

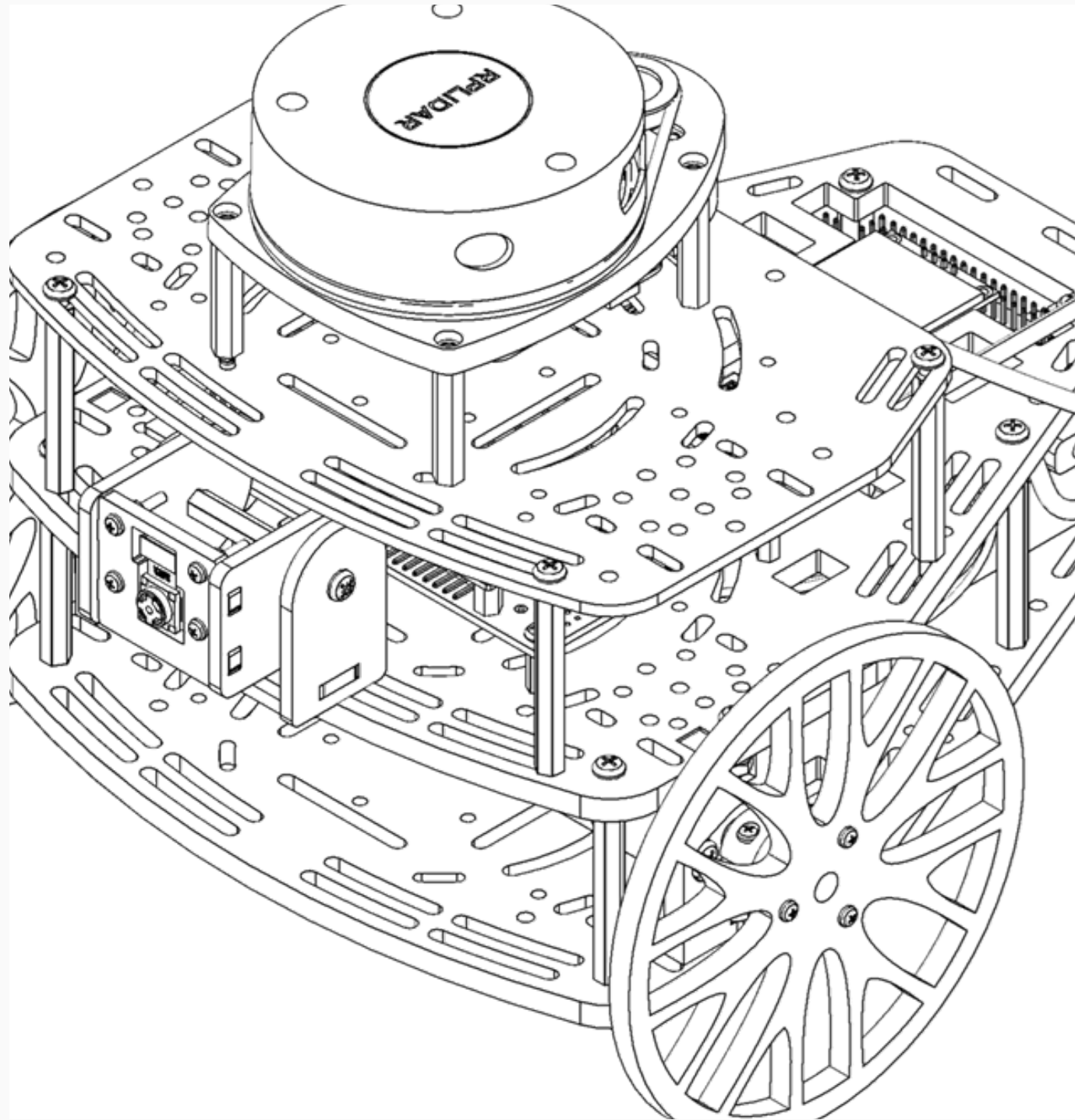


# CHALLENGE **LINE FOLLOWER**



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 dar seguimiento a una pista en base a las líneas en la que se encuentra, además 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.



# OBTETIVOS

## General

Mediante la cámara poder identificar conceptos como el seguidor de línea pero también mantener el análisis de color del semáforo, para poder influir en el desplazamiento del puzzlebot.

## Particulares

Implementar una lógica de seguidor de línea para establecer el error dentro del campo de visión

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

Integrar ambos sistemas 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

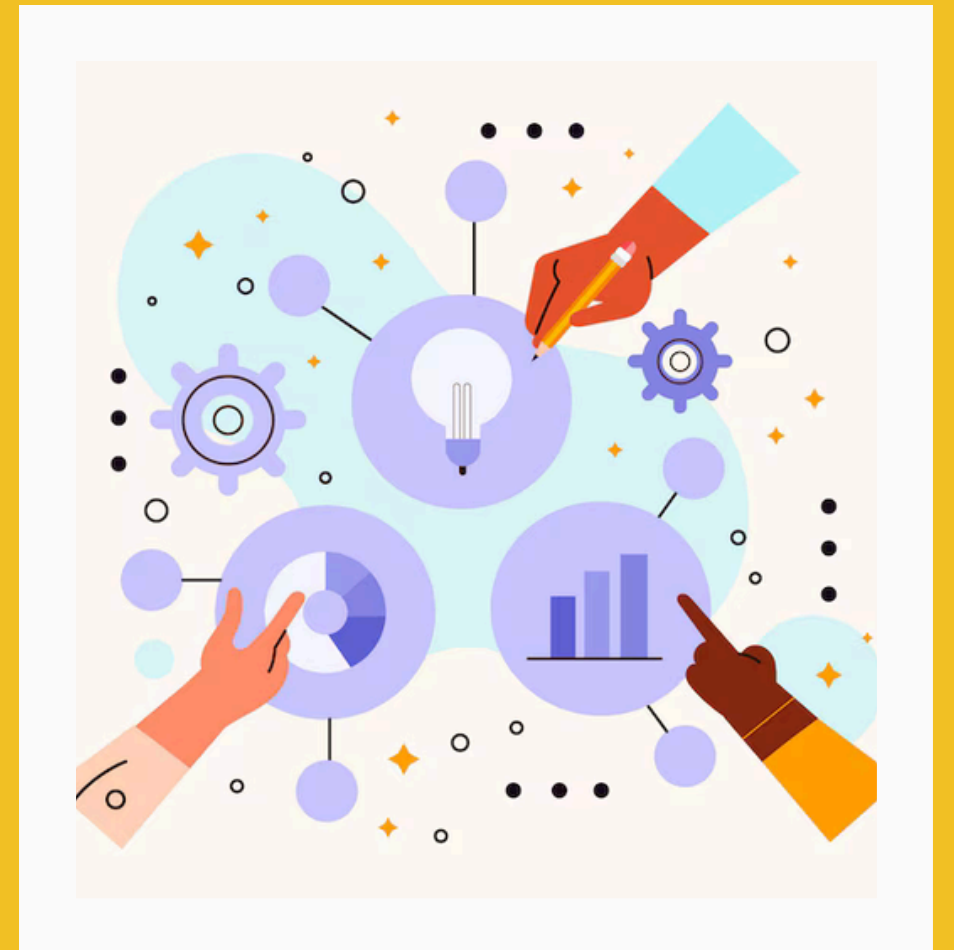
1. Creación de pista.
2. Adquisición de imagen.
3. Integración del semáforo.
4. Decisión y control.
5. Integración y prueba final.





