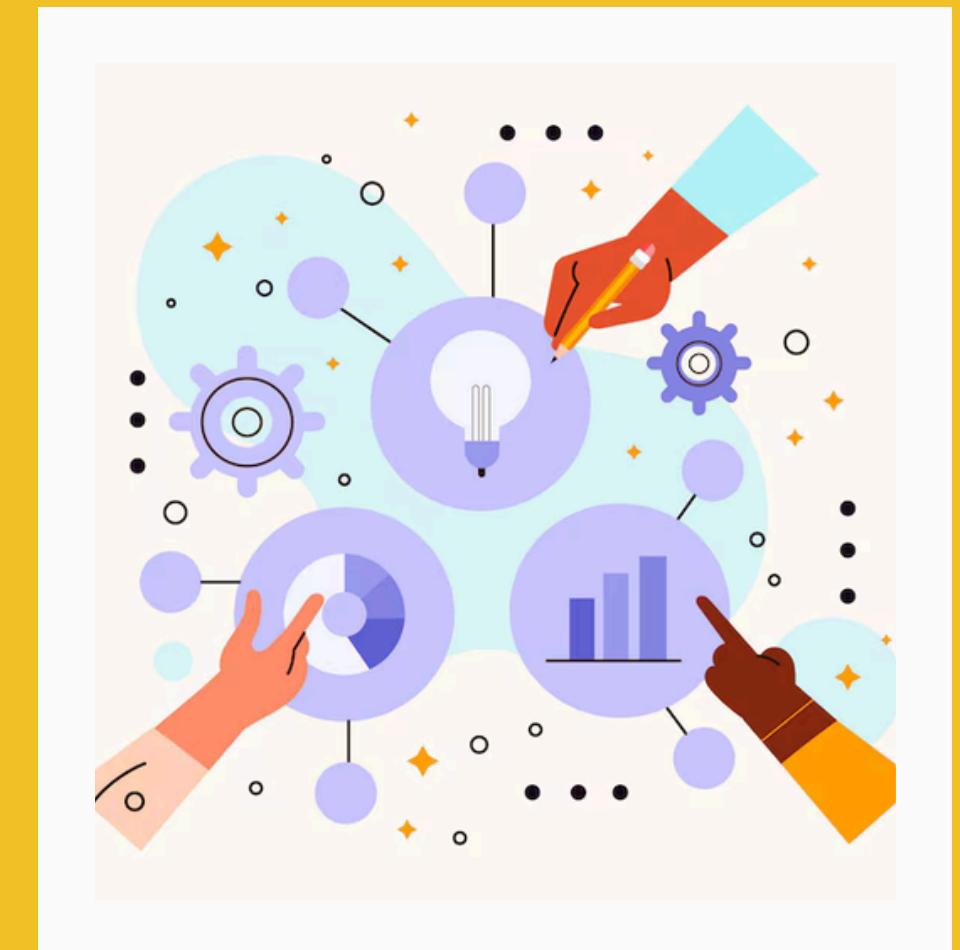
1. Preparación del entorno.
2. Adquisición de imagen.
3. Detección del semáforo.
4. Decisión y control.
5. Integración y prueba final.



