



Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE LA CIBERSEGURIDAD

Curso Académico 2023/2024

Trabajo Fin de Grado

**APROXIMACIÓN A LA CONDUCCIÓN AUTOMÁTICA DE JUGUETES
TELEDIRIGIDOS**

Autor: Pablo Gracia Correa

Directores: Jonathan Crespo Herrero
Gustavo Recio Isasi

Resumen

El trabajo trata sobre el desarrollo de un sistema basado en visión artificial diseñado para mejorar la interacción y control de elementos teledirigidos. La principal motivación para realizar este estudio es ofrecer al usuario una interfaz basada en la visión global del entorno con la que poder controlar de manera sencilla elementos teledirigidos. Este sistema puede ser usado en distintos escenarios como el mantenimiento de un jardín por un cortacésped o el estacionamiento autónomo en un aparcamiento de vehículos.

Los objetivos del trabajo se enfocan a establecer una conexión entre el ordenador y el robot que permita la transmisión de instrucciones, desarrollar un programa que sea capaz de detectar al robot y su orientación, y crear algoritmos que lleven al robot a un punto cualquiera en la imagen y le permitan una trayectoria.

En primer lugar, el proyecto se centra en el procesamiento de imágenes que, mediante una cámara, el sistema capture tanto el robot como su entorno. Esta captura es analizada para determinar la posición del robot y así calcular su orientación tras realizar un movimiento. Esto se consigue debido a que el programa detecta el color del robot.

A partir de estos datos, el sistema realiza el cálculo de instrucciones necesarias para que el robot llegue a un punto cualquiera marcado en la imagen o que siga una trayectoria sin detenerse.

Para llegar a un punto cualquiera se mandan instrucciones en función de la orientación que tiene el robot, se calcula la diferencia de esta orientación con el ángulo al punto y se manda una instrucción, después el robot para y se obtiene la orientación del robot y se calcula esta diferencia otra vez, y así continuamente hasta que llegue al punto final.

Para seguir la línea la mayor complicación ha sido mandar las instrucciones sin que el robot se detuviera, porque las instrucciones se tienen que mandar en tiempo real para que el robot corrigiera su rumbo continuamente, esto se consigue mediante retroalimentación. En todo momento se están calculando datos del robot como su orientación, el ángulo con el punto más cercano de la línea, el ángulo de la línea en ese punto... para poder enviar al robot la instrucción necesaria para permanezca en la línea.

Realizando el algoritmo para que siga la línea se ha encontrado un problema difícil de solucionar que es la latencia que tienen las instrucciones enviadas por el ordenador hasta que llegan al robot, que hace que el robot sea incapaz de seguir la línea correctamente.

Para solucionarlo se ha intentado plantear otros métodos, el primero utiliza el algoritmo se seguir una línea desarrollada anteriormente, pero cambiando que, cuando el robot está a una cierta distancia el programa para y lo devuelve a la línea. El segundo intenta estimar la posición y orientación del robot en el momento que le llega la instrucción. Y la última opción es ralentizar al robot cuando se esté aproximando al giro de la línea, ya que el robot se desvía con mayor facilidad en las curvas. Esto se logra mediante un PWM, que es un método mediante el cual se le mandan pequeños pulsos de activación al robot para que avance más despacio.

Finalmente, todas estas instrucciones son transmitidas al robot mediante el uso de su mando. Las instrucciones son enviadas por el ordenador a un GPIO, que es un chip con varios pines que se pueden controlar en tiempo de ejecución. Y gracias a una protoboard, está conectado al mando.

Además, se ha realizado un simulador, llamado “gemelo virtual”, que es un programa que representa al robot como un punto en la misma captura de imagen. Esta simulación incluye los errores de precisión que tiene el robot tanto en sus giros como avanzando y retrocediendo. Con este modelo se pueden probar los algoritmos de movimiento y navegación sin necesidad de realizar pruebas físicas.

Para acabar se han realizado una serie de experimentos para recoger variedad de datos de los distintos algoritmos planteados y así saber cómo de eficientes son los algoritmos. En el caso de seguir una línea también se comparan entre ellos para llegar a la conclusión de cuál es mejor.

Palabras Clave

Robot, Visión Artificial, Algoritmo, Píxel, Punto, Línea.

Índice de contenido

1.- Introducción.....	1
1.1- Motivación.....	1
1.2- Objetivos	2
1.3- Estructura del documento.....	2
2.- Estudio de alternativas	3
2.1 Conexiones.....	3
2.2 Detección del robot	4
2.3 Algoritmos de movimiento	5
3. Diseño.....	7
3.1 Diseño de la comunicación	9
3.1.2 Comunicación de la cámara con el robot.....	9
3.1.2 Comunicación del ordenador con el robot	10
3.2 Diseño de la movilidad.....	13
3.2.1 Diseño de algoritmo para llegar a un punto cualquiera	13
3.2.2 Diseño del algoritmo seguir una línea	14
4. Implementación	16
4.1 Movimiento del Robot.....	16
4.1 Detección del robot	16
4.2 Cálculo de la orientación del robot	17
4.3 Calibración de distancia.....	19
4.4 Calibración de giro	20
4.5 Gemelo virtual.....	22
4.6. Implementación de superposiciones gráficas	22
4.7 Movimiento en línea recta del robot	24
4.8 Llegar a un punto cualquiera	26
4.9 Seguir una línea.....	28
4.10 Latencia.....	35
4.10.1 Seguir línea volviendo a ella	35
4.10.2 Seguir línea estimando donde llega la instrucción	36
4.10.3 Ralentizar el giro del robot.....	40
5. Experimentos	43

5.1 Llegar a un punto cualquiera	43
5.1.1 Llegar a un punto justo delante del robot	43
5.1.2 Llegar a un punto justo detrás del robot	44
5.1.3 Llegar a un punto detrás del robot en diagonal	46
5.2 Seguir una línea recta	47
5.2.1 Seguir una línea recta enfrente del robot	48
5.2.2 Seguir una línea recta perpendicular al robot.....	51
5.3 Seguir una línea curva	56
5.3.1 Seguir una línea con una curva abierta	56
5.3.2 Seguir una línea con una curva cerrada	60
5.3.3 Seguir línea con dos curvas.....	63
6. Conclusiones y trabajos futuros.....	68
Referencias	70

Índice de figuras

Figura 1. Diseño de las comunicaciones en Robot Móvil Autónomo Seguidor Delínea..	3
Figura 2. Diagrama de diseño de Robot Móvil Autónomo Seguidor de Línea con Raspberry Pi y Sistema de Visión.....	4
Figura 3. Robot jugador de fútbol y pelota	4
Figura 4. Detección de la dirección del robot en base al centro de las ruedas y el centro del robot.....	5
Figura 5. Método de seguir línea con un sensor de luz sobre el robot.....	6
Figura 6. Diagrama del diseño del sistema	7
Figura 7. Diagrama de clases del código utilizado	8
Figura 8. Cámara OnePlus 9.....	9
Figura 9. Adafruit FT232H.....	10
Figura 10. Placa del mando del robot	10
Figura 11. Transparencia de la placa del mando con los botones de la placa por el lado contrario	11
Figura 12. Lado contrario de la placa del mando	11
Figura 13. Conexiones de cables con GPIO	11
Figura 14. Mando del robot.....	12
Figura 15. Remote Control Robot 2.4GHz RC Robot Toy for Kids – GILOBABY	12
Figura 16. Ruedas del robot	12
Figura 17. Diseño de movimiento del Robot	13
Figura 18. Diseño algoritmo llegar a un punto cualquiera	14
Figura 19. Diseño algoritmo seguir línea	15
Figura 20. Máscara aplicada para detectar área del color seleccionado	17
Figura 21. Imagen del robot detectado por el programa	17
Figura 22. Ángulos en un sistema de coordenadas en el que la coordenada y crece hacia abajo	18
Figura 23. Ángulos en un sistema de coordenadas común	18
Figura 24. Imagen del robot después de calcular su orientación.....	18
Figura 25. Ejemplo de punto objetivo superpuesto en la imagen.....	23
Figura 26. Ejemplo de línea superpuesta en la imagen.....	23

Figura 27. Ejemplo de ángulo superpuesto en la imagen.....	23
Figura 28. Ejemplo de región superpuesta en la imagen	23
Figura 29. Ejemplo de robot detectado en una región	23
Figura 30. Posición inicial para calibrar movimiento en línea recta	25
Figura 31. Posición final para calibrar movimiento en línea recta	25
Figura 32. Diagrama del algoritmo para llegar a un punto cualquiera	27
Figura 33. Ilustración una lista en python	28
Figura 34. Proceso de normalización de ángulos para que funcione la fórmula de la orientación	30
Figura 35. Algoritmo de seguir una línea.....	31
Figura 36. Gráfica del recorrido del robot utilizando el algoritmo de seguir una línea..	32
Figura 37. Gráfica ampliada del recorrido del robot utilizando el algoritmo de seguir una línea.....	32
Figura 38. Gráfica del recorrido del gemelo virtual utilizando el algoritmo de seguir una línea.....	33
Figura 39. Recorrido del robot tras ajustar el umbral de error entre las orientaciones...	34
Figura 40. Recorrido del robot ampliado tras ajustar el umbral de error entre las orientaciones.....	34
Figura 41. Gráfica del recorrido del robot siguiendo la línea volviendo a ella cuando se aleja	35
Figura 42. Gráfica ampliada del recorrido del robot siguiendo la línea volviendo a ella cuando se aleja.....	36
Figura 43. Gráfica del recorrido del robot estimando el siguiente punto	39
Figura 44. Gráfica del recorrido del robot ampliado estimando el siguiente punto	40
Figura 45. Detección punto de giro en curva cerrada.....	41
Figura 46. Detección punto de giro curva abierta	41
Figura 47. Situación de un punto justo delante del robot.....	43
Figura 48. Situación de un punto justo detrás del robot.....	45
Figura 49. Situación de un punto detrás y en diagonal del robot	46
Figura 50. Situación de una línea recta justo delante del robot.....	48
Figura 51. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta enfrente del robot utilizando el algoritmo de seguir una línea	49

Figura 52. Recorrido del gemelo virtual en una de las pruebas realizadas de seguir una línea recta enfrente del gemelo utilizando el algoritmo de seguir una línea	49
Figura 53. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta enfrente del robot utilizando el algoritmo de seguir una línea y volver a ella	51
Figura 54. Situación de una línea recta perpendicular al robot	51
Figura 55. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta perpendicular al robot utilizando el algoritmo de seguir una línea	53
Figura 56. Recorrido del gemelo virtual en una de las pruebas realizadas de seguir una línea recta perpendicular al gemelo utilizando el algoritmo de seguir una línea	54
Figura 57. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta perpendicular al robot utilizando el algoritmo de seguir una línea	55
Figura 58. Situación de una línea con una curva abierta.....	56
Figura 59. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea.....	57
Figura 60. Recorrido del robot en una de las pruebas realizadas des seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea	58
Figura 61. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea.....	59
Figura 62. Situación de una línea con una curva cerrada.....	60
Figura 63. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea.....	61
Figura 64. Recorrido del robot en una de las pruebas realizadas de una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea	62
Figura 65. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella.....	63
Figura 66. Situación de una línea con dos curvas.....	63
Figura 67. Recorrido del robot en una de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea.....	64
Figura 68. Recorrido del robot en una de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea.....	65
Figura 69. Recorrido del robot en una de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea volviendo a ella	67

Índice de tablas

Tabla 1. Medias de píxeles por 0.1s con distintos tiempos de pulsación	19
Tabla 2. Desplazamiento de píxeles en una de las pruebas de 0.5s de tiempo pulsado ..	20
Tabla 3. Medias de grados por 0.1s con distintos tiempos de pulsación.....	20
Tabla 4. Grados girados en una de las pruebas de 0.1s de tiempo pulsado para el giro lento.....	21
Tabla 5. Nuevas medias obtenidas con nuevas pilas para el robot	21
Tabla 6. Medias de grados por 0.1s con distintos tiempos de pulsación para el giro rápido	21
Tabla 7. Grados girados en una de las pruebas de 0.1s de tiempo pulsado para el giro sobre sí mismo	22
Tabla 8. Datos obtenidos para determinar cuál es el mejor tiempo de los pulsos	24
Tabla 9. Distintas pruebas realizadas con el uso de la fórmula de la orientación con resultados correctos e incorrectos	30
Tabla 10. Distintas pruebas realizadas con el uso de la fórmula de la orientación con todos los resultados correctos	30
Tabla 11. Parámetros del ángulo de movimiento del robot con la instrucción de ir recto	37
Tabla 12. Parámetros del ángulo de movimiento del robot con la instrucción de girar..	37
Tabla 13. Parámetros orientación del robot con la instrucción de ir recto	38
Tabla 14. Parámetros orientación del robot con la instrucción de girar.....	38
Tabla 15. Parámetros de distancia con la instrucción de ir recto.....	38
Tabla 16. Parámetros de distancia con la instrucción de girar	39
Tabla 17. Medias de velocidad en pixeles por 0.1s con distintos tiempos de los pulsos	41
Tabla 18. Datos recogidos de las pruebas realizadas para llegar a un punto delante del robot	44
Tabla 19. Medias recogidas de las pruebas realizadas para llegar a un punto delante del robot	44
Tabla 20. Datos recogidos de las pruebas realizadas para llegar a un punto delante del gemelo virtual	44
Tabla 21. Medias recogidas de las pruebas realizadas para llegar a un punto delante del gemelo virtual	44
Tabla 22. Datos recogidos de las pruebas realizadas para llegar a un punto detrás del robot	45

Tabla 23. Medias recogidas de las pruebas realizadas para llegar a un punto detrás del robot	45
Tabla 24. Datos recogidos de las pruebas realizadas para llegar a un punto detrás del gemelo virtual	46
Tabla 25. Medias recogidas de las pruebas realizadas para llegar a un punto detrás del gemelo virtual	46
Tabla 26. Datos recogidos de las pruebas realizadas para llegar a un punto detrás y en diagonal del robot	47
Tabla 27. Medias recogidas de las pruebas realizadas para llegar a un punto detrás y en diagonal del robot	47
Tabla 28. Datos recogidos de las pruebas realizadas para llegar a un punto detrás y en diagonal del gemelo virtual	47
Tabla 29. Medias recogidas de las pruebas realizadas para llegar a un punto detrás y en diagonal del gemelo virtual	47
Tabla 30. Datos recogidos en las pruebas realizadas de seguir una línea recta delante del robot utilizando el algoritmo de seguir una línea	48
Tabla 31. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot con el algoritmo de seguir una línea	48
Tabla 32. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del gemelo virtual utilizando el algoritmo de seguir una línea	49
Tabla 33. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot utilizando el algoritmo de seguir una línea	49
Tabla 34. Datos recogidos de las pruebas realizadas de seguir una línea recta delante del robot físico utilizando el algoritmo de seguir una línea y volver a ella	50
Tabla 35. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot físico utilizando el algoritmo de seguir una línea y volver a ella	50
Tabla 36. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea	52
Tabla 37. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea	52
Tabla 38. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al gemelo virtual utilizando el algoritmo de seguir una línea	53
Tabla 39. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al gemelo virtual utilizando el algoritmo de seguir una línea	54

Tabla 40. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea volviendo a ella	54
Tabla 41. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea volviendo a ella	55
Tabla 42. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea.....	56
Tabla 43. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea.....	57
Tabla 44. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea	57
Tabla 45. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea	58
Tabla 46. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea volviendo a ella	59
Tabla 47. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea volviendo a ella	59
Tabla 48. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea.....	60
Tabla 49. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea.....	61
Tabla 50. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea.....	61
Tabla 51. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea.....	62
Tabla 52. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella	62
Tabla 53. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella	63
Tabla 54. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea	64
Tabla 55. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea	64
Tabla 56. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea.....	65

Tabla 57. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea.....	65
Tabla 58. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea volviendo a ella.....	66
Tabla 59. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea volviendo a ella.....	66

Índice de fórmulas

Fórmula 1. Cálculo de la orientación	17
Fórmula 2. Píxeles por 0.1s	19
Fórmula 3. Ecuación general de la recta	24
Fórmula 4. Ecuación punto-pendiente de la recta	24
Fórmula 5. Variables de la ecuación punto-pendiente de la recta en la ecuación general	24
Fórmula 6. Distancia de un punto a una recta	25
Fórmula 7. Distancia de un punto a una recta con las variables sustituidas	25
Fórmula 8. Variación de orientación	25
Fórmula 9. Cálculo de la distancia desde la posición del robot al punto objetivo	26
Fórmula 10. Cálculo del tiempo que se debe mover el robot en función de la distancia que tiene al objetivo	26
Fórmula 11. Error entre la orientación del robot y el ángulo al punto objetivo	27
Fórmula 12. Cálculo del tiempo que se debe mover el robot en función de los grados de error con el objetivo	27
Fórmula 13. Cálculo de la orientación correcta para el robot siguiendo una línea	29
Fórmula 14. Cálculo del factor de ponderación	29
Fórmula 15. Cálculo del ángulo para saber si el robot se aleja o se acerca a la línea	33

1.- Introducción

La visión artificial es una disciplina científica que engloba métodos para adquirir, procesar, analizar y comprender imágenes del mundo real, con el objetivo de producir información que pueda ser procesada por un ordenador. De manera similar a cómo los seres humanos utilizamos nuestros ojos y cerebros para interpretar el entorno, la visión artificial busca replicar este efecto, permitiendo que los ordenadores perciban y comprendan imágenes para así actuar en consecuencia en diferentes situaciones. Esta comprensión se logra mediante la aplicación de principios de campos como la geometría, la estadística y la física.