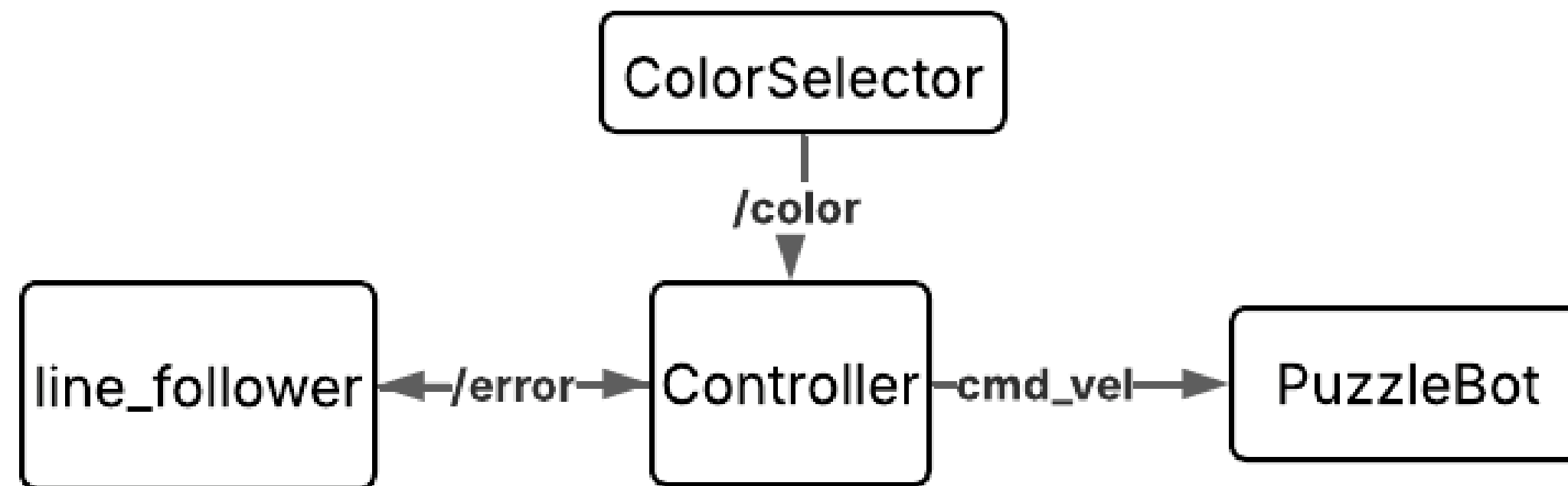
# METODOLOGÍA

- Creación o ocupar un nodo que se suscribe a el nodo `/image_raw`.
- Visualizar en tiempo real la img, en openCV, estableciendo un recorte para mantener la visión en la pista
- Implementar el nodo de detección de línea con OpenCV
  - Convertir imagen a escala de grises
  - Umbralización binaria inversa
  - Calcular centroide
- Publicar el resultado en el tópico `/error`
  - Reacción del robot en base a la publicación.
- Integración del semáforo
- Integración de los nodos con un archivo launch.
- Ejecución de pruebas.