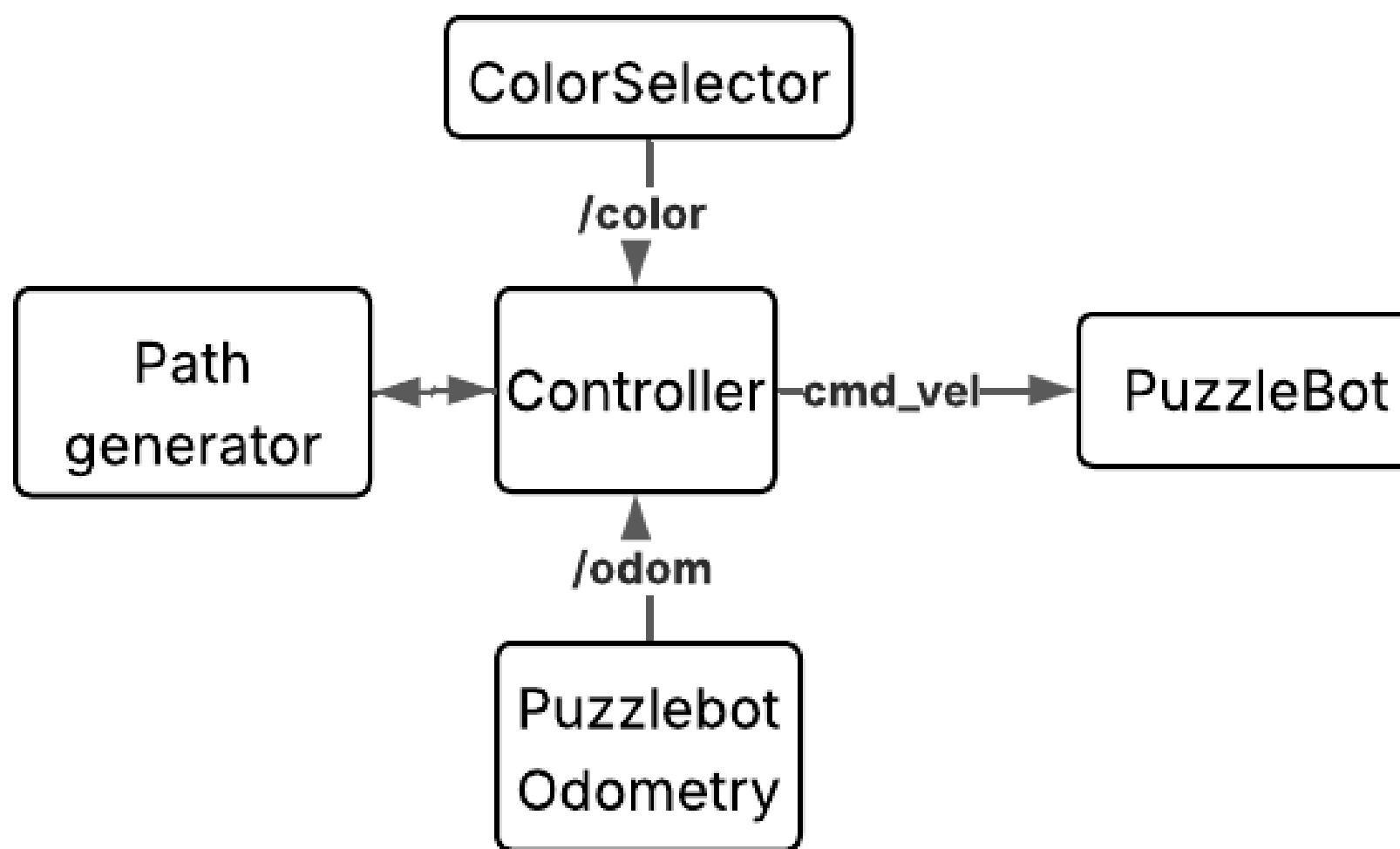


# METODOLOGÍA

- Verificar el funcionamiento del robot y cámara.
- Creación o ocupar un nodo que se suscribe a el nodo /image\_raw.
- Visualizar en tiempo real la img, en openCV, asegurando que la camara funcione.
- Implementar el nodo de detección con OpenCV
  - Convertir imagen a HSV.
  - Aplicar máscara para rojo, amarillo y verde.
  - Calcular área o contorno para decidir que luz esta activa.
- Publicar el resultado en el tópico /color
  - Reacción del robot en base a la publicación.
- Integración de los nodos con un archivo launch.
- Ejecución de pruebas.
- Grabar video para el reporte.