La finalidad de la visión artificial es desarrollar estrategias automáticas para el reconocimiento de patrones en imágenes. Uno de los ámbitos que más usa la visión artificial actualmente es la robótica, ya que los robots con cierta autonomía necesitan ser capaces de reconocer su entorno para tomar decisiones.

1.1- Motivación

La principal utilidad de este trabajo radica en que cualquier persona de manera intuitiva, sea capaz de controlar elementos teledirigidos gracias al uso de la visión artificial. Además, se busca desarrollar algoritmos eficientes permitan a estos dispositivos realizar tareas de forma autónoma. Estos algoritmos se basan en la visión global de todo el entorno donde se va a mover el dispositivo, incluyéndolo a él.

Este sistema puede ser aplicado en varios escenarios. Por ejemplo, en un futuro se podría controlar un aparcamiento de coches, detectando los huecos libres y mandando instrucciones a un coche para que consiguiese aparcar de manera eficiente y sin usar conductor.

En el mantenimiento de jardines, el sistema podría dirigir un cortacésped para que siguiera una ruta preestablecida por un jardín adaptándose a los obstáculos marcados por el usuario.

En cuanto a las competiciones de robots, se abren nuevas posibilidades en cuánto a las competiciones de robots ya que este sistema al ser capaz de detectar tanto el robot como un entorno. En competiciones como carreras, el sistema es capaz de detectar el robot, el camino y el resto de los robots competidores para así ser capaz de trazar la mejor ruta posible para llegar finalizar la carrera primero.

Para competiciones estilo fútbol se puede identificar al robot propio, el robot contrario, el campo y la pelota. A partir de esta información se mandan instrucciones para intentar marcar gol en la portería a la vez que se esquiva el robot rival.

Este sistema cuenta con numerosas ventajas en comparación con los sistemas tradicionales de control de elementos teledirigidos. En primer lugar, cuenta con gran facilidad de uso, ya que tiene una interfaz intuitiva basada en la visión artificial. Esto permite que cualquier usuario pueda manejar el elemento teledirigido sin tener

conocimientos técnicos sobre él. Además, este sistema es preciso ya que la visión artificial proporciona información exacta sobre el entorno, lo que facilita un control eficaz del dispositivo.

Otra ventaja es la adaptabilidad del sistema, el cual puede ajustarse en diferentes entornos con sólo cambiar la cámara. El sistema tiene un gran potencial, ya que abre un amplio abanico de posibilidades para el desarrollo de nuevos servicios relacionado con la robótica y automatización.

1.2- Objetivos

Este trabajo tiene como objetivo principal desarrollar una plataforma autónoma que sea capaz de realizar tareas con el uso de visión artificial y un mando a distancia. El primer requerimiento para lograrlo es establecer conexión entre el mando del robot y el ordenador, desarrollando un software que permita mandar las instrucciones al robot. El programa desarrollado debe detectar con precisión la posición del robot en el mapa de píxeles y ser capaz de obtener su orientación tras realizar un movimiento.

Se quiere implementar la capacidad del usuario de realizar ciertas superposiciones en la imagen que sirvan de utilidad que son: una o varias regiones de interés, un punto cualquiera en la imagen y una trayectoria recta o con curvas en cualquier parte de la imagen.

Una vez implementadas estas superposiciones se debe desarrollar otro software capaz de detectar que el robot está dentro o fuera de las regiones de interés. También se pretende diseñar y desarrollar un algoritmo que consiga que el robot llegue a un punto cualquiera marcado por el usuario. Y por último realizar otro algoritmo que permita que el robot siga una trayectoria marcada por el usuario sin que el robot se pare.

El fin de los algoritmos es que el robot llegue a cualquier punto de la imagen ya sea delante suya, detrás o en diagonal. Y en cuanto a la trayectoria el robot tiene que ser capaz de seguir una línea recta, una línea con una curva y una línea con dos curvas.

1.3- Estructura del documento

El documento está organizado en diferentes apartados. Comienza con el estudio de alternativas en el que se describen diferentes métodos de otros trabajos que podrían ser aplicables a este.

Se continua con el apartado de diseño. En este apartado se define todo el sistema utilizado y sus diferentes partes. Después se pasa al apartado de implementación en el que se profundiza todo lo hablado en diseño, y se describe como se han desarrollado todos los procesos y algoritmos utilizados.

El documento sigue con los experimentos. En este apartado se recogen datos de las diferentes pruebas realizadas para evaluar los algoritmos desarrollados anteriormente.

Para acabar se explican las conclusiones del proyecto y se proponen trabajos futuros teniendo en cuenta los errores que se han cometido desarrollando este.

2.- Estudio de alternativas

Para la realización de este trabajo, se han investigado diferentes proyectos que tienen partes similares. Se ha investigados sobre las conexiones de la cámara, el robot y de cómo se envían las instrucciones. También se ha explorado diferentes métodos para detectar el robot y cómo se han diseñado algunos algoritmos de movimientos para seguir líneas.

2.1 Conexiones

En [1] se usa un microcontrolador para recibir las instrucciones del ordenador y activar los motores del robot, que es algo que se puede utilizar en este proyecto para transmitir las instrucciones del ordenador al robot. El diseño de las conexiones es el de la **Figura 1** en el que se observa que primero se enciende el circuito usado por el microcontrolador para que lo detecte el ordenador. Una vez detectado se establece la conexión con un programa de Matlab y tras este paso, se envían las instrucciones al microcontrolador por USB.



Figura 1. Diseño de las comunicaciones en Robot Móvil Autónomo Seguidor Delínea

En otros proyectos se usa una placa Arduino directamente en el robot como se puede ver en [2]. Se envían los datos directamente a un módulo bluetooth implementado en la placa Arduino.

En [3] se usa una Raspberry Pi, un microordenador, para extraer las imágenes de la cámara, y así poder identificar la línea y transmitir los datos a un Arduino Nano. Este Arduino Nano es una tarjeta de desarrollo que está conectada mediante USB a la Raspberry Pi. Es la encargada de tomar las decisiones de movimiento del robot y activar sus motores. La cámara se conecta a través de un conector CSI que tiene la Raspberry Pi. Tal y como se muestra en la **Figura 2** el Arduino Nano está conectado mediante un Puente H a los motores. El puente H es un circuito que permite a los motores rotar en ambos sentidos. Usar el USB para conectarse al microordenador que toma las decisiones para posteriormente enviarlas a la tarjeta de desarrollo es una forma muy útil de tener todos los elementos conectados entre sí y es posible usarlo en el trabajo que se quiere desarrollar.

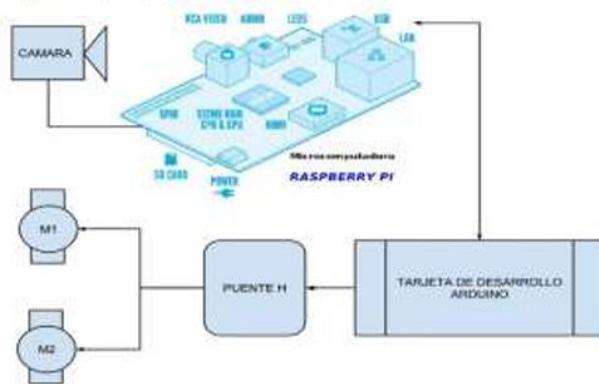


Figura 2. Diagrama de diseño de Robot Móvil Autónomo Seguidor de Línea con Raspberry Pi y Sistema de Visión

2.2 Detección del robot

Este proyecto trata sobre la detección de las posiciones de robots jugadores de fútbol. En [4] se usa la visión artificial para identificar a la pelota y a los jugadores, para conocer su posición en la cancha. Los robots son como se muestra en la **Figura 3**, llevan dos parches para poder identificarlos con la forma y color de estos parches. La orientación de los robots se determina con la posición relativa de estos parches. El uso de parches de colores es un método sencillo para que el robot se pueda identificar con facilidad en el trabajo que se desarrollará.

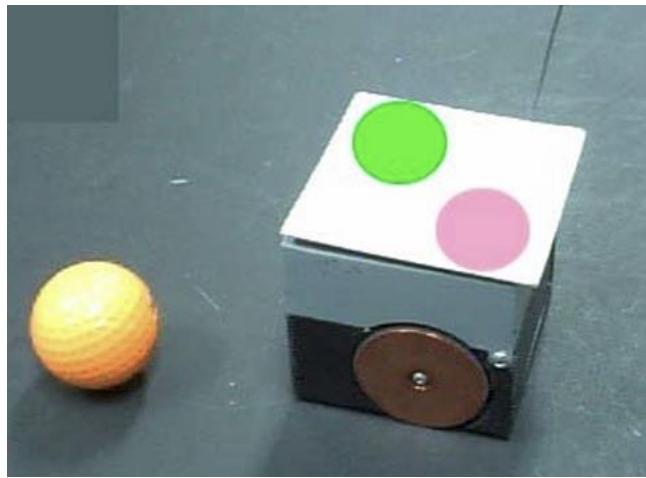


Figura 3. Robot jugador de fútbol y pelota

En [5] se utiliza la visión artificial únicamente para la detección de la línea, ya que la cámara está situada sobre el robot. Se emplea la librería OpenCV para capturar la imagen y posteriormente procesarla, compensando los errores relativos al brillo y al contraste. Para detectar la línea, se aplica umbralización, un proceso en el que los píxeles de la imagen se transforman según un cierto umbral (un color). Los píxeles que superan el umbral se establecen en 1, mientras que los que no lo superan se establecen en 0. De esta manera, en la imagen resultante, quedarán todos los píxeles de interés a 1 y el resto a 0, permitiendo descartar los píxeles no relevantes. Este método de umbralización es bastante usado para la detección de objetos y formas y también se usará en el proyecto a desarrollar para detectar el robot.

En [6] la imagen se convierte a HSV, esta transformación permite abordar todo tipo de cambios de iluminación. Tras esta transformación al igual que en el anterior proyecto se vuelve a aplicar el proceso de umbralización.

En [7], la orientación del robot se detecta de nuevo tras realizar el proceso de umbralización, calculando la posición del centro de las ruedas y del centro del robot. Esto permite calcular el punto medio entre las ruedas y, al conocer la posición del centro del robot, se obtiene la dirección de este, como se muestra en la **Figura 4**.

Únicamente calculando el punto medio de las dos ruedas no se podría determinar el sentido del robot, ya que podría ir en ambos sentidos. Sin embargo, al obtener la posición del centro del robot, se puede determinar que el sentido es el opuesto al de la ubicación de este centro, respecto al centro de las dos ruedas.

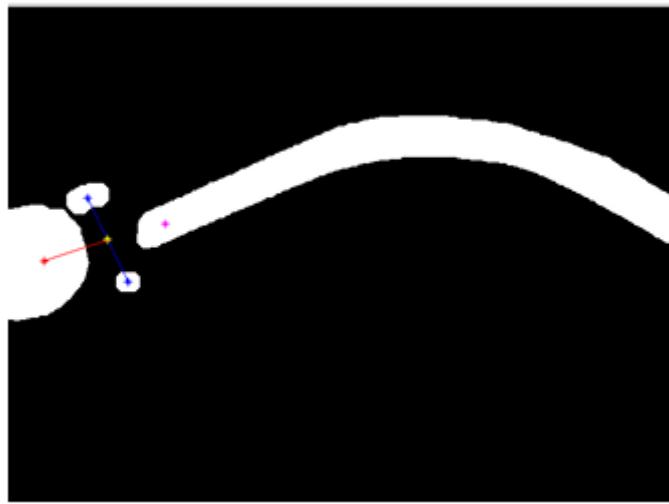


Figura 4. Detección de la dirección del robot en base al centro de las ruedas y el centro del robot

2.3 Algoritmos de movimiento

En [8] se dibuja a mano una línea que representa el recorrido que debe trazar el robot. Para que el robot realice este trazo, la línea se descompone en segmentos y, en función de la diferencia entre el punto final e inicial de cada segmento, se calcula la instrucción correspondiente que el robot debe seguir en ese momento para mantener el trazo, ya sea avanzar, retroceder, girar a la derecha o girar a la izquierda. Dibujar a mano la línea es una idea muy buena para implementar que el usuario describa la línea que quiere seguir el robot con una interfaz sencilla, por lo que se usará en el trabajo a desarrollar.

En [9] se estudian distintos tipos de métodos para seguir una línea dibujada físicamente con un robot de dos ruedas con un motor en cada una. Es en el propio robot donde están los sensores para detectar la línea.

En el primer método, el robot tiene un solo sensor de luz para detectar si está sobre la línea. Si detecta negro, es línea y si detecta blanco no lo es. Si detecta la línea gira a la izquierda, y si no, gira a la derecha, avanzando en zigzag como se ve en la **Figura 5**.

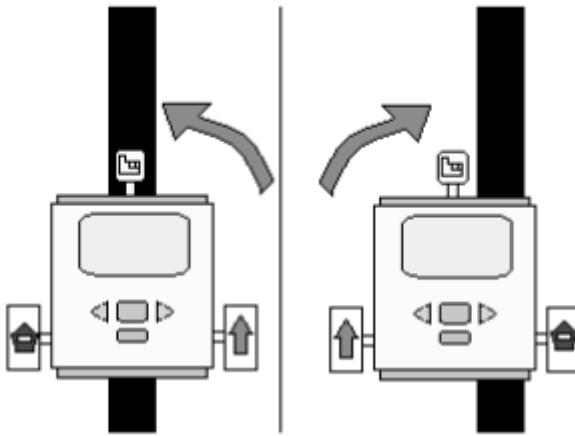


Figura 5. Método de seguir línea con un sensor de luz sobre el robot

Con dos sensores gira a la izquierda cuando el sensor de la derecha no detecta línea y el de la izquierda sí, y gira a la derecha cuando el sensor de la izquierda no detecta línea y el de la derecha sí. Avanza en zigzag de igual manera.

El siguiente método es con sólo un sensor capaz de detectar sobre qué color tiene más superficie el robot. Si el robot tiene más superficie sobre blanco, el robot gira hacia la derecha. Por el contrario, si detecta que tiene más superficie sobre el negro, gira a la izquierda. Estos tres métodos son muy interesantes para seguir una línea cuando el robot está sobre ella y se tendrán en cuenta en las implementaciones del algoritmo.

[10] es el más parecido al que se quiere realizar ya que la cámara es capaz de ver tanto la línea como el robot, no como el anterior trabajo que el sensor estaba sobre el robot. Como se ha explicado en el apartado 2.2, en este trabajo la dirección del robot se determina en base al centro del robot y el centro entre las dos ruedas. Para calcular la instrucción que se tiene que mandar al robot, primero se localiza un punto de la trayectoria haciendo un recorrido de toda la imagen para que ese punto sea el siguiente al que tiene que ir el robot. Con la diferencia entre la orientación del robot y el ángulo del centro de las ruedas al punto detectado, se envían las instrucciones. Dependiendo de ciertos umbrales el robot avanza, corrige su dirección a la derecha, o corrige su dirección a la izquierda.

La principal diferencia respecto a uno de los objetivos que tiene este trabajo es que el robot se detiene tras cada instrucción esperando recibir nuevas instrucciones, en vez de no detenerse. Pero que las instrucciones se tomen en función de la posición del robot respecto al siguiente punto de la línea al que tiene que ir, se implementará en el algoritmo de seguir una línea que se desarrollará.

3. Diseño

Como se puede ver en la **Figura 6** el diseño del sistema debe tener en cuenta varios elementos. La cámara para capturar imágenes, el ordenador para enviar instrucciones, el GPIO para hacer de intermediario entre el ordenador y el mando del robot, el mando del robot para hacerle llegar las instrucciones al robot y el robot.