# SYSTEM INTERCONNECTION DIAGRAM



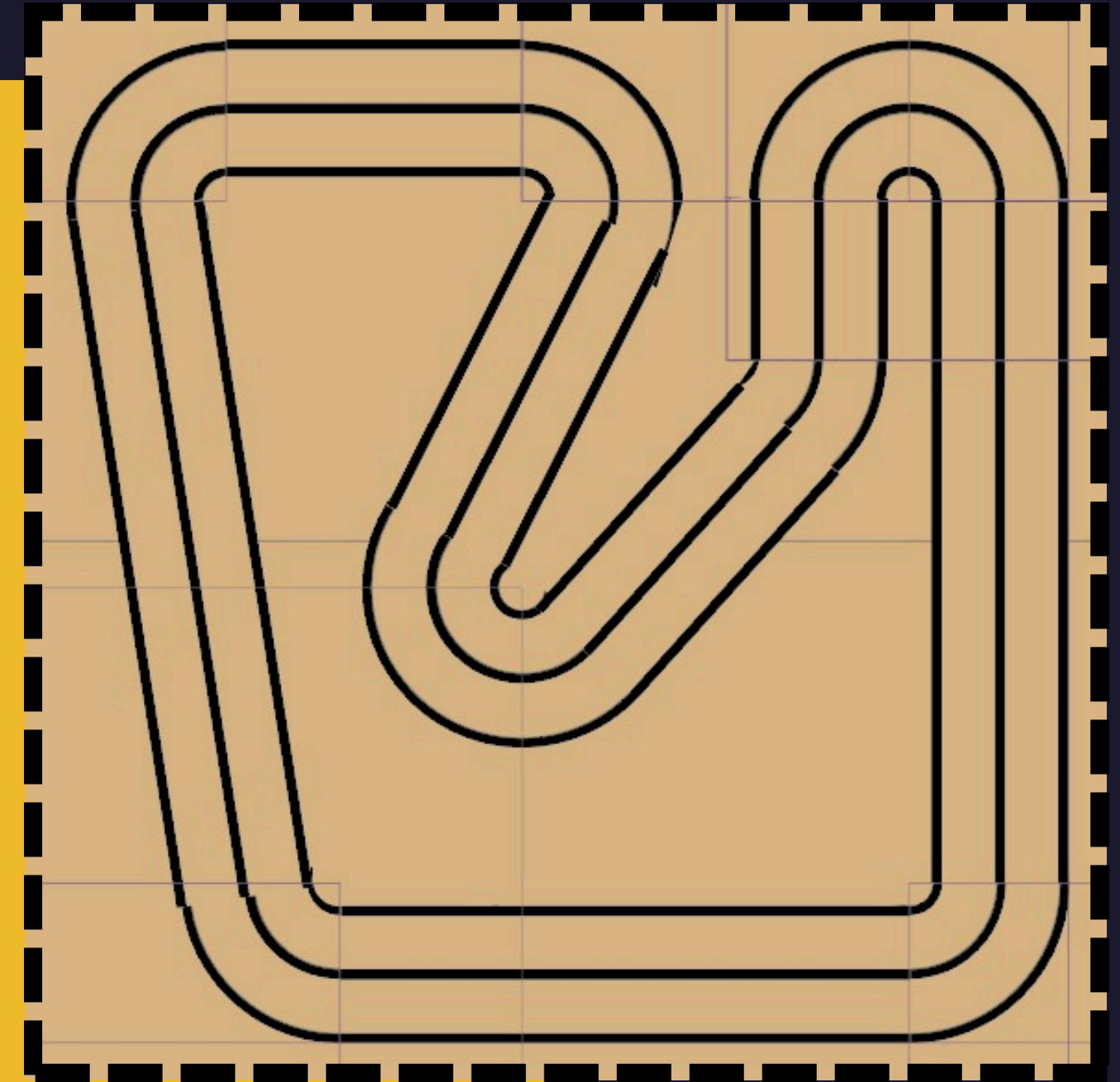
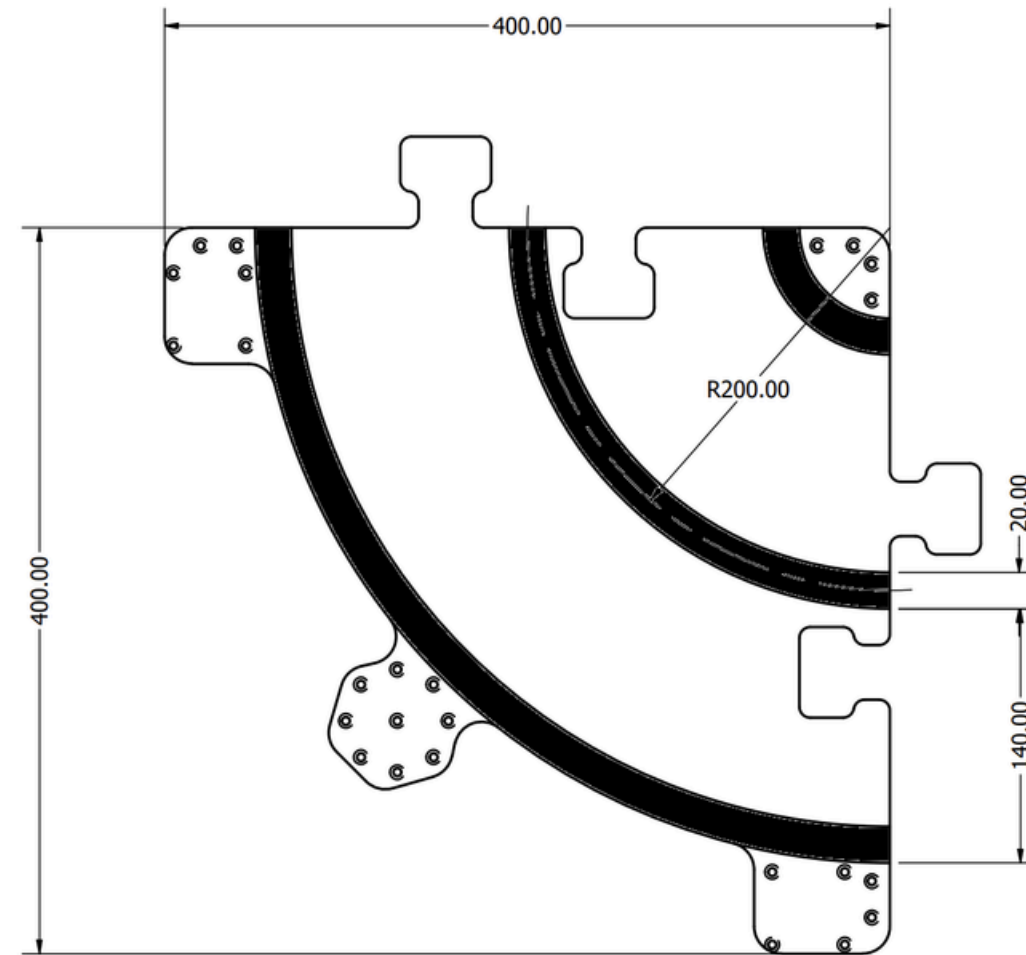
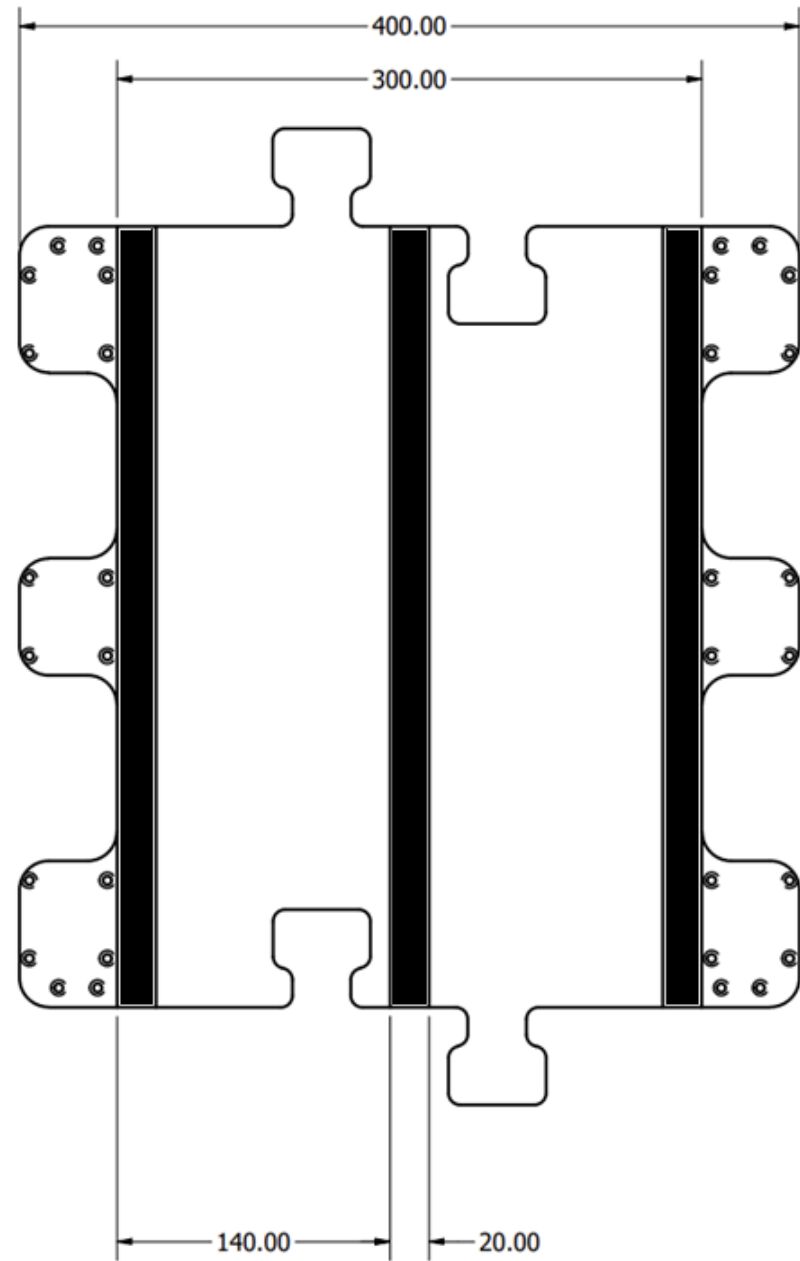
**ColorSelector** – Detección de colores → publica en /color.