# DIAGRAMA INTERCONEXION DEL SISTEMA



**ColorSelector** – Detección de colores → publica en `/color`.

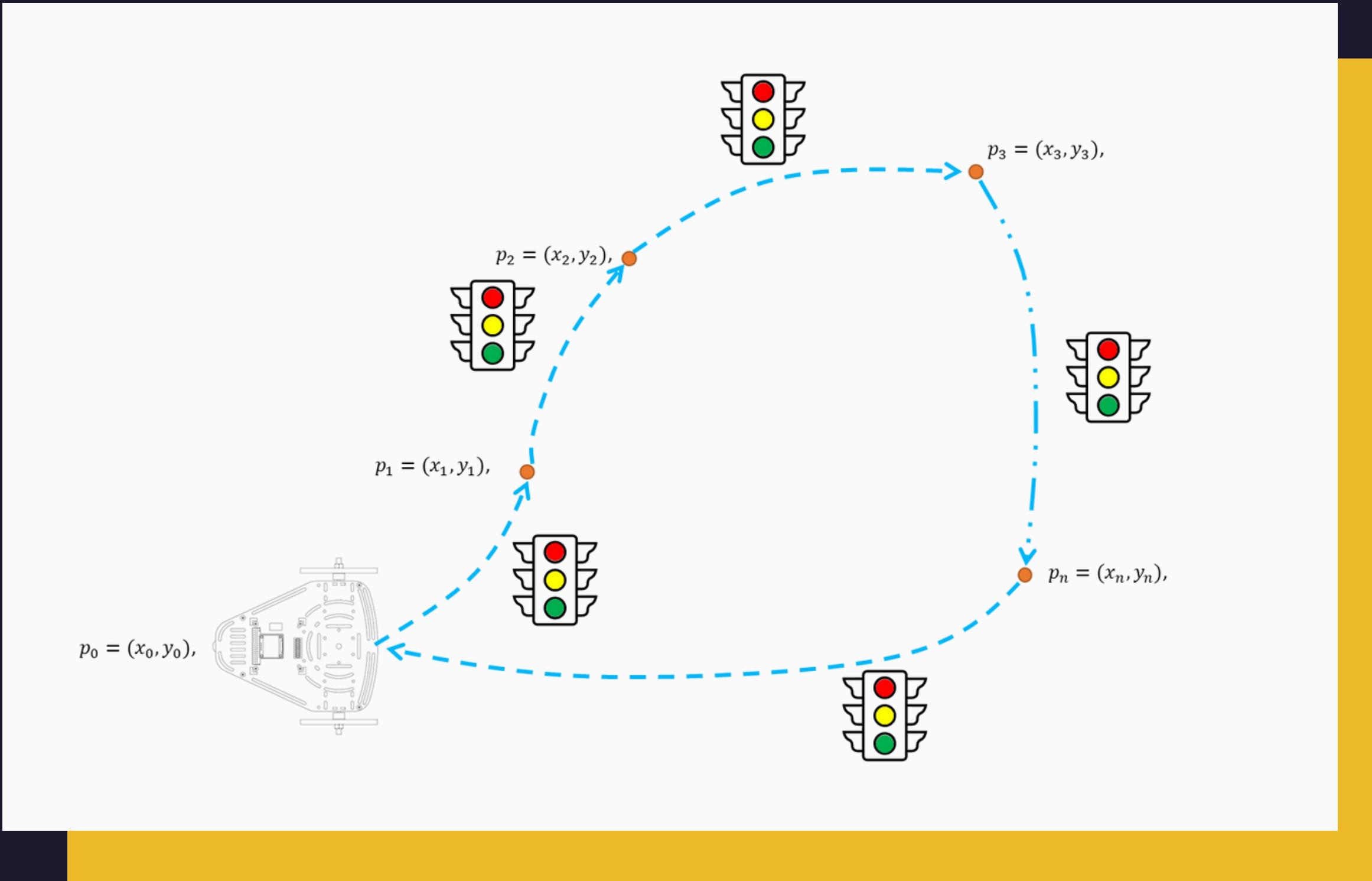
**PathGenerator** – Genera rutas ideales → envía a controller vía `cmd_vel`.