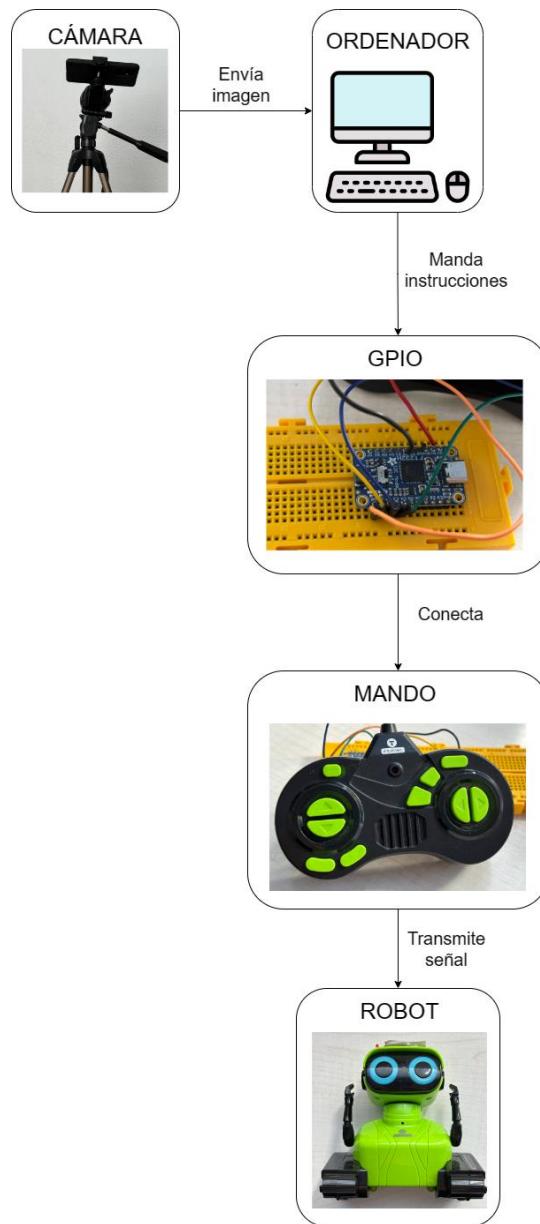


Figura 6. Diagrama del diseño del sistema

El diseño de este proyecto se basa principalmente en dos partes, la comunicación y los desplazamientos del robot. Estas dos partes tienen en común la visión, ya que se establecerá una comunicación entre la cámara y el ordenador para que las imágenes capturadas sean analizadas y se calculen instrucciones de movimiento en función de ellas. Estas instrucciones serán ejecutadas por el robot gracias a realizar otra comunicación entre el ordenador y el robot.

Para que las conexiones estén funcionen correctamente, se usarán librerías de Python que permitan mantener todas las partes bien conectadas. En la **Figura 7** se muestra el diagrama de clases utilizado. Habrá una clase principal que incluirá un listener, que estará pendiente de las teclas del teclado que se accionan para activar funciones de la clase GPIO, la cual controla el movimiento del robot. Además, en la clase principal hay otro objeto “vision”, que capturará las imágenes y realizará todos los cálculos que tengan que ver con ellas como por ejemplo la distancia entre dos puntos, la orientación del robot...

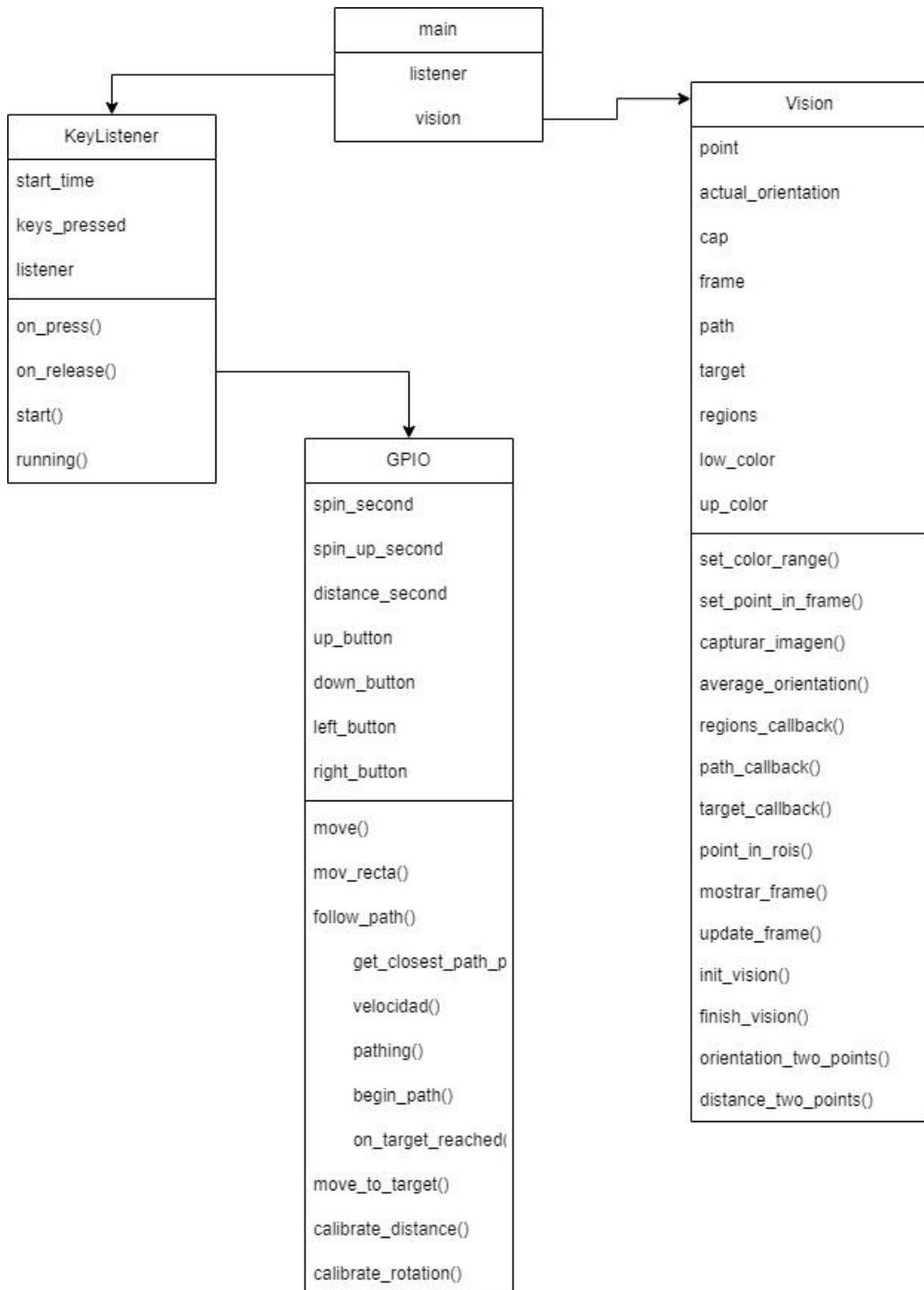


Figura 7. Diagrama de clases del código utilizado

3.1 Diseño de la comunicación

Dentro del diseño de la comunicación se pueden diferenciar dos partes que se explican a continuación. La comunicación de la cámara con el ordenador y la comunicación del ordenador con el robot.

3.1.2 Comunicación de la cámara con el robot

Como se puede ver en la **Figura 8** la cámara utilizada es la de un teléfono móvil modelo OnePlus 9.



Figura 8. Cámara OnePlus 9

Las especificaciones de esta son:

- Sensor principal: Sony IMX689 48 MP (1.12 µm), EIS, f/1.8
- Ultra angular: Sony IMX766 50 MP, f/2.2
- Lente monocromo: 2 Megapíxeles
- Otros: flash LED Dual, PDAF + CAF
- Vídeo: 8K 30fps, cámara lenta 480fps (HD), 240 fps (FHD)

El envío y recepción de imágenes de la cámara y el ordenador se realiza vía wifi. Se usa la aplicación llamada DroidCam [11] que mediante wifi o conexión por cable permite que la cámara de un teléfono móvil se use en un ordenador. En este caso se usa mediante wifi y por el puerto 4444, que es el que viene por defecto en la aplicación. Para usarlo se debe tener descargada la aplicación tanto en el teléfono móvil como el cliente en el ordenador.

3.1.2 Comunicación del ordenador con el robot

Para conseguir comunicar el ordenador con el robot mediante el mando que viene incluido con él, se ha tenido que usar un GPIO que se use como intermediario, concretamente el Adafruit FT232H [12] que se aprecia en la **Figura 9**.

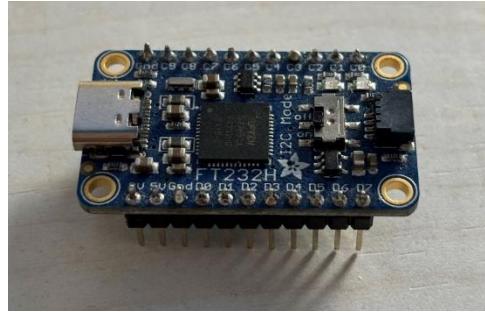


Figura 9. Adafruit FT232H

Este GPIO se conecta al ordenador mediante un cable USB a USB-C. Posteriormente, mediante el uso de una protoboard y cables adicionales se conectan los pines del GPIO a los botones del mando. Para asegurar las conexiones, se han soldado los cables a la placa del mando en los puntos donde están los botones tal y como se demuestra en la **Figura 10**.

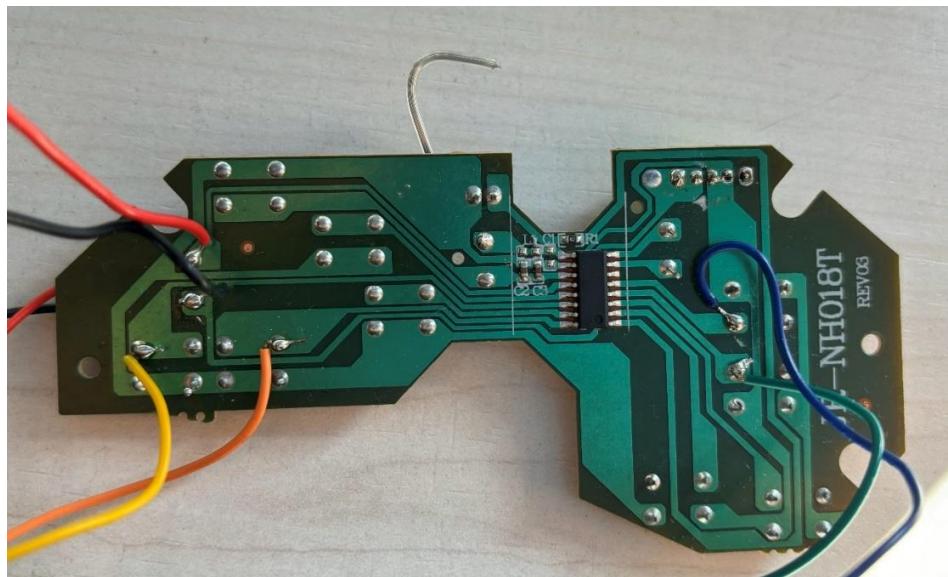


Figura 10. Placa del mando del robot

Gracias a un multímetro se ha podido comprobar que cuando se pulsan los botones, este marca 0 voltios, lo que significa que los botones se activan a bajo nivel. Por esto cuando se programe el código para activarlos, los botones se activarán a False.

En la **Figura 11** se muestra una transparencia en la que aparece la placa del mando y sus botones que están por el lado contrario. Y en la **Figura 12** se pueden ver los botones directamente.

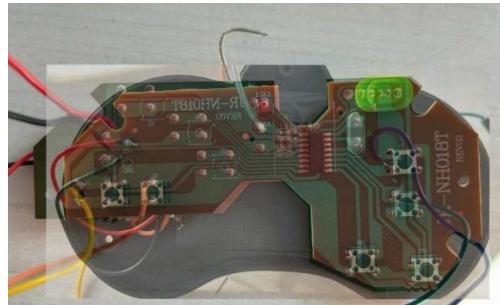


Figura 11. Transparencia de la placa del mando con los botones de la placa por el lado contrario



Figura 12. Lado contrario de la placa del mando

De la placa del mando salen 6 cables, cada uno con su función específica. El negro está destinado a la toma de tierra y el cable rojo (de alimentación) está conectado al pin de 3 voltios del GPIO que sustituye las 2 pilas AA que necesita el mando para funcionar. También podemos observar en la **Figura 13** otros 4 cables (naranja, amarillo, azul y verde) que se conectan a los pines C0, C1, C2 y C3 del GPIO. Estos son los cables que permiten mandar las instrucciones a cada botón del mando (arriba, abajo, izquierda y derecha).

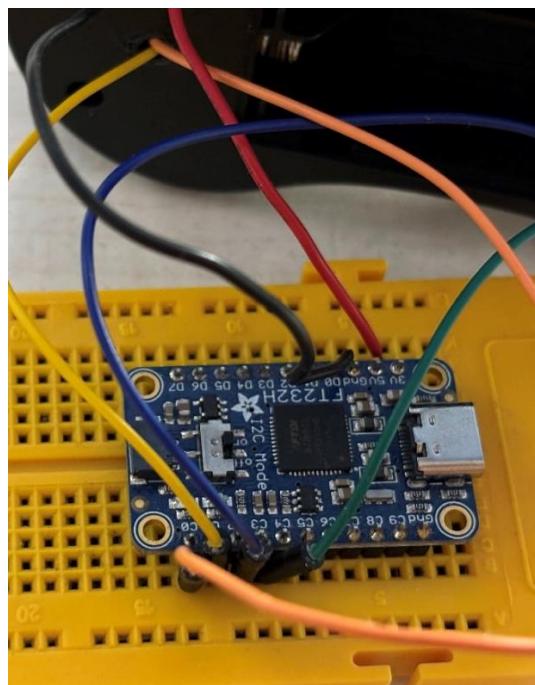


Figura 13. Conexiones de cables con GPIO

En la **Figura 14** se observa la apariencia del mando utilizado y los botones necesarios para su movimiento, que son aquellos que tienen flechas dibujadas.



Figura 14. Mando del robot

Una vez conectado el ordenador al GPIO y el GPIO al mando, ya se tiene conexión del ordenador al robot. El robot utilizado es “Remote Control Robot 2.4GHz RC Robot Toy for Kids – GILOBABY” [13] que aparece en la **Figura 15**.



Figura 15. Remote Control Robot 2.4GHz RC Robot Toy for Kids – GILOBABY

El robot cuenta con dos motores, uno a cada lado, que accionan una rueda cada uno. Además, el robot cuenta con otras 4 ruedas, dos a cada lado, pero que no tienen ninguna conexión con los motores. Estas 6 ruedas se observan en la **Figura 16**.

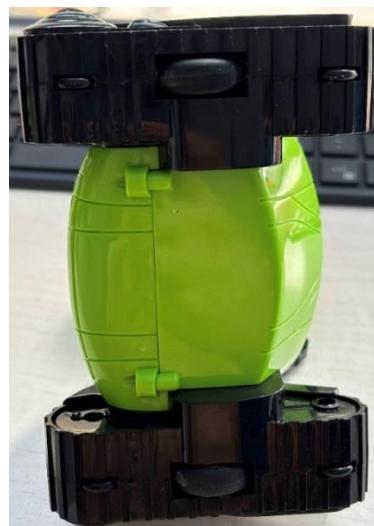


Figura 16. Ruedas del robot

3.2 Diseño de la movilidad

Dentro de los objetivos del proyecto se pueden diferenciar dos grandes aspectos, el desplazamiento del robot a un punto cualquiera y el desplazamiento que tiene que hacer el robot para seguir una línea. Pero el robot también se debe poder mover manualmente por si se necesita en cualquier circunstancia.

Para diseñar los movimientos posibles del robot hay que tener en cuenta que el robot posee distintos tipos de movimientos. Existen los movimientos de avance y retroceso que se ejecutan cuando se pulsan los botones de arriba y abajo respectivamente.

El robot también dispone de varios tipos de giros, cuando simplemente se pulsa izquierda o derecha, uno de los motores hace que una rueda avance y otra retroceda lo que hace que el robot gire sobre sí mismo. Y el otro tipo de giro es cuando se pulsa izquierda o derecha combinándolo con arriba o abajo. Lo que ocurre en este caso es que solo se acciona uno de los motores haciendo que solo una rueda se mueva avanzando o retrocediendo, en este caso el robot gira, siendo el centro de giro la rueda que no se acciona.

Como el robot se mueve mientras los botones del mando son pulsados, para controlarlo manualmente por el ordenador se usarán las flechas del teclado que serán registradas mediante un KeyListener detallado más adelante en el apartado 4.6. Y para que se pueda mover más o menos tiempo según el usuario desee, se registrará cuanto tiempo son pulsadas estas teclas y si varias teclas están siendo pulsadas a la vez. En la **Figura 17** se puede ver las dos maneras con las que el robot se va a poder mover, la primera que acabamos de describir y la segunda que sucederá con los algoritmos descritos en los siguientes apartados.

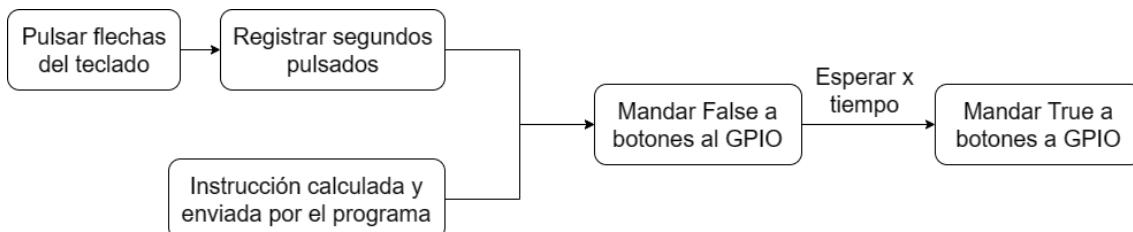


Figura 17. Diseño de movimiento del Robot

3.2.1 Diseño de algoritmo para llegar a un punto cualquiera

Para conseguir que el robot llegue desde un punto inicial hasta un punto cualquiera se necesita obtener distintos datos. En primer lugar, la orientación actual que lleva el robot, también el ángulo del punto donde está con respecto al punto objetivo, y la distancia hacia el punto objetivo.