**Line\_Follower**– Hace la visión del seguidor de línea y calcula el error→ envía el error a controller por medio de /error

**Controller**– Ajusta cmd\_vel en tiempo real usando /odom y /color para seguir la ruta.

**PuzzleBot**– Ejecuta los comandos de cmd\_vel (motores en el puzzlebot. )

# PRINCIPIO DE FUNCIONAMIENTO

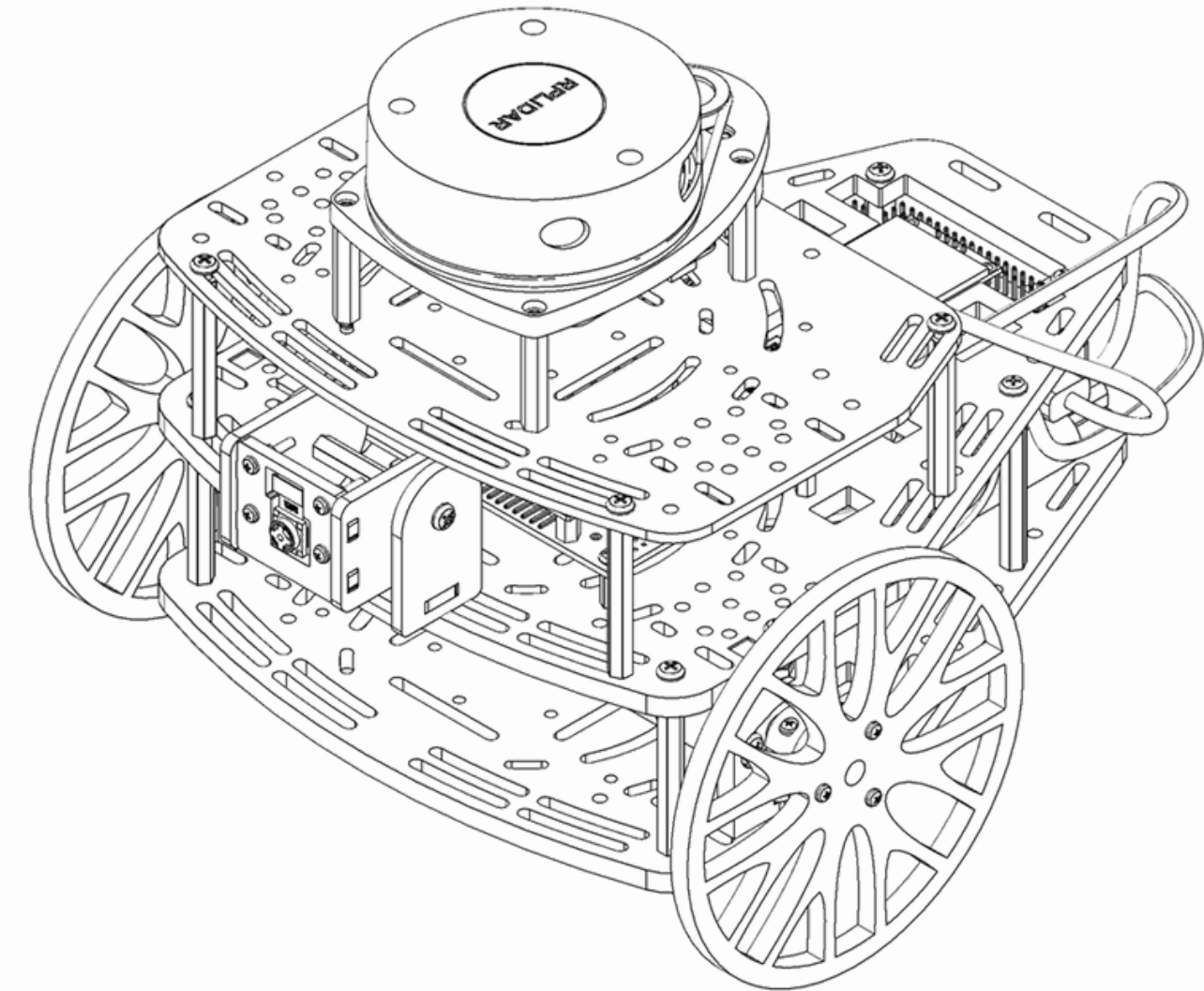


# CONTROLLER



E

- El algoritmo de visión y el controlador se unen para tener un controlador proporcional
- Establecemos velocidad lineal constante lo que permite el robot avanzar mientras ajusta su orientación
- Publica comandos de movimiento en el tópico `/cmd_vel`