**Controller**– Ajusta `cmd_vel` en tiempo real usando `/odom` y `/color` para seguir la ruta.

**PuzzleVot\_Odometry**– Proporciona posición real del robot → publica en `/Odom`.

**PuzzleBot**– Ejecuta los comandos de `cmd_vel` (motores en el puzzlebot. )

# PRINCIPIO DE FUNCIONAMIENTO



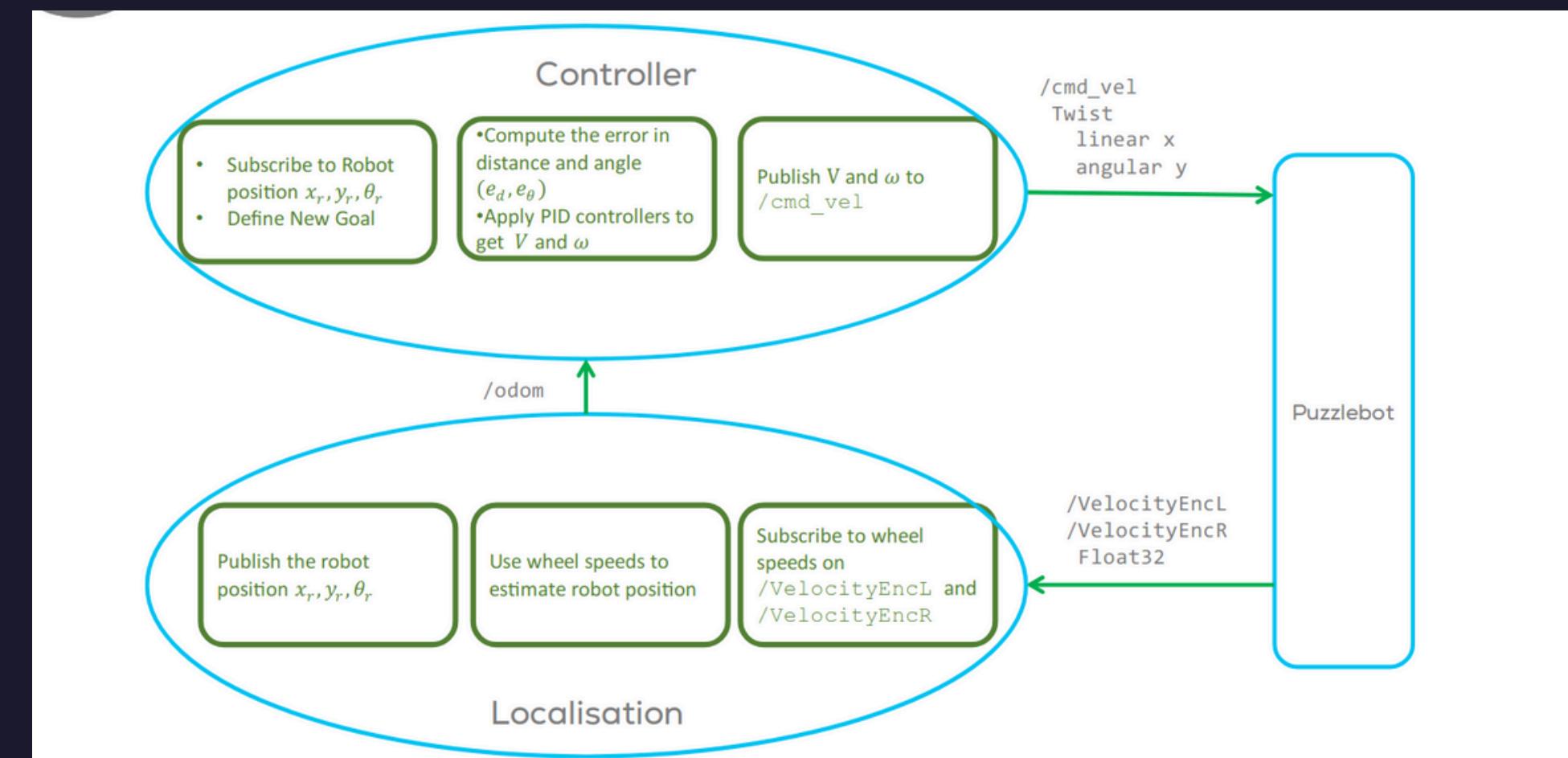
# PRINCIPIO DE FUNCIONAMIENTO

**Path Generator:** Este nodo genera una serie de puntos de referencia (waypoints) que representan la trayectoria que debe seguir el robot. Dicha trayectoria puede ser predefinida o dinámica, y se transmite al nodo Controller.