Para obtener la orientación actual del robot es necesario obtener la posición del robot tras cada movimiento, para así calcular la orientación con este punto y el anterior.

El cálculo del ángulo al punto objetivo y la distancia a él se hace de manera similar, se realiza con el punto donde está ubicado el robot y el punto objetivo.

Para que el robot consiga llegar al punto cada vez que se mueve hay que repetir el cálculo de todos los datos, porque como el robot no es preciso en sus movimientos, puede variar la posición y orientación final a la que se quería llegar con el cálculo del anterior movimiento.

El robot no se mueve continuamente, cada vez que se manda una instrucción esta acaba y el robot para, si este no está dentro de un umbral de distancia al punto objetivo, como se ve en la **Figura 18**, se vuelve a ejecutar todo el proceso otra vez, hasta que está lo suficientemente cerca del punto.

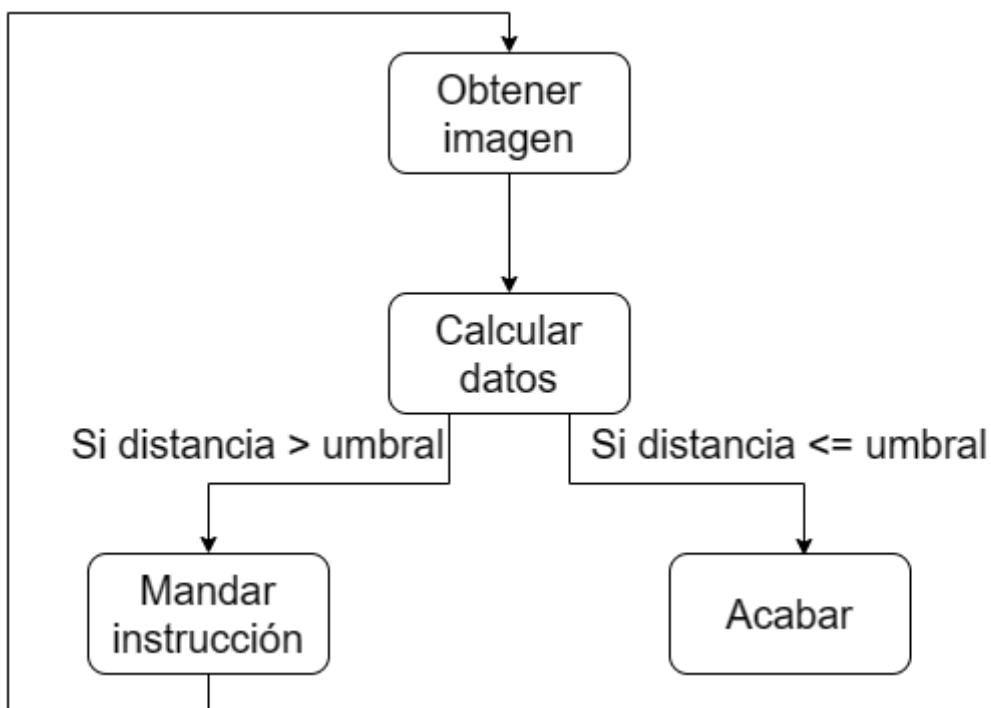


Figura 18. Diseño algoritmo llegar a un punto cualquiera

3.2.2 Diseño del algoritmo seguir una línea

La principal diferencia con el anterior diseño es que en este caso el robot no para después de cada instrucción, sino que el robot se mueve continuamente y el programa tiene que ir ajustando sus movimientos mientras se está moviendo.

El primer paso es que el robot llegue al punto inicial de la línea y se oriente con el mismo ángulo que esta. Para llegar al punto inicial se usará el anterior algoritmo y para orientarlo con la línea se harán pequeñas correcciones con izquierda y derecha hasta que el robot esté orientado correctamente con un pequeño umbral de error.

Una vez llegado al punto inicial y orientado, el robot empieza a moverse y el programa calcula las instrucciones que se le tienen que mandar continuamente.

Para conseguirlo se han tenido en cuenta los siguientes parámetros:

- Distancia del robot al punto más cercano

- Orientación del robot
- Ángulo del robot al punto más cercano
- Ángulo de la línea en el punto más cercano

En base a estos parámetros se ha obtenido una función detallada en el apartado 4.8 que calcula la orientación que debería llevar el robot haciendo una media ponderada entre el ángulo de la línea y el ángulo del robot con el punto más cercano de la línea. Y el peso que se le da a cada ángulo varía en función de la distancia entre el robot y el punto más cercano de la línea.

Todos estos valores se calculan cada vez que pasa cierto tiempo para que el movimiento del robot se ajuste continuamente y como se puede ver en la **Figura 19** se ejecuta en bucle hasta llegar al final de la línea.

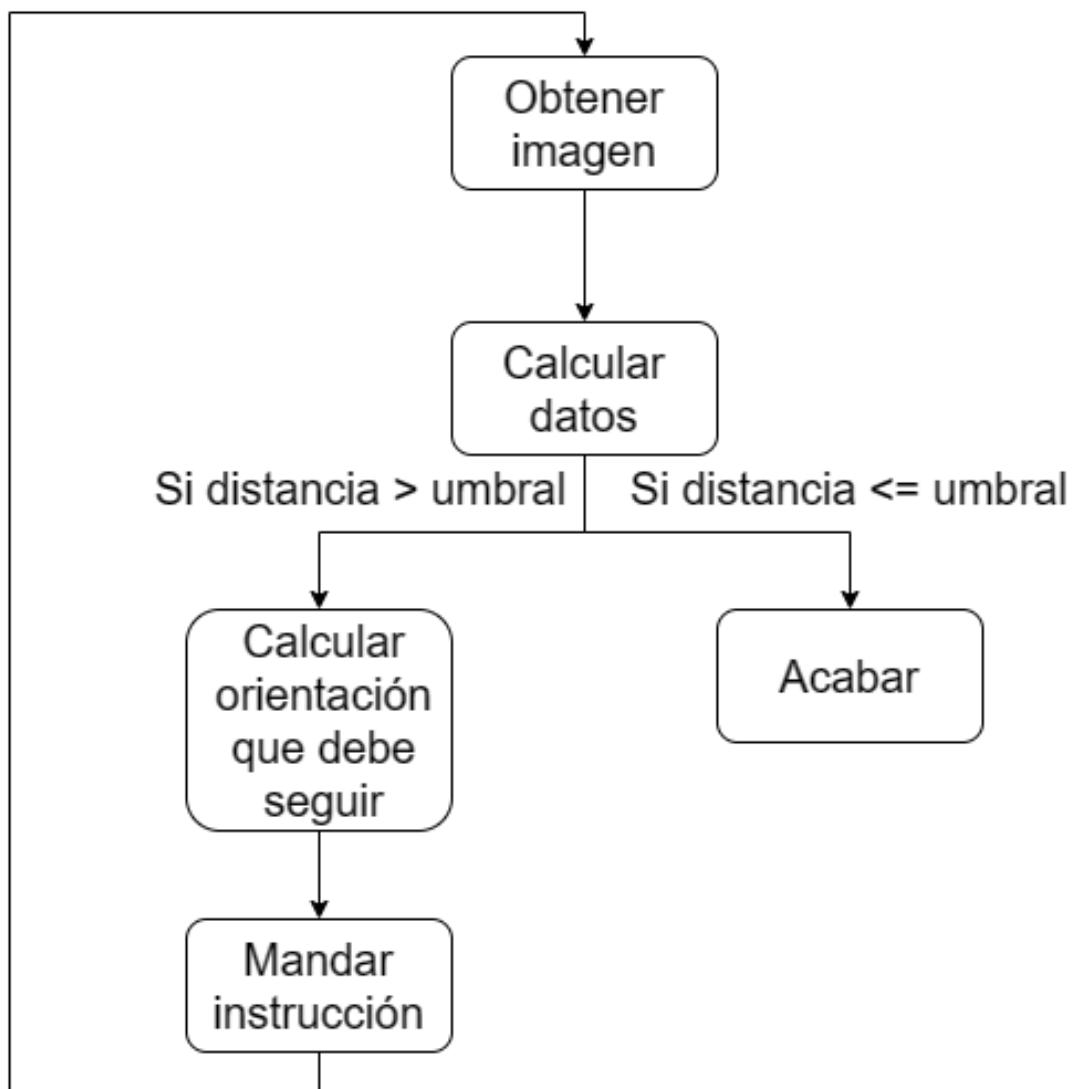


Figura 19. Diseño algoritmo seguir línea

4. Implementación

Tras explicar el diseño, se describirá como se ha implementado. Tanto los apartados más básicos como detectar el robot y su orientación como también los algoritmos de llegar a un punto y seguir una línea. También se describirán las soluciones planteadas a problemas que se han encontrado a la hora de desarrollar el algoritmo de seguir una línea.

4.1 Movimiento del Robot

Para poder accionar los motores del robot desde el ordenador se deben mandar instrucciones al GPIO para que se activen los pines correspondientes a los botones del mando que hacen que se muevan las ruedas del robot.

Primero se tiene que conectar correctamente el GPIO al ordenador y para ello se sigue el tutorial que proporciona el GPIO usado [14]. Tras instalar algunas librerías y crear una variable de entorno necesaria para que lleguen las instrucciones ya se tiene conexión con los motores del robot.

Para accionar los motores se importa la librería llamada “digitalio” que permite configurar y controlar los pines digitales en término de entrada y salida [15]. Guardando los pines que están conectados a los botones en variables ya se pueden activar y desactivar los motores a True y a False.

4.1 Detección del robot

Para usar la imagen de la cámara en el programa se va a usar la librería OpenCV [16] la cual permite entre otras cosas, capturar frames de la cámara y modificarlos para obtener información que se necesite o también exponer ciertos detalles al usuario. Las funciones utilizadas más importantes han sido: VideoCapture() para conectar la cámara al programa y read() para obtener los frames.

Para detectar al robot, el programa lo primero que hace es pedir que se seleccione un punto de la imagen donde este el robot, para así poder guardar el color de este.

Este color se ha guardado en RGB (rojo, verde, azul) y se transforma a HSV (Hue, Saturation, Value) porque este formato es más adecuado para manejar variaciones de iluminación. Se guarda un margen por arriba y por abajo del color elegido ya que en el entorno puede haber distintas iluminaciones que hagan que la cámara perciba distinto el color.

Una vez se tiene la imagen en HSV y se tiene el rango de colores que es de interés se utiliza una técnica conocida como binarización o umbralización para que los píxeles de interés se queden a 1 y el resto a 0. Esto se realiza aplicando una máscara sobre la imagen original con los márgenes obtenidos anteriormente que da como resultado una imagen similar a la **Figura 20**, en la que los píxeles de interés quedan en blanco y el resto de la imagen en negro.

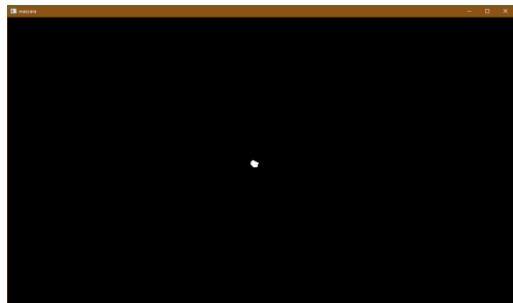


Figura 20. Máscara aplicada para detectar área del color seleccionado

Después de obtener los píxeles de interés tras haber seleccionado un color, se obtiene el área de este color. Se descartan áreas muy pequeñas porque pueden ser errores de detección y se calcula el centro del área, que será el píxel donde se ubique al robot quedando situado como en la **Figura 21**.



Figura 21. Imagen del robot detectado por el programa

4.2 Cálculo de la orientación del robot

El cálculo de la orientación del robot se lleva a cabo guardando el último punto donde ha estado el robot en una variable y guardando la posición actual en la que está el robot cuando se refresca el frame tras realizar un movimiento. Con la diferencia entre ambos puntos se puede calcular la orientación del robot respecto al eje de coordenadas con la **Fórmula 1**:

$$\tan^{-1} \frac{\Delta y}{\Delta x}$$

Fórmula 1. Cálculo de la orientación

Esta función devuelve el resultado en radianes y posteriormente se convierten a grados. Pero para esto primero hay que tener en cuenta que el frame de OpenCV tiene la matriz de píxeles empezando por el (0, 0) arriba a la izquierda y creciendo las coordenadas x hacia la derecha y las coordenadas y hacia abajo. Lo que hace que los ángulos sean complicados de entender porque serían como se observan en la **Figura 22**.

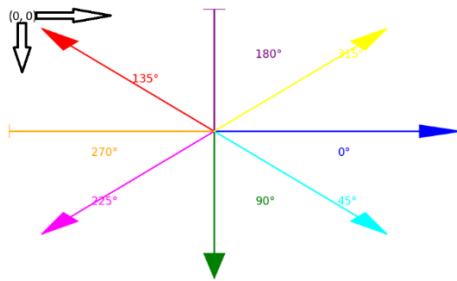


Figura 22. Ángulos en un sistema de coordenadas en el que la coordenada y crece hacia abajo

Por lo que para poder trabajar en un sistema de coordenadas común en el que la coordenada x crece hacia la derecha y la coordenada y crece hacia arriba, cada vez que se registra un punto en el sistema se tiene que transformar la coordenada y. Para realizar este cambio simplemente se resta la altura total de la imagen menos la coordenada y del punto que se quiere transformar. Con este cambio los ángulos quedan como se ve en la **Figura 23**, que son equivalentes a los que hay en un sistema de coordenadas común.

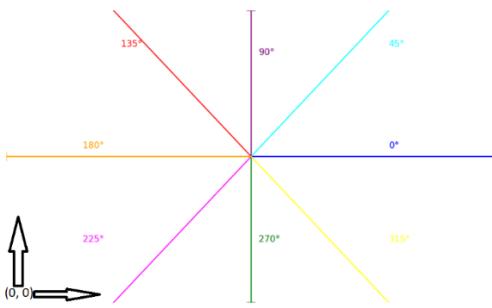


Figura 23. Ángulos en un sistema de coordenadas común

Una vez cambiadas las coordenadas y ya se puede obtener la orientación con la fórmula mencionada anteriormente y se puede poner textualmente el ángulo en la imagen acompañado de una flecha para que se perciba la orientación del robot a simple vista, quedando plasmado en la imagen como en la **Figura 24**.



Figura 24. Imagen del robot después de calcular su orientación

Es importante saber que la orientación del robot sólo se va a calcular cuando avanza o retrocede, ya sea sólo avanzando y retrocediendo o también con las combinaciones de giro como por ejemplo avanzar y derecha. Esto es porque cuando el robot gira sobre sí mismo al no haber ningún cambio entre el punto registrado antes de moverse y el punto tras realizar un movimiento no se puede calcular la nueva orientación por lo que, si se quiere saber la orientación después de girar el robot sobre sí mismo, este tiene que avanzar o retroceder para poder obtener la orientación con la fórmula mencionada anteriormente.

4.3 Calibración de distancia

Para saber cómo y cuánto se mueve el robot primero se debe calibrar la distancia que avanza cada vez que se acciona el botón de avanzar o retroceder. Debido a que la imagen está en píxeles y normalmente se trabajará con pulsaciones cortas en torno a 0.1 segundos, las mediciones obtenidas estarán en píxeles/0.1s.

La calibración de la distancia se debe realizar cada vez que se cambie la cámara de lugar, porque como la medida de la distancia está en píxeles, si la cámara está más lejos se moverá menos píxeles y viceversa.

Otro factor que se podría tener en cuenta es el de la perspectiva, ya que no es lo mismo que el robot este justo debajo de la cámara que este más lejos. En las pruebas realizadas durante todo el trabajo, al realizarse en un espacio relativamente pequeño y la cámara estar enfocada justo hacia abajo no se ha tenido en cuenta esto.

Para la calibración se ha programado una función que hace que el robot avance y retroceda alternadamente para que el robot no se salga de la imagen mientras que se recogen datos de los distintos desplazamientos. En cada movimiento guarda los píxeles desplazados y el tiempo que se ha movido y al final saca la media de píxeles por cada 0.1 segundos.

La **Fórmula 2** es utilizada para calcular cuantos píxeles se mueve por 0.1s es:

$$\frac{píxeles * 0.1}{t}$$

Fórmula 2. Píxeles por 0.1s

En la **Tabla 1** se pueden observar las distintas medias obtenidas como resultado de realizar 25 pruebas avanzando y retrocediendo con los respectivos tiempos pulsados. Por ejemplo, en la primera prueba para desplazamientos de 0.05 segundos, la media es que se mueve 56 píxeles por 0.1 segundos.