Puzzlebot



# Máquina de estados



## STOP

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



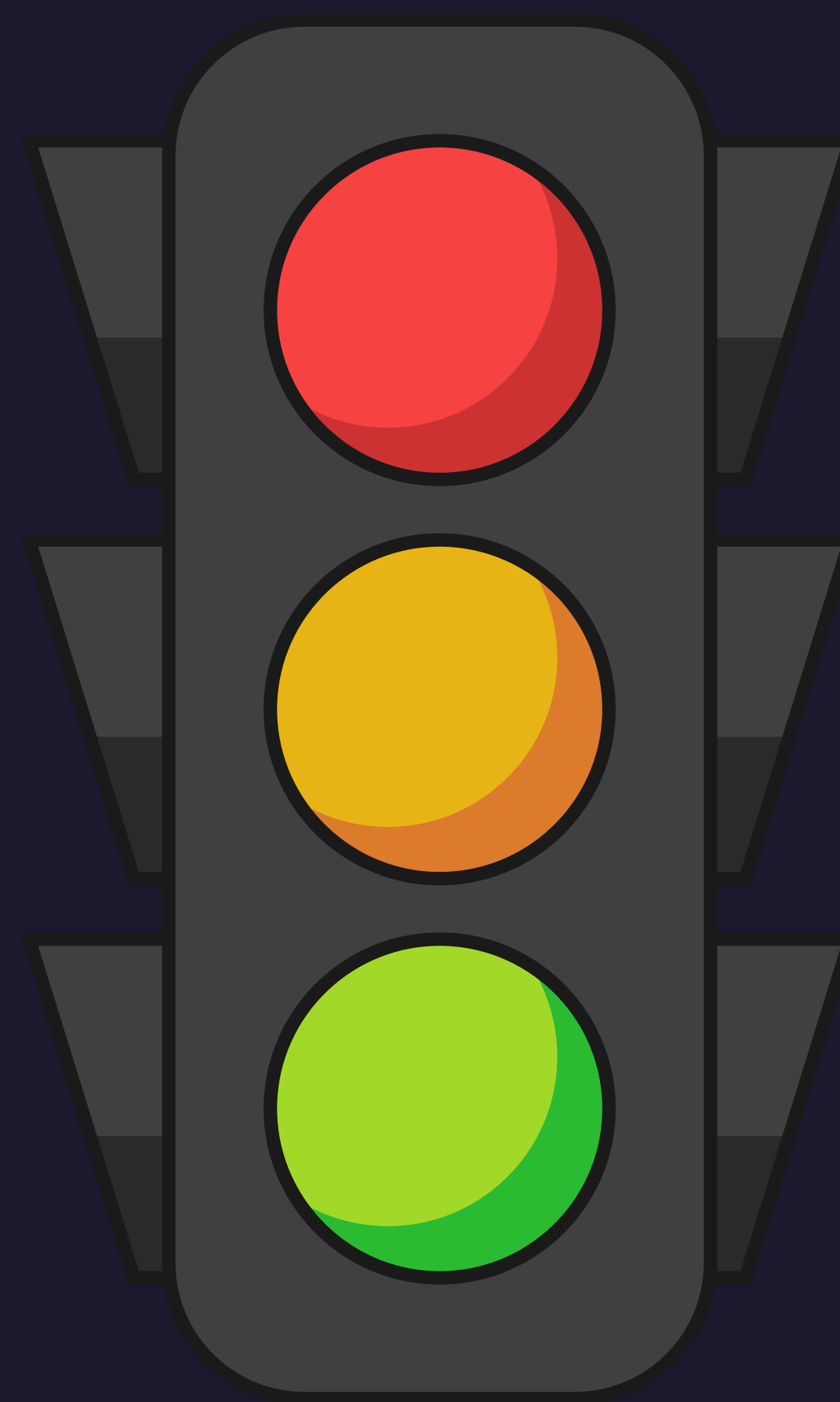
## SLOW

- 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 para la detección de línea

## 01 Conversión a escala de grises

- Reducir la imagen de color a una sola canal de intensidad, lo cual simplifica el procesamiento.

## 02 Umbralización binaria inversa

- Convertir la imagen en blanco y negro. Los píxeles más oscuros (líneas negras) se convierten en blancos (255) para facilitar su detección.

## 03 Filtrado Morfológico

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

## 04 Detección de contornos

- Detectar los bordes de las regiones blancas en la imagen binaria. Se usa para encontrar los blobs (posibles líneas)

# Algoritmos para la detección de línea

## 05 Cálculo de centroide mediante momentos

- Determinar el centro geométrico de cada contorno relevante, útil para saber dónde está cada línea.

## 06 Asignación de etiquetas por proximidad

- Clasificar los centroides detectados en "left", "center" y "right" basándose en su posición horizontal y su historial. Esto permite robustez ante detecciones inconsistentes.

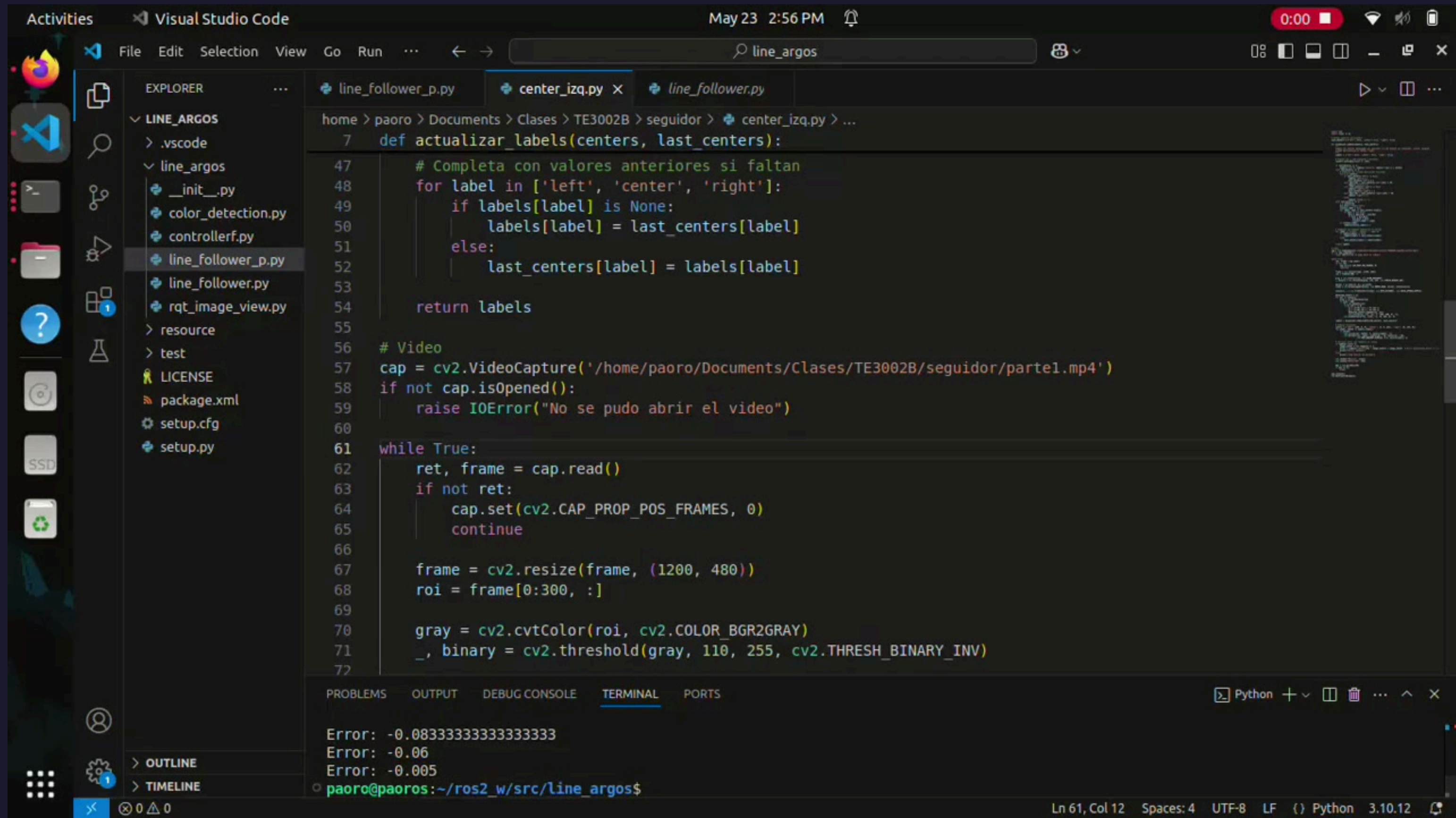
## 07 Cálculo del error de seguimiento

- Determinar el error de alineación entre la línea central detectada y el centro de la imagen. Este valor puede ser usado por un controlador para corregir la trayectoria del robot.