**Color Selector:** Se encarga de detectar el estado de los semáforos, generando una señal de color (/color) que representa el estado actual del cruce. Esta señal será utilizada por el Controller para modificar el comportamiento del robot

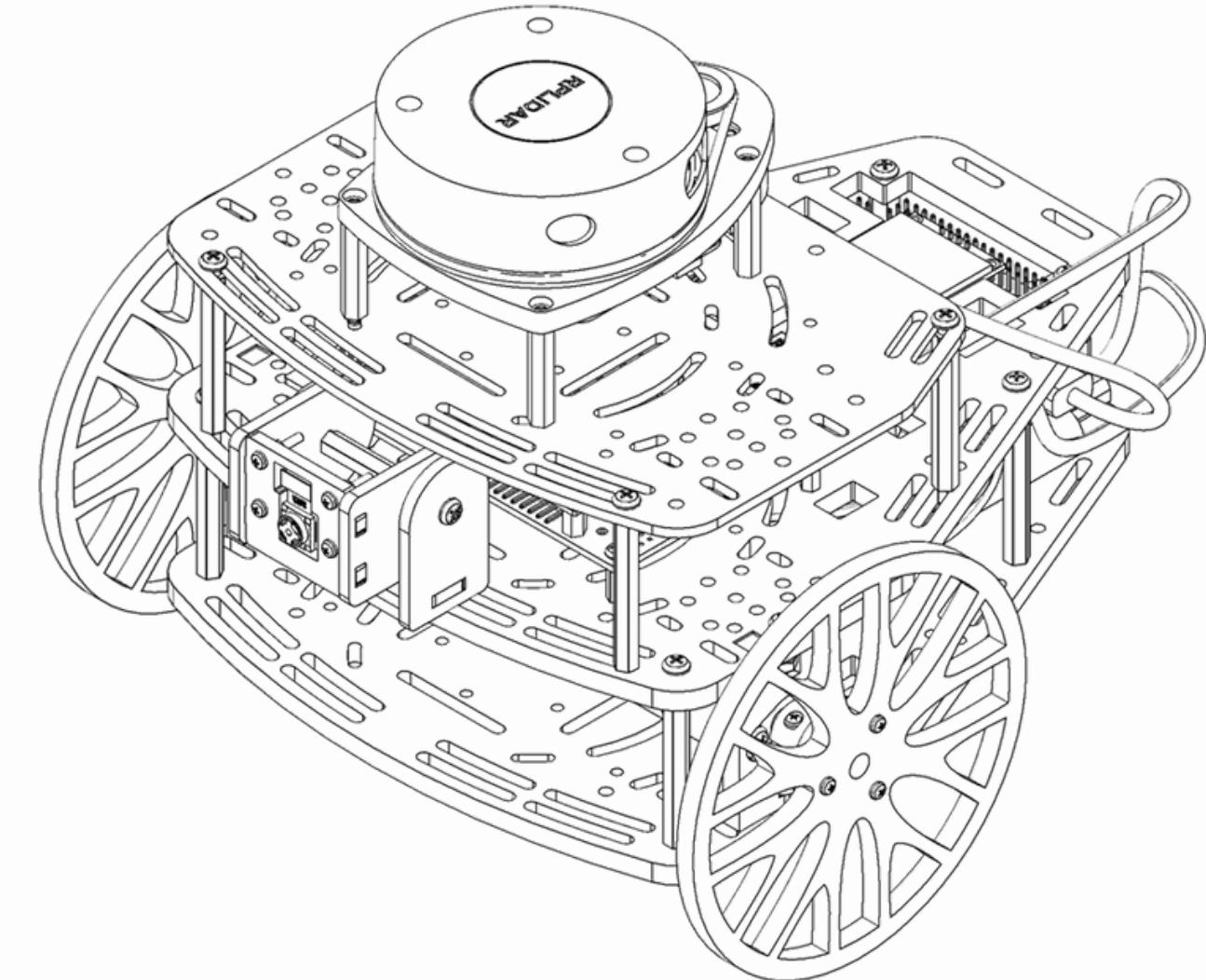
**Puzzlebot Odometry:** Nodo encargado de proporcionar la información de la posición y orientación del robot mediante el tópico /odom. Esta información es fundamental para el control preciso del movimiento y la navegación.