Tiempo pulsado	Distancia para 0.1s
0.05s	56 píxeles/0.1s
0.05s	51 píxeles/0.1s
0.1s	34 píxeles/0.1s
0.1s	29 píxeles/0.1s
0.1s	33 píxeles/0.1s
0.2s	43 píxeles/0.1s
0.2s	46 píxeles/0.1s
0.2s	46 píxeles/0.1s
0.3s	44 píxeles/0.1s
0.3s	41 píxeles/0.1s
0.3s	39 píxeles/0.1s
0.4s	39 píxeles/0.1s
0.4s	31 píxeles/0.1s
0.4s	37 píxeles/0.1s

Tabla 1. Medias de píxeles por 0.1s con distintos tiempos de pulsación

Dentro de cada prueba se aprecian movimientos muy distintos en cada avance y retroceso, lo que indica que a pesar de que las medias de la tabla 1 sean similares, los movimientos no son nada precisos y difieren mucho unos de otros. En la **Tabla 2** se presentan algunas cantidades de píxeles que el robot ha avanzado o retrocedido dentro de la puebla de 0.5s de tiempo pulsado.

Desplazamiento en 0.5s
20 píxeles
155 píxeles
14 píxeles
22 píxeles
5 píxeles
12 píxeles
...

Tabla 2. Desplazamiento de píxeles en una de las pruebas de 0.5s de tiempo pulsado

4.4 Calibración de giro

Para calibrar el giro se ha realizado un programa que manda un giro de un determinado tiempo al robot y luego le manda avanzar para poder saber la orientación final. Para obtener los grados girados en ese tiempo se resta la orientación final menos la inicial.

Al igual que en los desplazamientos se obtiene una medida en 0.1s que es un tiempo similar al de las instrucciones que le serán ordenadas al robot. En la **Tabla 3** se evidencian las medias obtenidas de realizar 25 giros por prueba con distintos tiempos de giro.

Tiempo pulsado	Media de grados girados para 0.1s
0.1s	49º/0.1s
0.1s	44º/0.1s
0.1s	46º/0.1s
0.2s	38º/0.1s
0.2s	42º/0.1s
0.2s	34º/0.1s
0.3s	37º/0.1s
0.3s	34º/0.1s
0.4s	34º/0.1s
0.4s	34º/0.1s
0.1s	38º/0.1s

Tabla 3. Medias de grados por 0.1s con distintos tiempos de pulsación

En la **Tabla 4** se presentan algunos de los giros que ha realizado en la prueba de 0.1s de tiempo pulsado. Tal y como sucede en los desplazamientos, en los giros también hay grandes diferencias de grados girados en diferentes giros.

Grados girados en 0.1s
61º
25º
32º
42º
46º

59°
25°
17°
28°
35°
...

Tabla 4. Grados girados en una de las pruebas de 0.1s de tiempo pulsado para el giro lento

Por lo que, aunque el resultado las medias de resultados parecidos, claramente no es preciso el robot. Además, se puede observar que a medida que se han ido haciendo pruebas, la media de grados girados ha bajado. Esto ocurre porque también afecta cuanta carga tengan las pilas. Cambiando las pilas y volviendo a ejecutar el código nos da nuevos resultados que se pueden ver en la **Tabla 5** con una media de giro más alta que las últimas pruebas realizadas, por lo que cada cierto tiempo habría que calibrar el giro.

Tiempo pulsado	Media Grados girados
0.1s	62°
0.1s	63°

Tabla 5. Nuevas medias obtenidas con nuevas pilas para el robot

Las anteriores medias se han recopilado para el tipo de giro en el que se combinan dos botones, arriba o abajo con izquierda o derecha. Se va a realizar la misma prueba para el otro tipo de giro, en el que sólo está pulsado el botón de izquierda o derecha. En la **Tabla 6** se muestran los resultados obtenidos de estas pruebas con distintos tiempos pulsados, realizando 25 giros en cada una.

Tiempo pulsado	Media Grados girados para 0.1s
0.05s	121°/ds
0.05s	103°/ds
0.05s	115°/ds
0.1s	69°/ds
0.1s	91°/ds
0.1s	85°/ds
0.2s	71°/ds
0.2s	76°/ds
0.2s	72°/ds
0.3s	70°/ds
0.3s	68°/ds
0.3s	68°/ds

Tabla 6. Medias de grados por 0.1s con distintos tiempos de pulsación para el giro rápido

Se vuelve a observar la diferencia de grados girados de un movimiento a otro en la **Tabla 7**. Este giro al ser más rápido que el otro, hay mayor diferencia de los grados girados entre un giro y otro.

Grados girados en 0.1s
201°
52°
42°
89°

153°
80°
92°
105°
124°
107°
...

Tabla 7. Grados girados en una de las pruebas de 0.1s de tiempo pulsado para el giro sobre sí mismo

4.5 Gemelo virtual

El gemelo virtual es un programa a parte del original para simular el robot como un punto y poder hacer distintas pruebas sin necesidad de usar el robot. Para que este simulado de la mejor forma posible es necesario implementarle las imprecisiones en los movimientos, tanto avanzando y retrocediendo como girando. De esta manera también se podrá comprobar que los algoritmos utilizados funcionan correctamente a pesar de que el robot sea impreciso, ya que es algo inherente en su mecánica y que siempre va a existir.

Para ello cada vez que el punto avanza una función random va a hacer que le punto se desvíe un poco cuando avanza o retrocede. Igual en los giros, como se ha podido observar que cada giro puede sufrir grandes variaciones, también se ha implementado una función random que hace que el punto al girar un determinado tiempo gire más o menos de lo calculado.

Además, como en el programa original solo se puede saber la orientación del robot cuando este ha avanzado, en el gemelo virtual es igual, aunque al realizar un giro el programa obtenga cual es la orientación final de robot, esta ni se muestra ni se usa para nada hasta que el punto ha avanzado o retrocedido.

4.6. Implementación de superposiciones gráficas

Para que el robot llegue a un punto, siga una línea, se detecte que está en una región prohibida o que aparezca su orientación en la imagen se ha tenido que implementar unas funciones que registran pulsaciones del ratón en el frame capturado, y que estas pulsaciones sirvan para poder hacer las superposiciones en la imagen que deseé el usuario.

Para realizar todas estas superposiciones se han usado en gran medida funciones de la librería OpenCV [17]. Para dibujarlas por encima de la imagen se ha usado: circle() para dibujar puntos, line() para las líneas, rectangle() para las regiones de interés y putText() y arrowedLine() para mostrar los ángulos.

Se ha programado un KeyListener que aparte de registrar las teclas de movimiento del robot, también registra que número del teclado se ha pulsado, ya que cada número sirve para poder realizar una marca distinta o activar algún algoritmo de desplazamiento.

- Punto Objetivo. Para marcar el punto objetivo al que el robot tiene que llegar, se activa la función que registra esta marca pulsando el número correspondiente en el teclado y después se pulsa con el ratón sobre la imagen. Este punto se guarda en una variable y se dibuja en el frame como se muestra en la **Figura 25**.

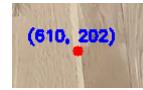


Figura 25. Ejemplo de punto objetivo superpuesto en la imagen

- Línea. Para guardar la línea y marcarla en la imagen se activa esta función con el teclado igual que en el punto y se hace clic izquierdo en la imagen para empezar a guardar el trazo del ratón, se vuelve a hacer clic izquierdo para acabar de trazar la línea. Se guardan todos los puntos por dónde ha pasado el ratón en una lista y con ella se dibuja la línea como se ve en la **Figura 26**.



Figura 26. Ejemplo de línea superpuesta en la imagen

- Ángulos en la imagen. Tanto para superponer la orientación del robot en la imagen como para superponer la orientación de la línea cuando el robot este siguiéndola se usa una función de OpenCV que dibuja una flecha en la imagen como se muestra en la **Figura 27**.



Figura 27. Ejemplo de ángulo superpuesto en la imagen

- Regiones. Para marcar las regiones se sigue el mismo procedimiento que para marcar la línea. La diferencia es que se guarda el punto donde se clica primero el ratón y también el punto donde se vuelve a clicar. Teniendo estos dos puntos se calcula todos los puntos que están en el área del rectángulo trazado y se guardan en una matriz. En la imagen se muestra los bordes de dicha región como se ve en la **Figura 28**. La detección del robot dentro de la región se consigue comprobando si el píxel del robot está dentro de la matriz generada y se muestra como en la **Figura 29**.



Figura 28. Ejemplo de región superpuesta en la imagen



Figura 29. Ejemplo de robot detectado en una región

4.7 Movimiento en línea recta del robot

El robot cuando avanza hacia delante o hacia atrás no consigue ir recto debido a algún fallo de hardware en el robot que hace que a medida que avanza vaya trazando una trayectoria curva hacia la derecha.

Por eso, para que el robot consiga hacer un movimiento en línea recta se ha tenido que programar una función que corrija este error. Esta función envía pequeños pulsos de activación al botón de izquierda del mando para conseguir corregir este desvío.

Se tiene que calibrar cuánto tiempo está pulsado el botón izquierdo y cuánto tiempo sin pulsar. En la **Tabla 8** se ven los resultados obtenidos de haber realizado pruebas con distintos tiempos pulsados y no pulsados. Los resultados son la distancia desviada del punto donde acaba el robot, respecto a la recta que debería haber trazado, y la variación de la orientación de como empieza a como acaba.

Tiempo Pulsado	Tiempo no Pulsado	Distancia desviada	Variación de la orientación
0.05s	0.05s	315 px	193º
0.05s	0.1s	423 px	129º
0.05s	0.2s	223 px	37º
0.05s	0.2s	316 px	20º
0.05s	0.2s	55 px	-2º
0.05s	0.2s	333 px	50º
0.05s	0.3s	147 px	17º
0.05s	0.3s	2 px	5º
0.05s	0.3s	306 px	12º
0.05s	0.3s	56 px	-2º
0.05s	0.3s	22 px	11º
0.1s	0.4s	465 px	100º
0.1s	0.4s	653 px	-281º

Tabla 8. Datos obtenidos para determinar cuál es el mejor tiempo de los pulsos

La distancia desviada se obtiene de calcular la ecuación de la recta con el punto inicial donde se detecta el robot y su orientación. Y una vez obtenida esta ecuación la distancia se calcula con la **Fórmula 7** de punto a recta con el punto final detectado.

Para llegar a esa fórmula partimos de la ecuación general de la recta en la **Fórmula 3**. Y de la ecuación punto-pendiente de la recta en la **Fórmula 4**, ya que podemos obtener esta recta con el punto inicial de esta y su pendiente. Igualando a 0 obtenemos las variables de la **Fórmula 5** para sustituir posteriormente.

$$Ax + By + C = 0$$

Fórmula 3. Ecuación general de la recta

$$y - y_0 = m(x - x_0) \Rightarrow mx - y + (y_0 - mx_0) = 0$$

Fórmula 4. Ecuación punto-pendiente de la recta

$$A = m, B = -1, C = y_0 - mx_0, x = \text{punto_final}_x, y = \text{punto_final}_y$$

Fórmula 5. Variables de la ecuación punto-pendiente de la recta en la ecuación general

La **Fórmula 6** es la fórmula de un punto cualquiera a una recta. Y la Fórmula 7 es igual pero con los valores obtenidos en la Fórmula 5.

$$distancia = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}$$

Fórmula 6. Distancia de un punto a una recta

$$distancia = \frac{|m * punto_final_x - punto_final_y + (y_0 - m * x_0)|}{\sqrt{m^2 + (-1^2)}}$$

Fórmula 7. Distancia de un punto a una recta con las variables sustituidas

Con la **Fórmula 8** se obtiene la variación de la orientación respecto a cuando empieza y cuando acaba.

$$\Delta_{orientación} = orientación_{final} - orientación_{inicial}$$

Fórmula 8. Variación de orientación

La **Figura 30** muestra la posición inicial de una de las pruebas realizadas y en la **Figura 31** se ve la posición final con la que acaba el robot.

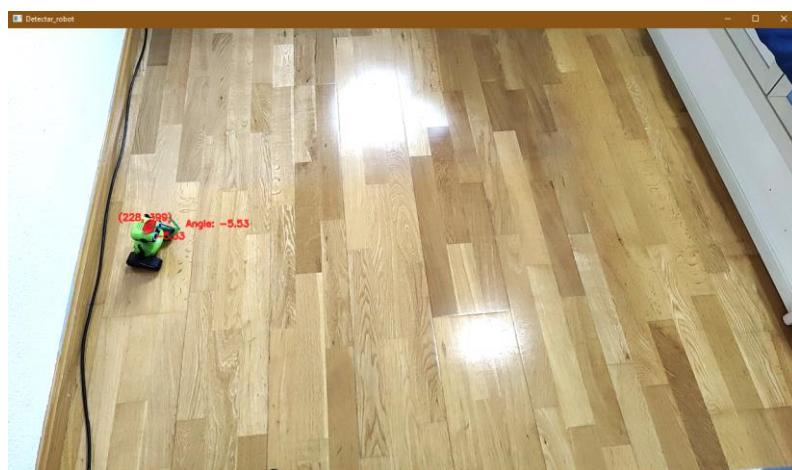


Figura 30. Posición inicial para calibrar movimiento en línea recta

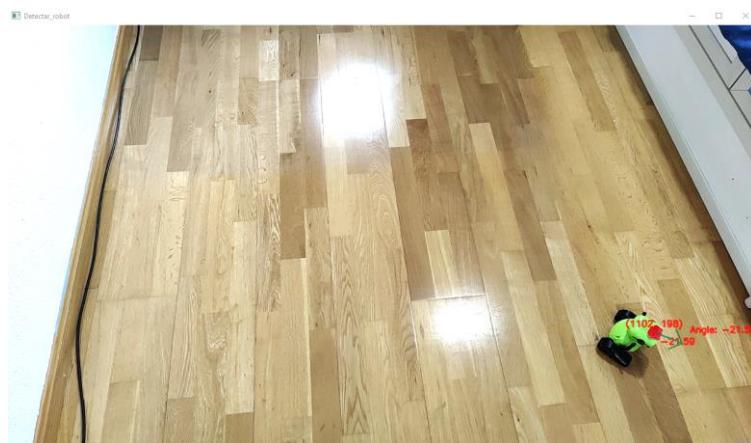


Figura 31. Posición final para calibrar movimiento en línea recta

4.8 Llegar a un punto cualquiera

Para implementar el algoritmo descrito antes se ha usado una función que se ejecuta en bucle hasta que el robot llega al punto objetivo. Antes de que se empiece a ejecutar este bucle el usuario debe marcar el punto objetivo al que quiere que el robot llegue.

El bucle comienza obteniendo el punto de la ubicación del robot para calcular la distancia de este hasta el punto final, si es mayor que un cierto umbral seguirá ejecutando por el contrario finalizará la función. Si sigue ejecutando el siguiente paso es calcular el ángulo del robot con el punto objetivo, y una vez obtenido este ángulo se obtiene el error de orientación al punto objetivo, restando su orientación actual con este ángulo calculado.

Una vez obtenido el error de orientación, si este es menor a un umbral de 30° , el robot avanzará. Si el error está entre 155° y 180° , el robot retrocederá. En cualquier otro caso, el robot deberá girar.

Si la instrucción es avanzar o retroceder se determina cuánto tiempo se tiene que pulsar el respectivo botón en función de la distancia al punto final y las medias calculadas en el apartado 4.3. La distancia se calcula con la **Fórmula 9** y el tiempo con la **Fórmula 10**.

$$\text{distancia} = \sqrt{\Delta x^2 + \Delta y^2}$$

Fórmula 9. Cálculo de la distancia desde la posición del robot al punto objetivo

$$t = \frac{0.1 * \text{distancia}}{\text{media}}$$

Fórmula 10. Cálculo del tiempo que se debe mover el robot en función de la distancia que tiene al objetivo

Además, debido a que el robot tiene imprecisiones en el movimiento no se permite moverse más de 0.4 segundos por avance o retroceso por si el robot se desvía mucho.

En el caso de que la instrucción sea girar, se ha visto que los ángulos entre los que varía el robot son menores cuando hace el giro combinando desplazamiento y giro, en comparación con girar sólo izquierda o derecha. Inicialmente se ha probado realizando todos los giros combinando desplazamiento y giro. Sin embargo, debido a que estos movimientos son circulares hay muchas veces que, al estar cerca del punto, cuando gira no consigue orientarse bien al punto porque se queda dando vueltas a su alrededor, por lo que se decidió cambiar la estrategia. Cuando el robot ya está a un cierto umbral de distancia del punto objetivo, los giros se harán girando sobre sí mismo activando solamente los botones de izquierda o derecha.

Para determinar si tiene que girar a la derecha o la izquierda: si el ángulo de error es negativo entonces tiene que girar a la izquierda y en el caso contrario, a la derecha. También se tiene que calcular cuánto tiempo debe girar el robot para orientarse al punto objetivo, y se hace de la misma forma que la distancia, viendo cuántos grados tiene que girar y con las medias obtenidas en el apartado 4.3 se obtiene el tiempo. Con la **Fórmula 11** se calcula el error de orientación que tiene el robot respecto al punto objetivo y con la **Fórmula 12** se obtiene el tiempo. En la **Fórmula 11** el ángulo del robot al punto se obtiene con la Fórmula 1.

$$\epsilon_{orientación} = ángulo_robot_punto - orientación_robot$$

Fórmula 11. Error entre la orientación del robot y el ángulo al punto objetivo

$$t = \frac{0.1 * \epsilon_{orientación}}{\text{media}}$$

Fórmula 12. Cálculo del tiempo que se debe mover el robot en función de los grados de error con el objetivo

El algoritmo descrito está ilustrado en la **Figura 32**, en la que se puede ver el bucle que realiza el programa hasta llegar a al punto final.