## 08 Visualización con anotaciones

- Dibujar sobre la imagen para depurar y mostrar visualmente los centros detectados y su clasificación.

# Avances



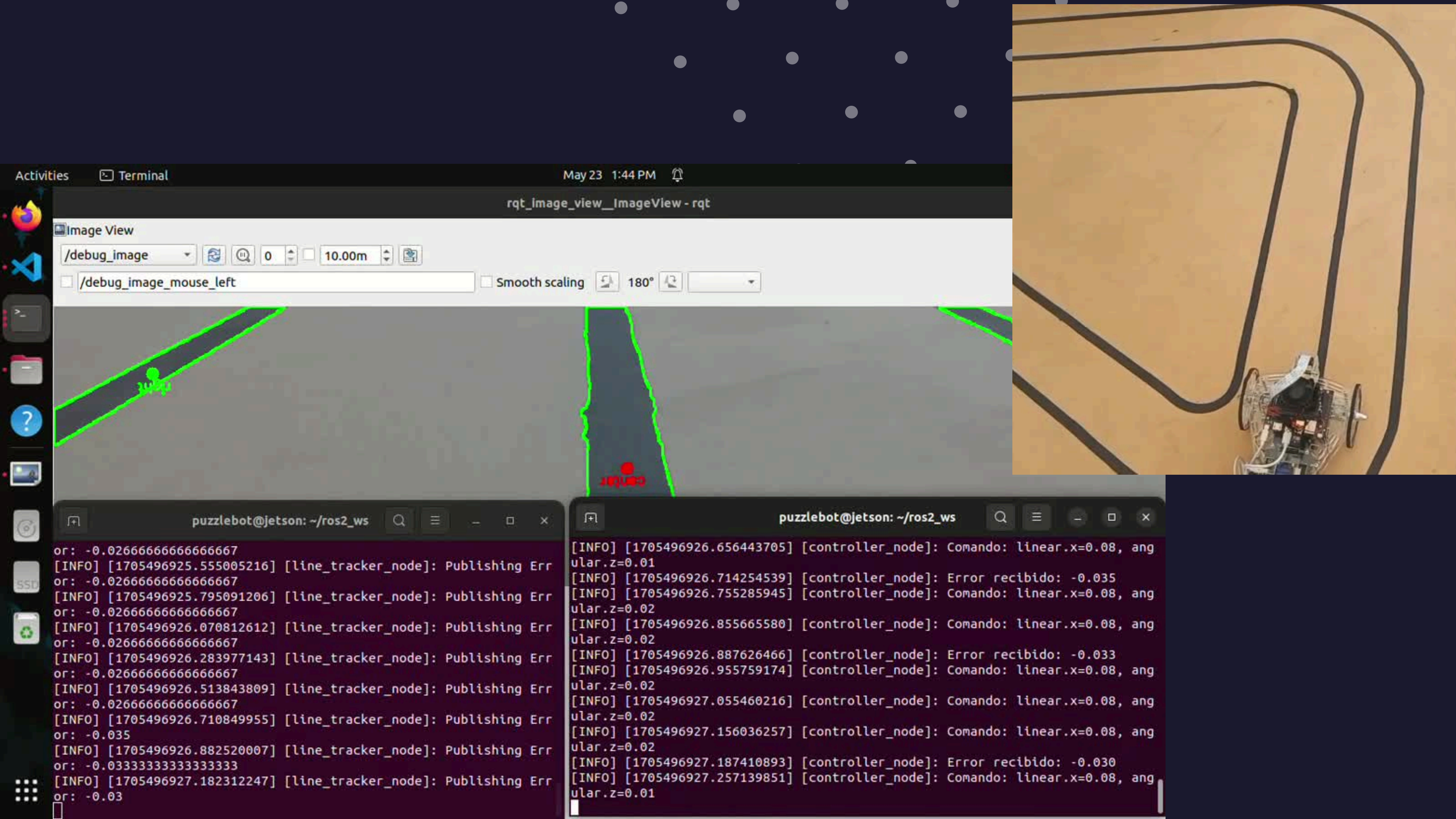
Visual Studio Code interface showing a Python script for line following. The Explorer sidebar shows a project named 'LINE\_ARGOS' with files like 'line\_follower\_p.py' and 'line\_follower.py'. The main editor shows the code for 'center\_izq.py'.

```
def actualizar_labels(centers, last_centers):  
    # Completa con valores anteriores si faltan  
    for label in ['left', 'center', 'right']:  
        if labels[label] is None:  
            labels[label] = last_centers[label]  
        else:  
            last_centers[label] = labels[label]  
    return labels  
  
# Video  
cap = cv2.VideoCapture('/home/paoro/Documents/Clases/TE3002B/seguidor/partel.mp4')  
if not cap.isOpened():  
    raise IOError("No se pudo abrir el video")  
  
while True:  
    ret, frame = cap.read()  
    if not ret:  
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)  
        continue  
  
    frame = cv2.resize(frame, (1200, 480))  
    roi = frame[0:300, :]  
  
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)  
    _, binary = cv2.threshold(gray, 110, 255, cv2.THRESH_BINARY_INV)
```

Terminal output:

```
Error: -0.08333333333333333  
Error: -0.06  
Error: -0.005  
paoro@paoros:~/ros2_w/src/line_argos$
```