**Controller:** Es el nodo central del sistema. Recibe los puntos deseados desde el Path Generator, el color del semáforo desde ColorSelector y la odometría desde Puzzlebot Odometry. Su función principal es implementar un controlador PID que permita al robot seguir la trayectoria con precisión, minimizando el error entre la posición actual y la deseada.



# PRINCIPIO DE FUNCIONAMIENTO

- **El algoritmo de visión y el controlador deben estar correctamente ajustados.**
- **El controlador debe tener en cuenta las perturbaciones, no linealidades y el ruido.**
- **Utilizar un archivo de configuración o un parámetro en el archivo de lanzamiento**



Puzzlebot

# Máquina de estados



## STOP

- Alto cuando vea luz roja
- hasta que vea una luz verde.



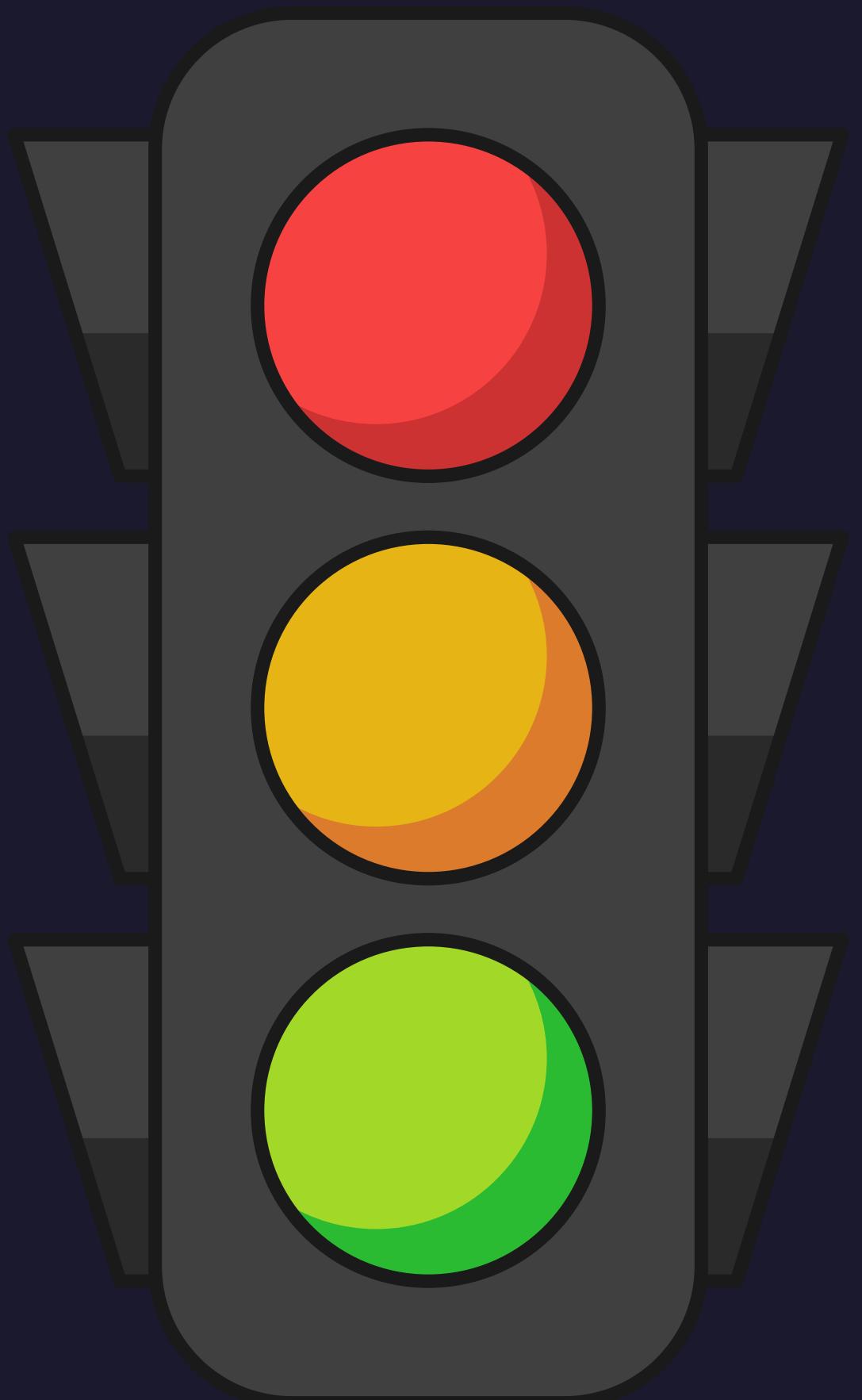
## SLOW

- Cuando vea una luz amarilla, conduzca lentamente
- Hasta que vea una luz roja para detenerse.



## GO

- Al ver luz verde continuar con el recorrido



# Algoritmos de visión por computadora

01

## Conversión de BGR a HSV

- Se convierte la imagen de la cámara del formato BGR a HSV.
- Facilita la separación de colores independientemente de la iluminación.

02

## Segmentación por Color con cv.inRange()

- Se crean máscaras binarias que aíslan los colores:
- Para el rojo se usan dos rangos HSV porque está en los extremos del espacio de matiz.
- Verde y amarillo se dividen en dos rangos cada uno para mejorar la cobertura en presencia de luces brillantes.