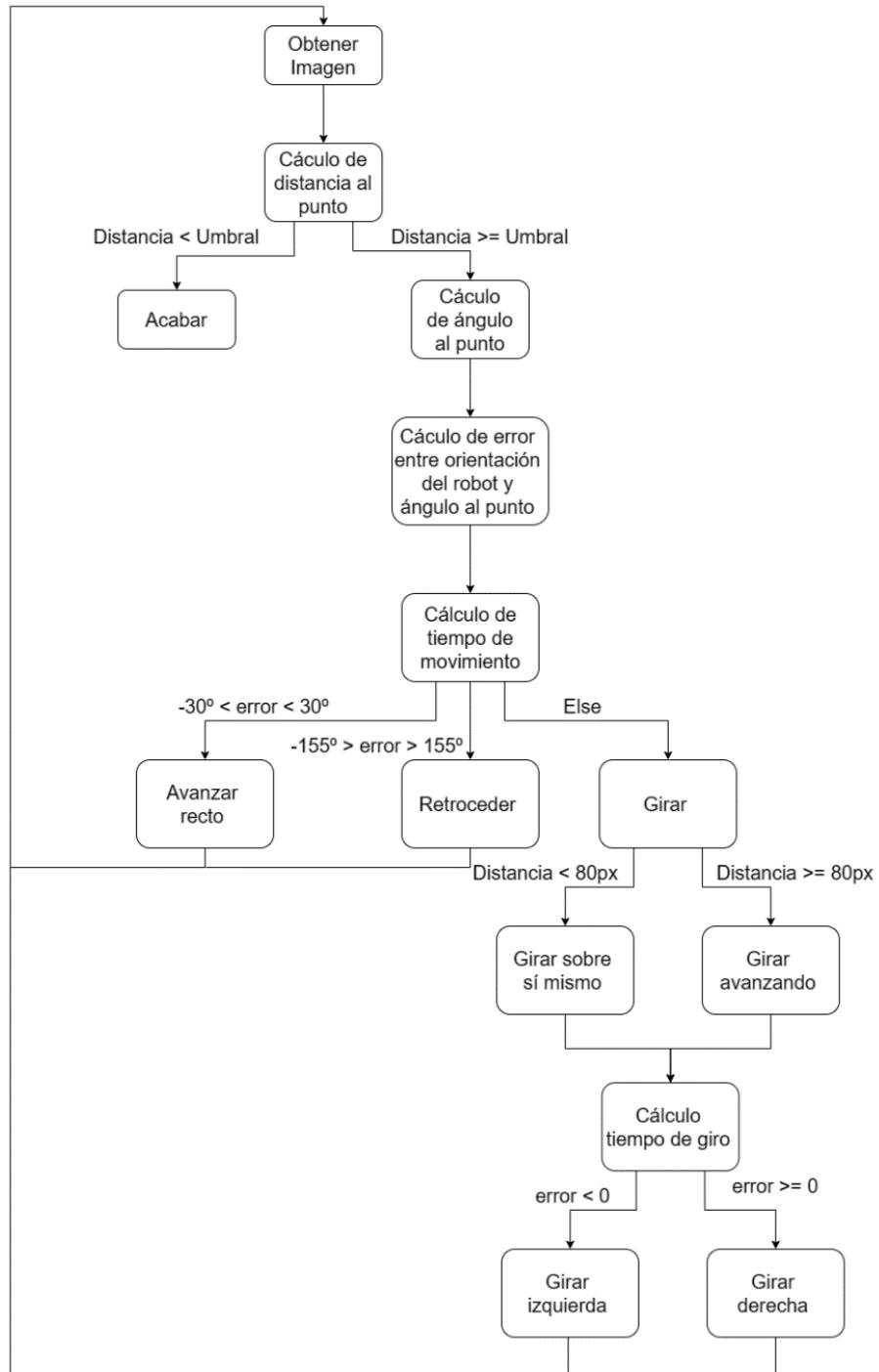


Figura 32. Diagrama del algoritmo para llegar a un punto cualquiera

4.9 Seguir una línea

Como se explicó en el apartado de diseño, la principal diferencia respecto a llegar hasta un punto con seguir una línea es que el robot no tiene que detenerse entre instrucción e instrucción.

En primer lugar, se establece el punto inicial de la línea como punto objetivo para que, utilizando el algoritmo de llegada a un punto, el robot alcance el inicio de la línea. Una vez llegado a este punto, el robot debe tener la misma orientación que tiene la línea, para ello se calcula el ángulo de la línea con el primer punto y el tercero de esta. Se compara la orientación del robot con la de la línea y, si el error es menor que 0 gira sobre sí mismo a la izquierda, y en caso contrario, a la derecha. Este código se ejecuta hasta que el robot se alinee con la línea dentro de un umbral de error. Para que se determine la orientación del robot tras cada giro, se moverá alternativamente hacia delante y hacia atrás porque, como se ha explicado en el apartado 4.2 al girar sobre sí mismo no hay diferencia entre el punto inicial y final, por lo que no se puede calcular la orientación.

En el momento en el que el robot está en el principio de la línea y su orientación alineada con esta, comienza a moverse. Cada 0.5 segundos, se captura una nueva imagen para determinar su posición. Cada vez que se captura la imagen se calculan los parámetros descritos en la parte de diseño:

-Ángulo de la línea en el punto más cercano al robot: Este ángulo se obtiene una vez identificado el punto de la línea más cercano al robot. Y se calcula comparando el punto anterior con el siguiente utilizando la Fórmula 1.

-Punto más cercano de la línea al robot: Para calcular este punto, se examinan todos los puntos de la línea y se selecciona el que tiene la menor distancia al robot. En caso de que sea el primer o el último punto de la línea, se selecciona el siguiente o el anterior punto respectivamente. Esto se hace para poder calcular el ángulo de la línea correctamente, ya que como se ve en la **Figura 33**, el anterior punto del primero de la lista en Python es el último, y el siguiente al último es el primero.

length = 5				
	'p'	'r'	'o'	'b'
index	0	1	2	3
negative index	-5	-4	-3	-2
				'e'

Figura 33. Ilustración una lista en python

-Distancia del robot al punto más cercano: Esta distancia se calcula con las componentes x, y de ambos puntos de igual forma que se calcula la distancia del robot al punto objetivo en el apartado 4.8.

-Orientación del robot: La orientación se calcula cada vez que se muestra la imagen con la Fórmula 1.

-Ángulo del robot hacia la línea: Esta medida es el ángulo que se forma si se uniera en línea recta el punto donde está el robot con el punto más cercano de la línea a él. Y se calcula una vez obtenidos estos dos puntos con la **Fórmula 1**.

Después de obtener estos parámetros, el siguiente paso es calcular la orientación que debe tener el robot en el punto en el que está para seguir la línea correctamente. Esta orientación no puede ser simplemente el ángulo que forma el robot con el punto más cercano de la línea, ya que, de ser así, el robot no avanzaría si no que se movería perpendicularmente a la línea en bucle. Por este motivo se ha desarrollado la **Fórmula 13** que calcula esta orientación dependiendo de la distancia del robot a la línea.

$$\text{orientación} = \text{factor} * \text{ángulo_robot_línea} + (1 - \text{factor}) * \text{ángulo_línea}$$

Fórmula 13. Cálculo de la orientación correcta para el robot siguiendo una línea

De esta manera, cuánto más cerca este de la línea más peso tiene el ángulo de la línea y cuanto más lejos está, más peso tiene el ángulo que se establece entre el robot y el punto de la línea más cercano al robot. Así el robot avanzará en la dirección de la línea si está sobre ella, si está muy lejos, se dirigirá hacia ella, y si está a una distancia intermedia, avanzará hacia la línea mientras sigue su dirección.

El factor de multiplicación se obtiene como se ve en la **Fórmula 14**. En la que se considera el ancho de la imagen dividido entre 3 como una distancia lo suficientemente lejos para que tenga que ir perpendicularmente a la línea. Si el robot estuviera más lejos de esa distancia, la orientación que debe llevar es directamente el ángulo del robot con la línea, es decir, dirección perpendicular a la línea desde el punto donde está el robot.

$$\text{factor} = \frac{\text{distancia_robot_línea}}{\frac{\text{ancho_imagen}}{3}}$$

Fórmula 14. Cálculo del factor de ponderación

Se ha observado que utilizando la **Fórmula 13** hay ciertas posiciones relativa entre el robot y la línea en la que el resultado de la orientación es erróneo si se usan con los ángulos entre [-180, 180] y correctos si se usan entre [0,360] y viceversa. Por lo que no hay una manera de usar los ángulos definitiva que funcione en todos los casos. En la **Tabla 9** se pueden ver distintos cálculos realizados con la fórmula y se observa en los resultados que hay veces que los mismos ángulos, una vez entre [-180, 180] y otra entre [0,360] dan resultados distintos. En verde se muestran los resultados correctos y en naranja los incorrectos.

Distancia	Ángulo de la línea	Ángulo del robot con la línea	Orientación final
80	180	90	166°
80	-180	90	-144°
80	0	90	11°
80	360	90	324
80	300	270	296°
80	-60	-90	-64°
80	170	90	159°

80	-170	90	-135°
80	190	90	176°
80	10	90	20°
80	-10	90	3°
80	350	90	315°

Tabla 9. Distintas pruebas realizadas con el uso de la fórmula de la orientación con resultados correctos e incorrectos

Durante todo el proyecto se ha trabajado con los ángulos normalizados entre $[-180^\circ, 180^\circ]$ pero en este punto se ha visto que para aplicar la fórmula correctamente depende de cada caso. Para encontrar una forma de la que funcione siempre se va a partir de ambos ángulos entre $[0, 360]$ y si su diferencia es mayor que 180 , entonces se normalizarán entre $[-180^\circ, 180^\circ]$ tal y como se muestra en la figura 20. El resultado final se mostrará siempre entre $[-180^\circ, 180^\circ]$. En la **Tabla 10** se realizan las mismas operaciones que en la **Tabla 9** pero en este caso se observa que con los ángulos que son iguales, pero en distinto rango, el resultado es el mismo tras aplicar lo mencionado anteriormente.

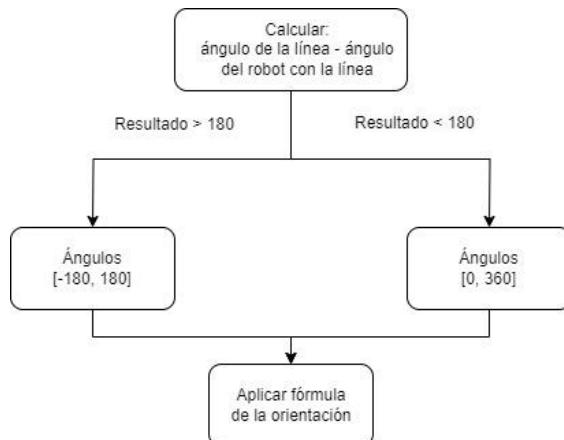


Figura 34. Proceso de normalización de ángulos para que funcione la fórmula de la orientación

Distancia	Ángulo de la línea	Ángulo del robot con la línea	Orientación final
80	180°	90°	168°
80	-180°	90°	168
80	0°	90°	12°
80	360°	90°	12°
80	300°	270°	-64°
80	-60°	-90°	-64°
80	170°	90°	159°
80	-170°	90°	176°
80	190°	90°	176°
80	10°	90°	20°
80	-10°	90°	3°
80	350°	90°	3°

Tabla 10. Distintas pruebas realizadas con el uso de la fórmula de la orientación con todos los resultados correctos

Tras calcular la orientación que debe seguir el robot lo siguiente es determinar qué instrucción hay que mandarle para que siga esta orientación. Primero se calcula el error de orientación entre su orientación y la que tiene que seguir. Si el error es pequeño (menor que cierto umbral), el robot sigue recto. Si el error es negativo, gira a la izquierda y si es positivo, gira a la derecha. Ambos giros se realizan mientras el robot sigue avanzando, resultando un movimiento combinado de avance y giro.

El robot no se detiene en ningún momento ya que las instrucciones son enviadas cada 0.5s tras calcular todos los parámetros, cuando se captura una nueva imagen. El robot sólo se detiene cuando ya está cerca del punto final de la línea. En la **Figura 35** se muestra el algoritmo descrito anteriormente

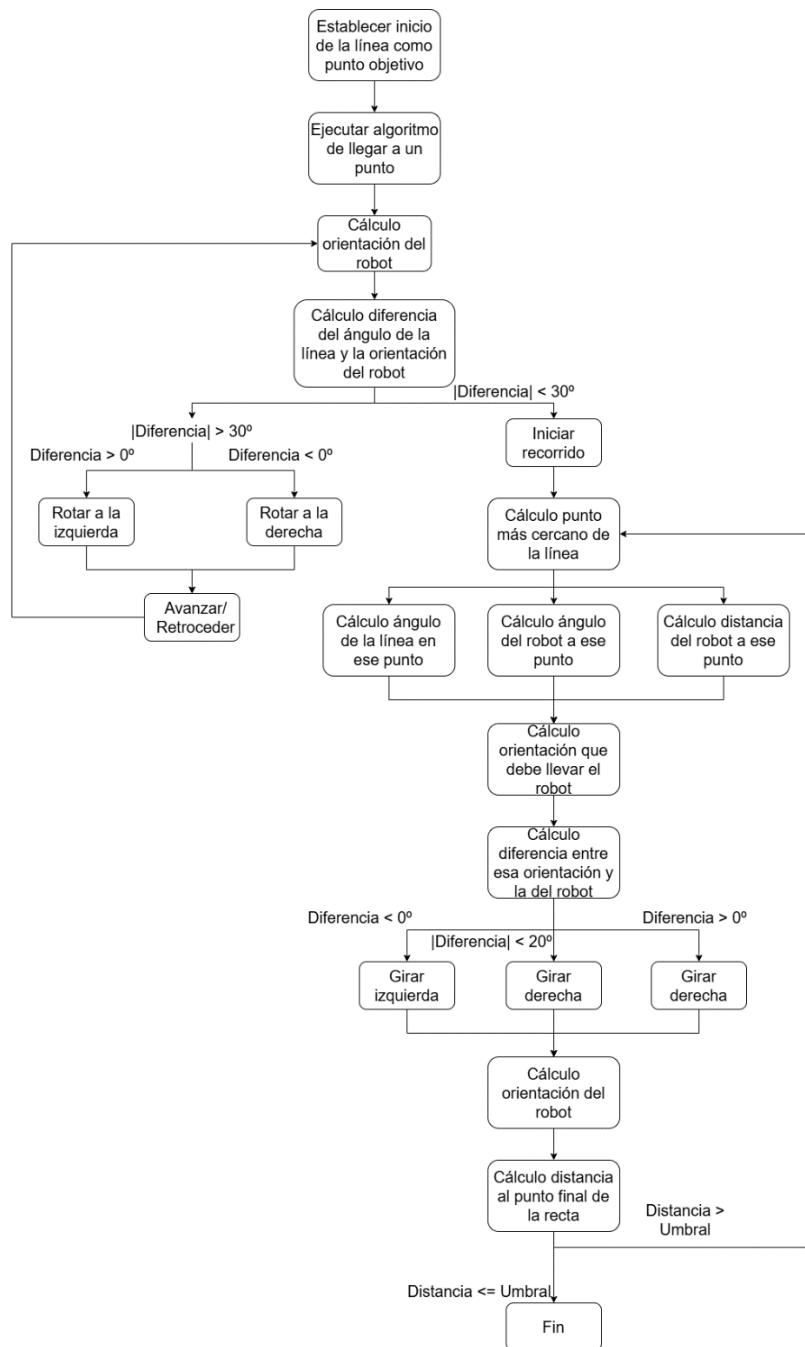


Figura 35. Algoritmo de seguir una línea

Ejecutando esta función el robot no consigue mantenerse en la línea de manera continua. Se ha desarrollado un programa para ver gráficamente que instrucción se manda en cada momento y porque hace esas trayectorias el robot. Este programa se ha realizado gracias a los logs que genera el programa y muestra gráficas como la de la **Figura 36** con los puntos en azul por donde ha pasado el robot, y puntos en rojo para ilustrar la línea.

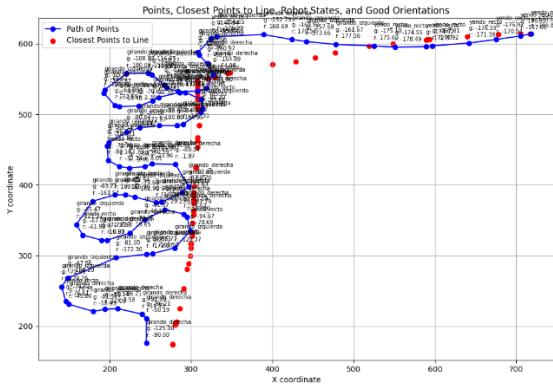


Figura 36. Gráfica del recorrido del robot utilizando el algoritmo de seguir una línea

Si se amplia la gráfica, en la **Figura 37**, se puede ver lo que sucede. A simple vista se aprecia que se envían instrucciones pero el robot no comienza a ejecutarlas hasta unos movimientos más tarde. En el punto rodeado, se envía la instrucción de girar a la derecha, pero , el robot no empieza a girar hasta el punto señalado por la flecha.