Activities Terminal

May 23 1:44 PM

rqt\_image\_view\_ImageView - rqt

Image View

/debug\_image

0

10.00m

/debug\_image\_mouse\_left

Smooth scaling

180°

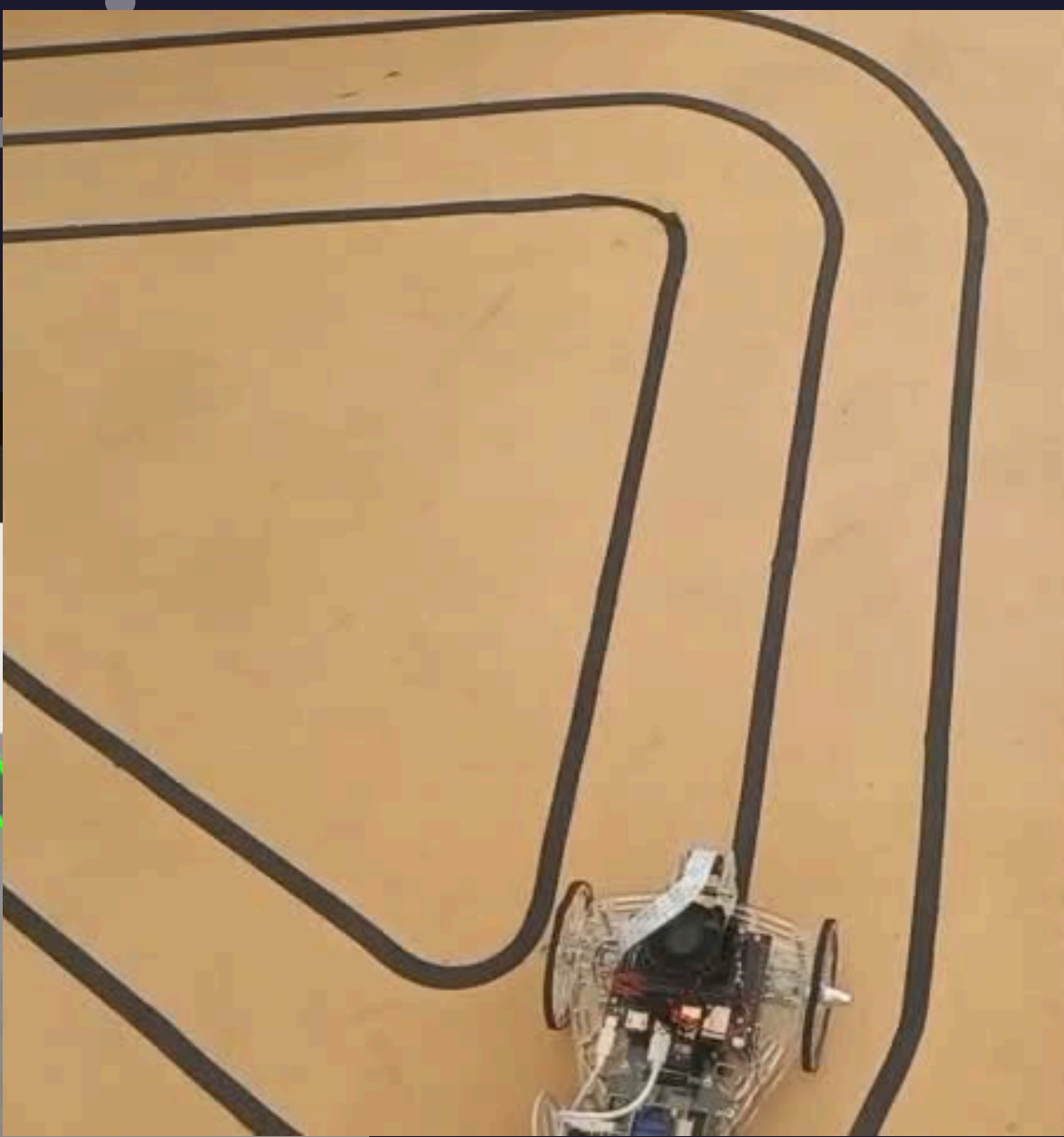


puzzlebot@jetson: ~/ros2\_ws

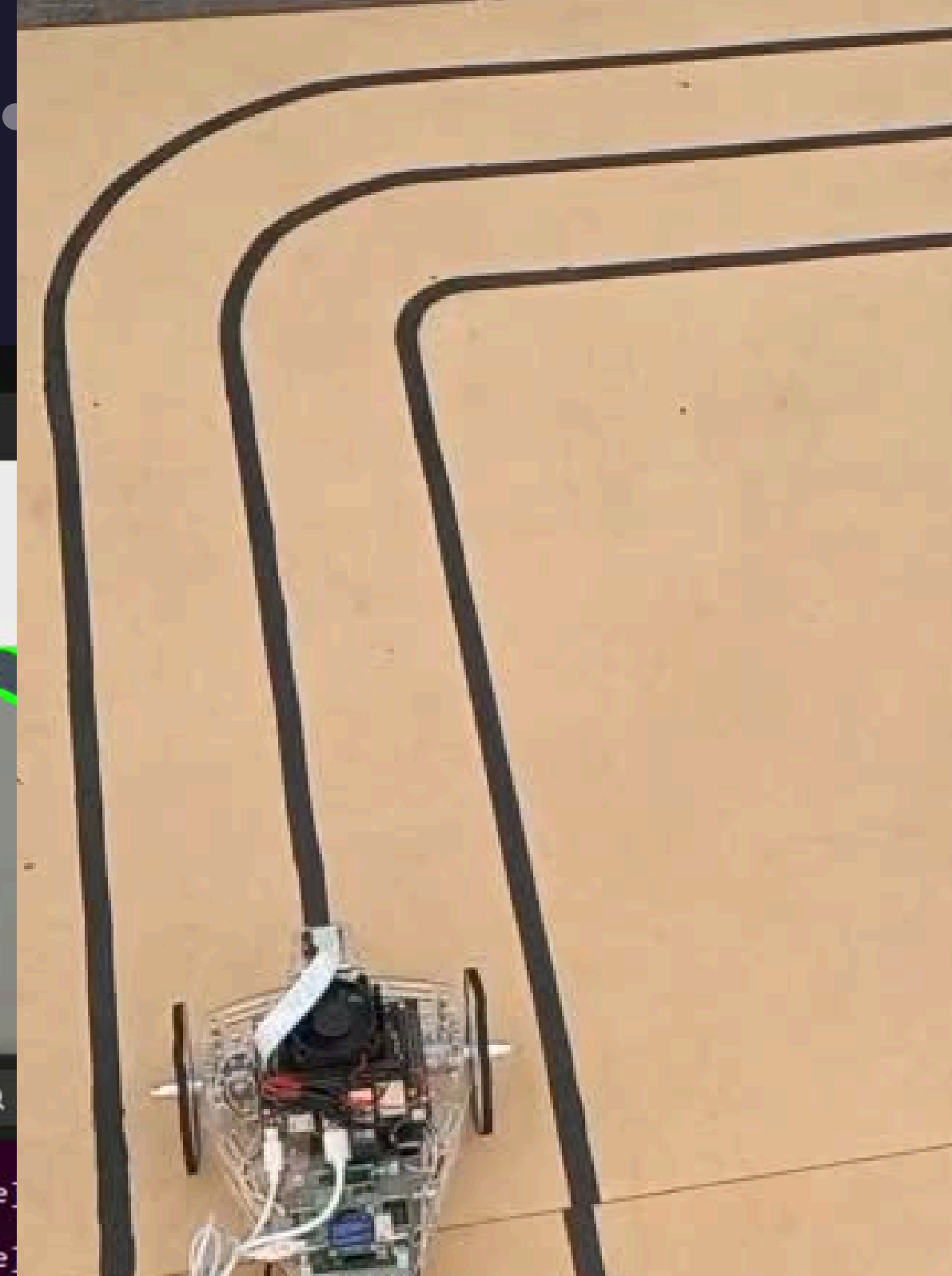
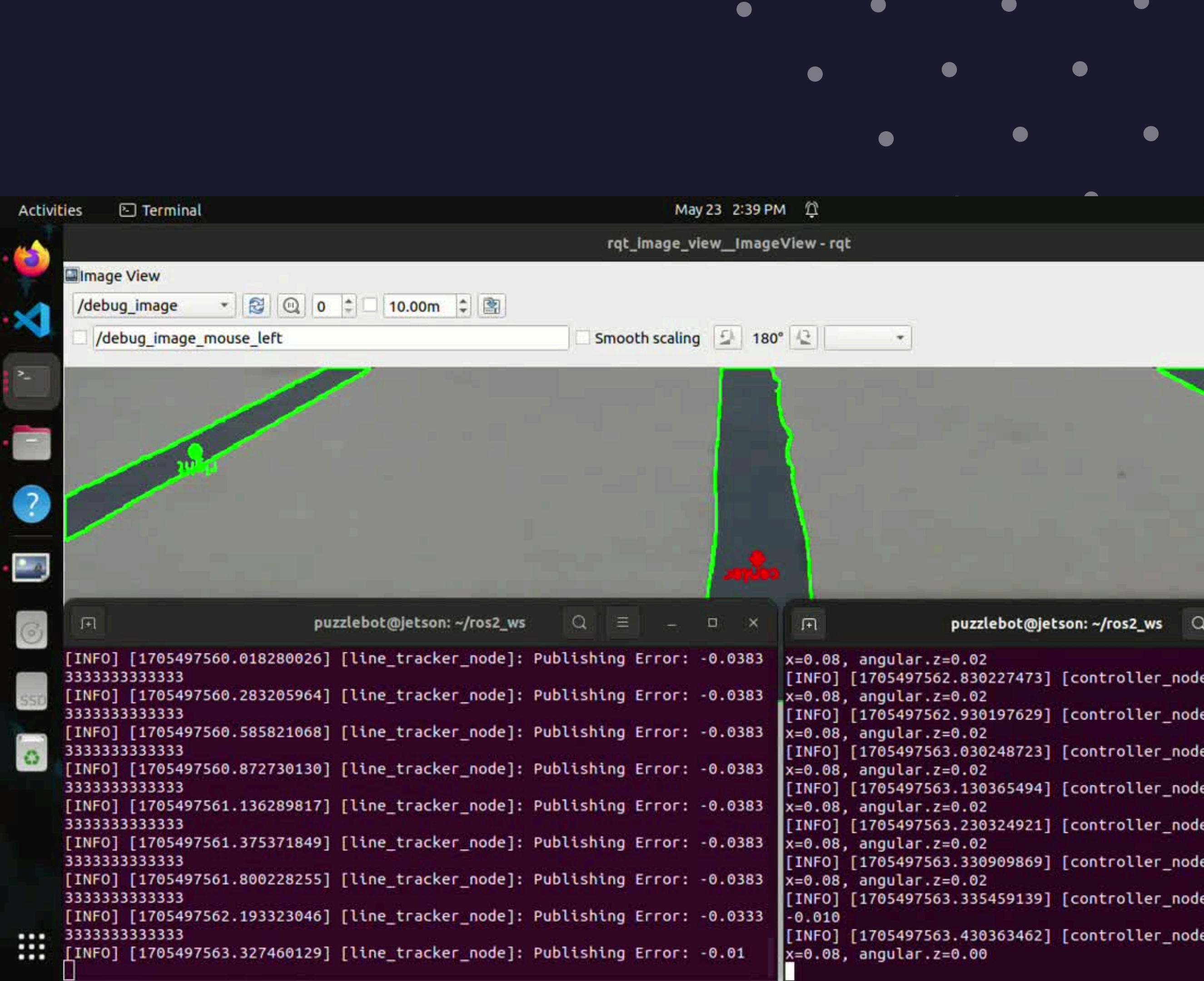
```
or: -0.02666666666666667
[INFO] [1705496925.555005216] [line_tracker_node]: Publishing Error: -0.02666666666666667
[INFO] [1705496925.795091206] [line_tracker_node]: Publishing Error: -0.02666666666666667
[INFO] [1705496926.070812612] [line_tracker_node]: Publishing Error: -0.02666666666666667
[INFO] [1705496926.283977143] [line_tracker_node]: Publishing Error: -0.02666666666666667
[INFO] [1705496926.513843809] [line_tracker_node]: Publishing Error: -0.02666666666666667
[INFO] [1705496926.710849955] [line_tracker_node]: Publishing Error: -0.035
[INFO] [1705496926.882520007] [line_tracker_node]: Publishing Error: -0.03333333333333333
[INFO] [1705496927.182312247] [line_tracker_node]: Publishing Error: -0.03
```

puzzlebot@jetson: ~/ros2\_ws

```
[INFO] [1705496926.656443705] [controller_node]: Comando: linear.x=0.08, angular.z=0.01
[INFO] [1705496926.714254539] [controller_node]: Error recibido: -0.035
[INFO] [1705496926.755285945] [controller_node]: Comando: linear.x=0.08, angular.z=0.02
[INFO] [1705496926.855665580] [controller_node]: Comando: linear.x=0.08, angular.z=0.02
[INFO] [1705496926.887626466] [controller_node]: Error recibido: -0.033
[INFO] [1705496926.955759174] [controller_node]: Comando: linear.x=0.08, angular.z=0.02
[INFO] [1705496927.055460216] [controller_node]: Comando: linear.x=0.08, angular.z=0.02
[INFO] [1705496927.156036257] [controller_node]: Comando: linear.x=0.08, angular.z=0.02
[INFO] [1705496927.187410893] [controller_node]: Error recibido: -0.030
[INFO] [1705496927.257139851] [controller_node]: Comando: linear.x=0.08, angular.z=0.01
```









# THANK YOU