03

## Filtrado Morfológico (cv.morphologyEx)

Se aplica apertura morfológica (erosión + dilatación) para eliminar ruido y pequeños falsos positivos.

04

## Análisis de Área con cv.countNonZero()

- Se cuentan los píxeles activos de cada máscara.
- Se detecta el color del semáforo como el que tenga mayor cantidad de píxeles por encima de un umbral.

# VENTAJAS

Los algoritmos de segmentación por color en el espacio HSV son simples, rápidos y adecuados para detección en tiempo real de objetos con colores bien definidos, como las luces de un semáforo.

## HSV vs BGR

El espacio HSV desacopla el color del brillo, lo cual permite detectar colores de forma más estable bajo diferentes condiciones de iluminación.

## Operaciones morfológicas

Mejoran la calidad de la segmentación al reducir el ruido y falsos positivos por píxeles aislados.

## Umbralización por color

Técnica simple, rápida y efectiva para detectar colores bien definidos como los de un semáforo. No requiere aprendizaje ni datos previos.

## Cálculo de área

El conteo de píxeles permite una decisión cuantitativa y rápida sobre cuál color predomina en la imagen.

# DESVENTAJAS

Debido a la capacidad de cómputo limitada del Puzzlebot, no es viable implementar algoritmos avanzados de visión, lo que obliga a utilizar técnicas simples como la detección por color, que son más vulnerables a errores por iluminación o elementos distractores.