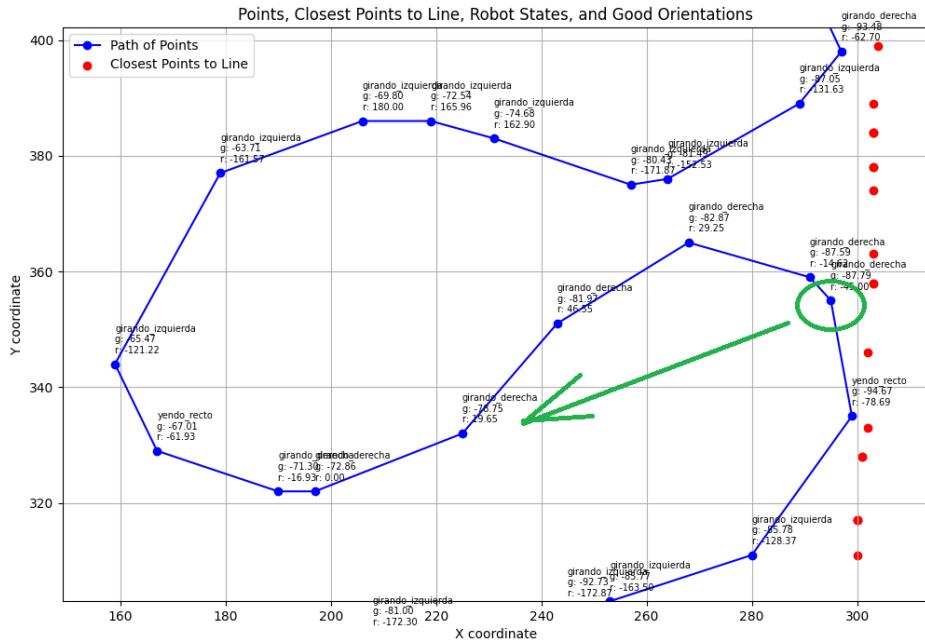


Figura 37. Gráfica ampliada del recorrido del robot utilizando el algoritmo de seguir una línea

Se prueba el mismo código en el gemelo virtual, y en la **Figura 38** se muestra el recorrido seguido por el robot simulado. A pesar de haber simulado todas las imprecisiones el robot virtual logra seguir la línea con una trayectoria mucho más precisa que el robot físico.

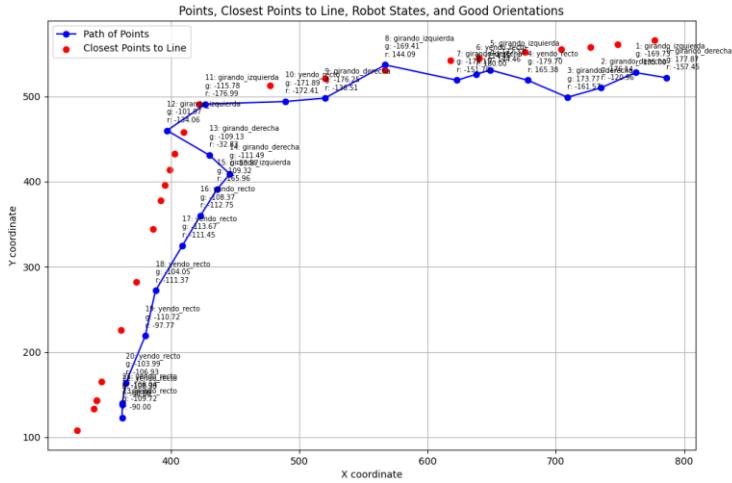


Figura 38. Gráfica del recorrido del gemelo virtual utilizando el algoritmo de seguir una línea

Para intentar solucionar que el robot haga esos recorridos en zigzag se ha cambiado el umbral del error entre la orientación del robot y la orientación que debe seguir en el momento que se tiene que decidir si seguir recto o girar. Teniendo la orientación del robot, el ángulo de la línea y el ángulo del robot al punto más cercano de la línea se puede saber si el robot se está acercando o alejando de ella con la **Fórmula 15**.

$$\text{ángulo} = |\text{orientación_robot} - (\text{ángulo_línea} + \text{ángulo_robot_línea})|$$

Fórmula 15. Cálculo del ángulo para saber si el robot se aleja o se acerca a la línea

Si el resultado está entre 0 y 90 o entre 270 y 360 significa que el robot se está acercando a la línea y si está entre 90 y 270 el robot se está alejando de ella. Cuando el robot se acerque el umbral será mayor para que no tenga que girar tantas veces cuando se esté acercando.

En la **Figura 28** y en la **Figura 29** se observa un resultado similar ya que cuando la instrucción que tiene que ejecutar el robot es de ir recto, este sigue girando con la anterior instrucción. Los colores de los puntos significan lo mismo que en las anteriores figuras, el morado es la trayectoria del robot fuera de la línea, es decir, a más de 80 píxeles de distancia de la línea, la cual se ha considerado como la distancia límite para determinar que el robot está fuera de la línea.

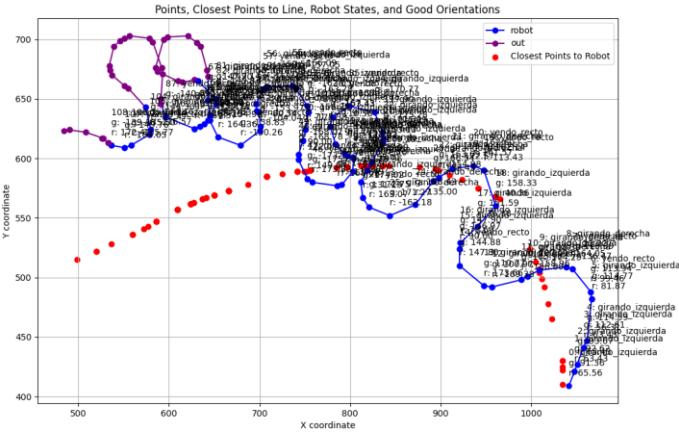


Figura 39. Recorrido del robot tras ajustar el umbral de error entre las orientaciones

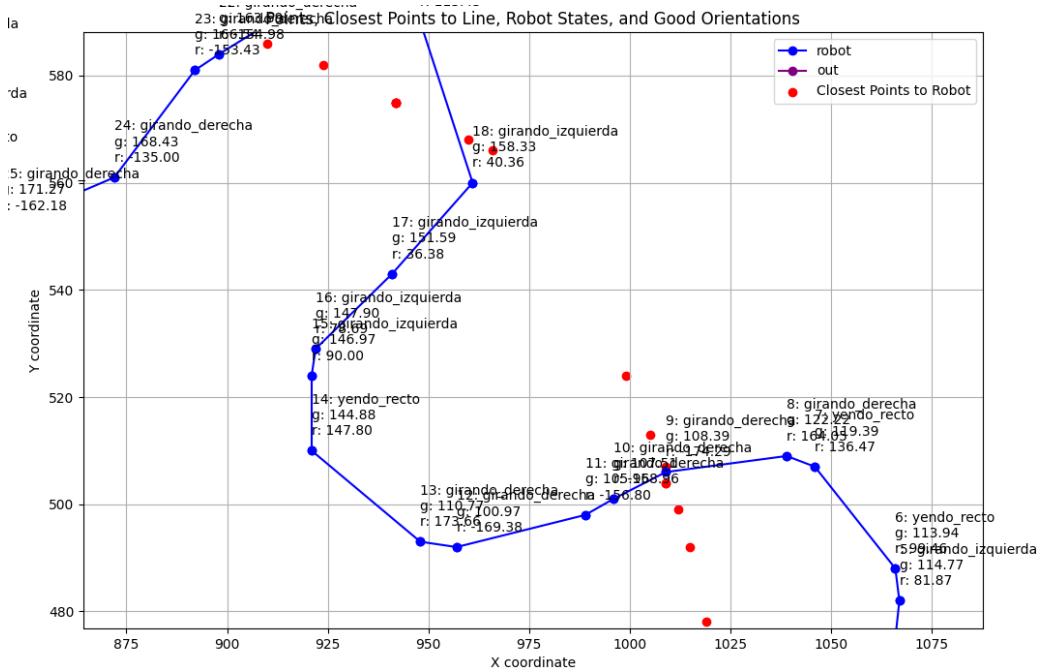


Figura 40. Recorrido del robot ampliado tras ajustar el umbral de error entre las orientaciones

Se ha llegado a la conclusión de que las instrucciones enviadas por el ordenador llegan con cierta latencia al robot y por eso empieza a ejecutarlas más tarde. Para solucionar este problema se han valorado varias opciones.

4.10 Latencia

En la primera opción para solucionar el problema de la latencia se da por hecho que el robot se va a salir de la línea por lo que cuando se detecte que el robot está fuera de la línea, el robot se va a detener y se ejecutará el movimiento inicial de llegada al inicio de la línea y alinearse con ella, pero el punto objetivo será el más cercano de la línea al robot.

En la segunda se va a intentar estimar el punto en el que le llega la instrucción al robot y con la última opción cuando el robot se acerque a una curva se ralentizará su velocidad para que no se salga de la línea con tanta facilidad.

Como última opción se ralentizará el robot en las proximidades de las curvas para evitar que se desvíe con tanta facilidad en ellas. Se utilizará un método llamado PWM para ralentizar el robot, que consiste en enviar pequeños pulsos de activación al robot en vez de que avance continuamente.

4.10.1 Seguir línea volviendo a ella

En este método se modifica el comportamiento del robot cuando se detecta que está a más de una cierta distancia de la línea, establecida en 80 píxeles. En ese caso el robot se detiene y se vuelve a ejecutar el código inicial para que el robot llegue a un punto, que será el punto de la línea más cercano al robot. Una vez sobre la línea, el robot se alinea de nuevo con ella, igual que al inicio.

En la **Figura 41**, se puede ver en azul el movimiento del robot cuando está cerca de la línea y en naranja el movimiento cuando se ha detectado que estaba más de 80 píxeles de distancia. Cuando se detecta que está lejos se puede observar el camino en naranja que ha seguido para volver al punto más cercano de la línea.

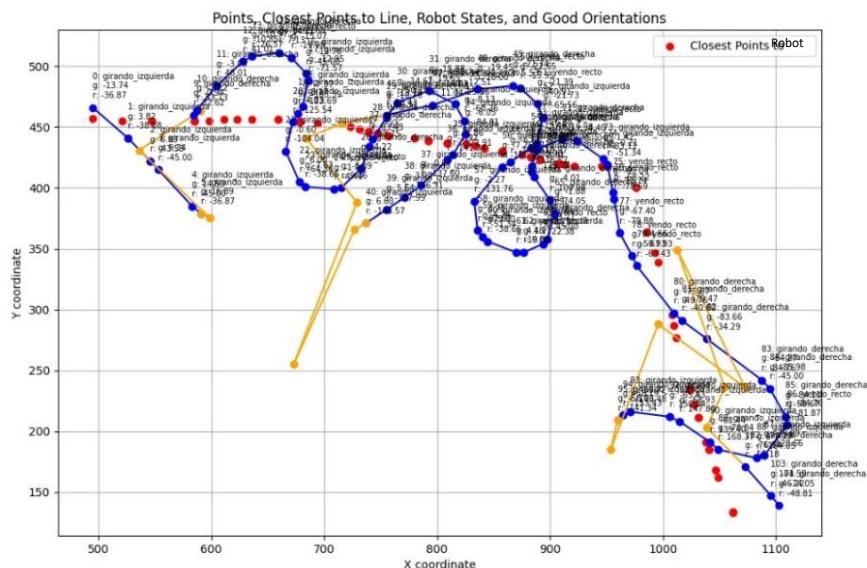


Figura 41. Gráfica del recorrido del robot siguiendo la línea volviendo a ella cuando se aleja

Como se puede observar en la **Figura 42**, el resultado es similar al del método anterior, ya que las instrucciones siguen llegando con latencia. Esto provoca que, aunque este cerca de la línea el robot continúe haciendo movimientos en zigzag. Además, cuando se detecta que el robot está lejos y debe detenerse para volver a la línea, el problema de la latencia provoca que el robot siga avanzando una vez mandada la instrucción de detenerse.

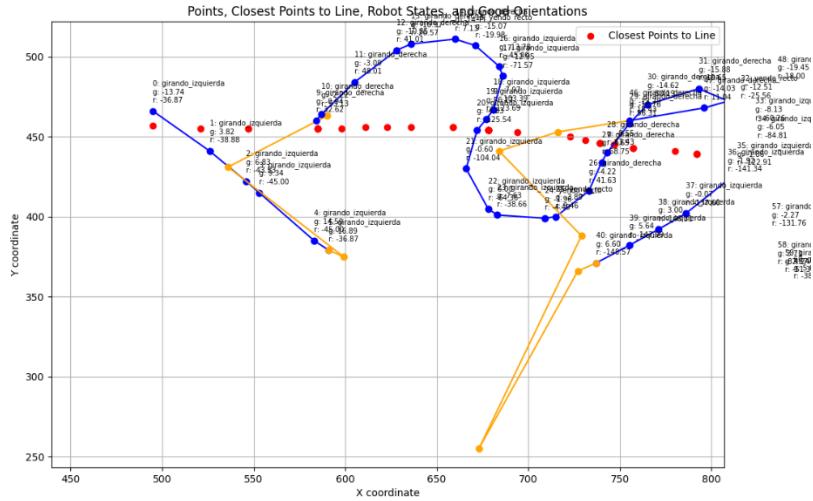


Figura 42. Gráfica ampliada del recorrido del robot siguiendo la línea volviendo a ella cuando se aleja

Otro inconveniente de este método es que, una vez llegado a la línea, debido a la imprecisión en los giros del robot, en ocasiones tarda bastante en alinearse con línea, lo que ralentiza mucho que siga avanzando por ella. Además, debido al umbral de error en la diferencia entre la orientación del robot y la línea, es muy difícil que el robot se alinee exactamente con la línea, y con el mínimo avance, el robot se desvía nuevamente.

4.10.2 Seguir línea estimando donde llega la instrucción

Con este método se intentará estimar el punto dónde estará el robot en el momento que ejecute la instrucción. De esta manera se enviarán las instrucciones teniendo en cuenta el punto estimado y no el punto actual.

Los movimientos del robot también dependen de cuantas veces seguidas ha sido mandada la misma instrucción, no gira igual cuando con la primera instrucción de giro que cuando ha recibido 5 seguidas.

Para estimar el punto, se van a calcular 3 variables distintas, y cada una desde 0 instrucciones seguidas, hasta 9 instrucciones seguidas (en todas las pruebas realizadas rara vez se han superado las 9 instrucciones consecutivas). Además, estas 3 variables para los 10 posibles casos se calculan para cada tipo de instrucción, seguir recto o girar.

Las variables son:

-Cuantos píxeles avanza el robot entre una instrucción y otra.

-La **diferencia de orientación** del robot entre una instrucción y otra: Esta variable se refiere a la diferencia entre la orientación que tiene el robot en una instrucción y la orientación en la siguiente instrucción.

-El **ángulo de movimiento** del robot: A diferencia de la anterior variable, esta recoge la dirección con la que se ha movido el robot para llegar del punto donde está cuando se envía una instrucción, al punto donde está en la siguiente instrucción.

Teniendo en cuenta todo esto se ha registrado el movimiento del robot numerosas veces para calcular cual es la media de cada movimiento con el número de instrucciones consecutivas, y el número de ejecuciones en las que se ha producido esa situación mientras se recogían los datos para calcular las medias.

En la **Tabla 11** están recogidas las medias del ángulo de movimiento del robot cuando debe ir recto. Como se puede observar, a medida que aumenta el número de instrucciones seguidas, el ángulo disminuye, lo que indica que el robot sigue una trayectoria más recta. En la **Tabla 12** se presentan las medias del mismo parámetro, pero para el caso en el que el robot debe girar. Se observa que inicialmente el ángulo de giro es menor, alcanza su máximo con 5 instrucciones seguidas y luego vuelve a descender.

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	89.66°	250
1	56.03°	95
2	38.62°	14
3	7.35°	3
4	6.86°	3
5	6.96°	3
6	6.11°	3
7	5.92°	2
8	5.32°	2
9	6.99°	2

Tabla 11. Parámetros del ángulo de movimiento del robot con la instrucción de ir recto

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	54.02°	291
1	44.60°	287
2	82.19°	280
3	98.45°	271
4	107.02°	262
5	111.68°	233
6	83.72°	171
7	78.89°	85
8	76.03°	39
9	57.21°	11

Tabla 12. Parámetros del ángulo de movimiento del robot con la instrucción de girar

En la **Tabla 13** se pueden ver las medias del parámetro de orientación cuando el robot tiene que ir recto. Al igual que en la **Tabla 12**, cuanto menor es el número de instrucciones consecutivas, mayor es la media, lo que indica que el robot no sigue una trayectoria recta

hasta que recibe varias instrucciones consecutivas. En la **Tabla 14** están las medias para el mismo parámetro, pero con la instrucción de girar.