## Business Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Future Plan

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Product Offer

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Target Market

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



# Justificación

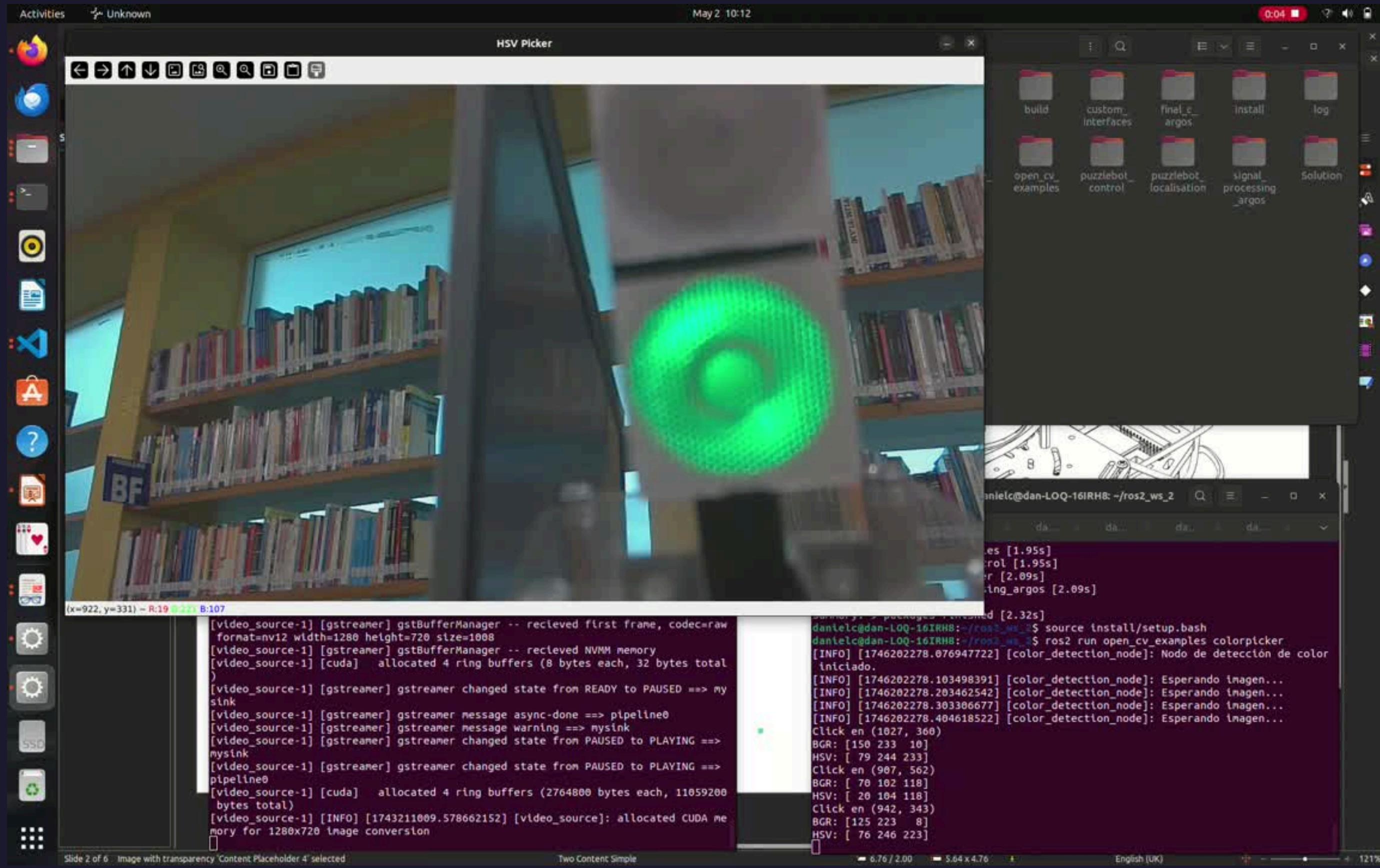
- Se utiliza el espacio de color HSV porque permite una segmentación más estable frente a variaciones de iluminación. Fundamental para identificar correctamente los colores rojo, amarillo y verde del semáforo.
- Se definen múltiples rangos HSV por color, especialmente para el rojo, ya que se ubica en los extremos del espectro HSV. Esta estrategia mejora la tolerancia a condiciones variables de luz.
- Se aplican operaciones morfológicas para eliminar ruido y mejorar la detección de regiones de interés.
- Cada color se procesa por separado. Esta separación facilita la integración con otros nodos del sistema, como el controlador de navegación.
- La arquitectura modular y el procesamiento de imagen aseguran una detección clara y lista para mejoras

# Avances

\*\*\*



# Avances



...

# GRACIAS