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	111.37°	250
1	63.22°	95
2	39.97°	14
3	11.64°	3
4	9.88°	3
5	9.99°	3
6	8.32°	3
7	7.36°	2
8	8.27°	2
9	8.71°	2

Tabla 13. Parámetros orientación del robot con la instrucción de ir recto

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	59.74°	291
1	67.61°	287
2	118.82°	280
3	141.53°	271
4	139.01°	262
5	138.93°	233
6	125.29°	171
7	120.42°	85
8	108.08°	39
9	84.48°	11

Tabla 14. Parámetros orientación del robot con la instrucción de girar

En la **Tabla 15** se presentan las medias de la distancia avanzada cuando el robot tiene que ir recto. En este caso se observa que, a medida que aumenta el número de instrucciones consecutivas la distancia es mayor. Esto se debe a que, si el robot sigue una trayectoria recta avanzará más que si sigue una curva. Por eso, al principio, las medias son más bajas debido a que sigue una trayectoria curva. En la **Tabla 16** se muestran las medias para cuando el robot está girando.

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	49.86px	250
1	53.19px	95
2	57.88px	14
3	92.74px	3
4	78.18px	3
5	74.54px	3
6	79.73px	3
7	94.63px	2
8	111.42px	2
9	100.01px	2

Tabla 15. Parámetros de distancia con la instrucción de ir recto

Instrucciones Consecutivas	Media	N.º de ejecuciones
0	55.20px	291
1	55.91px	287
2	51.96px	280
3	48.60px	271
4	46.91px	262
5	47.39px	233
6	47.80px	171
7	47.84px	85
8	44.76px	39
9	42.14px	11

Tabla 16. Parámetros de distancia con la instrucción de girar

Tras haber obtenido todas las medias, en lugar de estimar directamente el punto donde llegará la instrucción, se intentará estimar únicamente el siguiente punto. Esto permitirá comprobar si, con los datos recogidos, se pueden estimar correctamente los puntos donde estará el robot cuando llegue la instrucción. En una primera prueba reflejada en la **Figura 43** y en la **Figura 44** se puede observar un claro error en los puntos estimados, que aparecen en color verde.

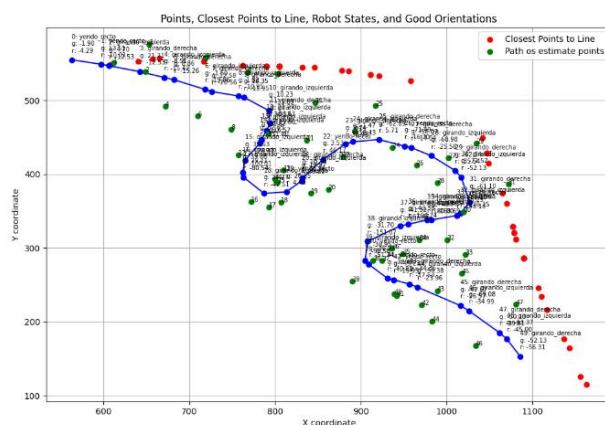


Figura 43. Gráfica del recorrido del robot estimando el siguiente punto

Se observa que los puntos estimados están lejos de los puntos reales. Esto se debe a que, por la latencia, cuando se cambia de instrucción el robot sigue ejecutando la instrucción anterior. Por ejemplo, si el robot está girando a la derecha y la siguiente instrucción es ir recto, el robot sigue girando a la derecha. Aunque se envíe la instrucción de ir recto varias veces consecutivas, las primeras veces el robot aún está recibiendo las instrucciones de girar a la derecha. Esto hace que la media de las primeras instrucciones seguidas esté alterada ya que, el robot sigue moviéndose con las instrucciones anteriores. Y por eso los puntos estimados quedan lejos de los reales, ya que la mayoría se han calculado con medias de movimientos que corresponden a movimientos anteriores.

Además, en la prueba realizada las instrucciones se han calculado en función del punto siguiente que se ha estimado. En la **Figura 44**, las 5 primeras instrucciones son: recto, recto, izquierda, derecha, izquierda. Que no tienen mucho sentido al estar mal estimados

los puntos. Pero se puede ver que la trayectoria que sigue el robot es prácticamente una línea recta. Esto es porque como solo hay 1 estado de giro no es suficiente como para que el robot gire.

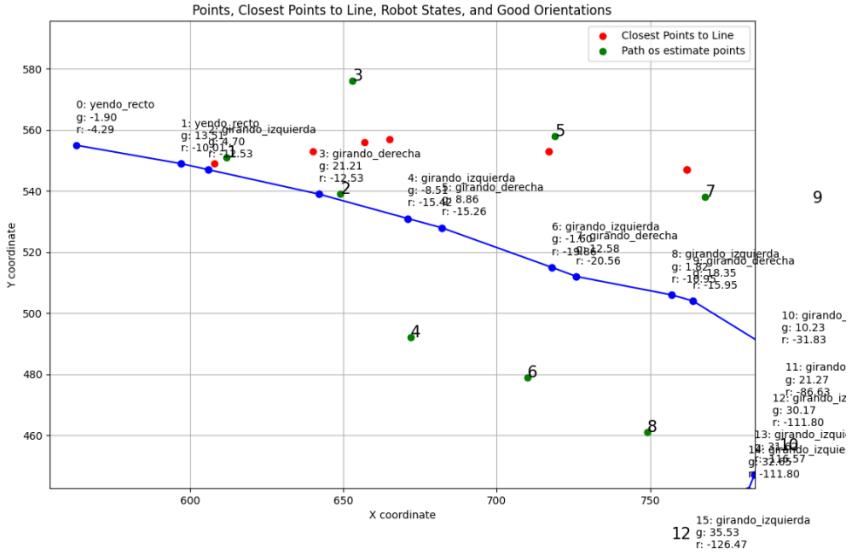


Figura 44. Gráfica del recorrido del robot ampliado estimando el siguiente punto

Por lo que a parte de las instrucciones seguidas que hayan sido enviadas al robot, tambien afecta cuantas instrucciones seguidas habían sido enviadas antes de cambiar de instrucción.

Además, se ha observado que la latencia no es fija, lo que dificulta estimar el punto donde llegará la instrucción sin saber cuándo recibirá. Esto, sumado a que estimar el siguiente punto simplemente, requiere considerar una gran cantidad de factores, hace que este método no sea viable.

4.10.3 Ralentizar el giro del robot

Otra opción es usar un PWM para ralentizar la velocidad del robot cuando se aproxima a curvas de la línea, intentando que así no se desvíe tanto. Primero, se ha desarrollado una función que permite detectar los giros de la línea. Para ello, se calcula la media de los ángulos entre los 10 primeros puntos de la línea, comparando el primer punto con el segundo, el segundo con el tercero, y así sucesivamente. Luego se compara cada punto de la línea, uno a uno, con la media de los ángulos calculada entre los 10 puntos anteriores y los 10 siguientes de manera similar; el décimo anterior con el noveno siguiente, el noveno anterior con el octavo anterior, y así sucesivamente hasta el noveno siguiente con el décimo siguiente. Si la diferencia entre la primera media y la media de un punto supera los 30 grados, ese punto se detecta como una curva y, a partir de ahí, la nueva media a comparar es la de ese punto.

En la **Figura 46** y en la **Figura 47**, se pueden ver distintas líneas y en color verde los puntos detectados como curvas.

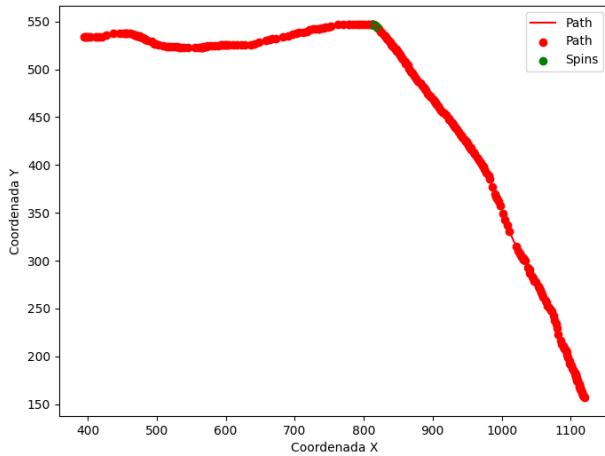


Figura 46. Detección punto de giro curva abierta

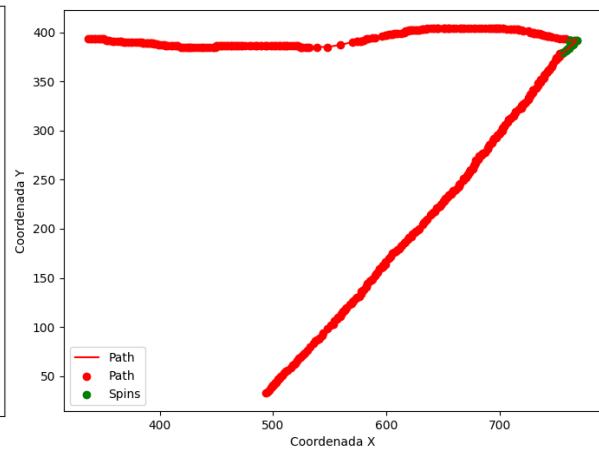


Figura 45. Detección punto de giro en curva cerrada.

Debido al problema de la latencia, es necesario enviar la instrucción de ralentizarlo con antelación. Sin embargo, esto no debería ser un problema, ya que simplemente hará que el robot avance más despacio.

Para ralentizar el robot se va a usar el método de PWM que ralentiza el robot mandando pulsos intermitentes de activación a los motores. Primero se va a calcular las velocidades medias cambiando los tiempos de los pulsos, realizando pruebas y recogiendo la media de cuantos pixeles avanza en 0.1s. Estas medias se pueden ver en la **Tabla 17**.

Tiempo pulsado	Tiempo detenido	Píxeles/ds
0.05	0.05	26.59
0.05	0.05	28.90
0.05	0.05	25.88
	0	41.53
	0	46.83
	0	44.57
0.1	0.1	13.45
0.1	0.1	20.19
0.1	0.1	16.44
0.2	0.1	27.28
0.2	0.1	25.43
0.2	0.1	21.82

Tabla 17. Medias de velocidad en pixeles por 0.1s con distintos tiempos de los pulsos

Este método utiliza el código que detiene el robot cuando se sale de la línea, para intentar que avance en línea recta hasta el punto de giro. En las proximidades de ese punto, se reducirá la velocidad para realizar el giro y luego continuar por la línea.

Para asegurar que el robot avance recto se combinará este método con la función desarrollada en el apartado 4.7 para mantener el robot en línea recta sin desviarse. Habrá un pulso dedicado al botón de avanzar para reducir la velocidad y otro al botón de la izquierda para evitar que se desvíe a la derecha.

Los problemas que surgen con este método son similares a los que encontrados anteriormente. Primero, cuando el robot no se desplaza en línea recta sobre la línea y recibe la instrucción de girar, el robot comienza a hacer los movimientos en zigzag

generados por la latencia, lo hace que ralentizar el robot al aproximarse a una curva no tenga sentido en este caso.

Otro problema que surge con este método es el mismo que sucedía con el método de seguir la línea y volver a la línea cuando el robot se ha desviado. Cuando el robot regresa a la línea y debe alinearse con ella, en ocasiones no lo hace de manera precisa por varios motivos. Primero, que para determinar la orientación del robot después de girar, es necesario enviar una instrucción al robot para que avance o retroceda. Después de varias maniobras avanzando y retrocediendo para calcular la orientación, el robot termina saliéndose de la línea, repitiendo el problema inicial.

Además, como ya se ha comentado anteriormente, el robot no es preciso en sus movimientos, por lo que muchas veces las instrucciones calculadas no lo alinean perfectamente con la línea. Como se ha permitido un umbral de error, este umbral provoca que el robot se desvíe con rapidez cuando avanza recto, ya que no está correctamente alineado.

Si se permitiera un umbral de error más pequeño el robot tardaría mucho en alinearse con la línea y al final se acabaría saliendo de ella como se ha comentado antes. Por otro lado, si el umbral es más grande, permite que el robot avance sin estar perfectamente alineado con la línea lo que hace que se vuelva a desviar.

5. Experimentos

A continuación, se realizarán una serie de experimentos para tomar medidas y comprobar la eficacia de los algoritmos planteados. Primero, se recogerán datos del algoritmo de llegada a un punto cualquiera detallado en el apartado 4.8 utilizando el robot físico y se compararán los resultados del mismo algoritmo en el gemelo virtual.

Luego, se probará el algoritmo de seguir una línea, desarrollado en el apartado 4.9, y se comparará con el mismo algoritmo en el gemelo virtual, así como con el algoritmo de seguir una línea volviendo a ella cuando se desvíe, descrito en el apartado 4.10.1.

Todos los experimentos han sido realizados en el mismo entorno: un suelo de madera sin obstáculos, donde el robot puede moverse libremente. También es importante aclarar que, a pesar de intentar cada prueba dentro de cada experimento sea lo más similar posible, en ocasiones, al tener que colocar el robot a mano, las pruebas tenían ligeras variaciones. Sin embargo, estas diferencias han sido mínimas.

5.1 Llegar a un punto cualquiera

Se realizarán tres pruebas, cada una de ellas aumentando la dificultad para que el robot llegue al punto. Este punto estará aproximadamente a la misma distancia en los 3 casos, y se medirá la cantidad de instrucciones necesarias para llegar y el tiempo que tarda en alcanzarlo. El tiempo en estos experimentos simplemente sirve como referencia ya que, los movimientos del robot físico son algo más lentos que los del robot virtual.

5.1.1 Llegar a un punto justo delante del robot

Este es el caso más sencillo ya que, como se ve en la **Figura 47**, el punto esta justo delante del robot.

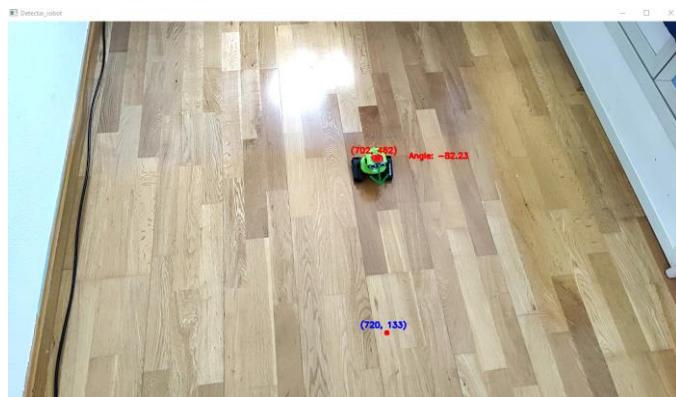


Figura 47. Situación de un punto justo delante del robot

Robot físico

En la **Tabla 18** se muestran los datos obtenidos de 5 pruebas realizadas. La media de tiempo y movimientos, que se puede ver en la **Tabla 19**, es de 10.6 segundos y 5.2 movimientos, lo que indica que cada movimiento dura aproximadamente 2 segundos. Esta relación se mantiene en todas las pruebas realizadas. También se observa que la

mayoría de las pruebas están en torno a los 5 movimientos, aunque hay una con 7. Esto se debe a la imprecisión en los movimientos del robot, como se veía en la calibración en los apartados 4.3 y 4.4. Si en un movimiento se ha desviado considerablemente, ha sido necesario emplear más movimientos.

Movimientos	Tiempo
4	8.3s
7	14.9s
6	11.5s
5	9.9s
4	8.4s

Tabla 18. Datos recogidos de las pruebas realizadas para llegar a un punto delante del robot

Medida	Media
Movimientos	5.2
Tiempo	10.6s

Tabla 19. Medias recogidas de las pruebas realizadas para llegar a un punto delante del robot

Gemelo virtual

Como se aprecia en la **Tabla 20**, en las 4 de las 5 pruebas realizadas, el robot ha necesitado 3 movimientos. En la otra prueba, el robot ha necesitado 5 movimientos. Debido a que el gemelo virtual simula las impresiones en los giros y avances del robot, al igual que usando el robot físico, en una de las pruebas el gemelo ha necesitado 5 movimientos, dos más que en el resto de las pruebas. Esto se debe a que, si en un movimiento el robot se desvía considerablemente, es necesario compensar con más movimientos.

Movimientos	Tiempo
3	4.5s
3	5.4s
5	9.9s
3	4.5s
3	4.3s

Tabla 20. Datos recogidos de las pruebas realizadas para llegar a un punto delante del gemelo virtual

La media de movimientos que se ve en la **Tabla 21** es aproximadamente de 2 movimientos menos respecto al robot físico. Esto da a entender que en el gemelo virtual el algoritmo es ligeramente más efectivo que en el robot físico.

Medida	Media
Movimientos	3.4
Tiempo	5.7s

Tabla 21. Medias recogidas de las pruebas realizadas para llegar a un punto delante del gemelo virtual

5.1.2 Llegar a un punto justo detrás del robot

En este experimento se complica la situación del punto se posiciona justo detrás del robot como se aprecia en la **Figura 48**.

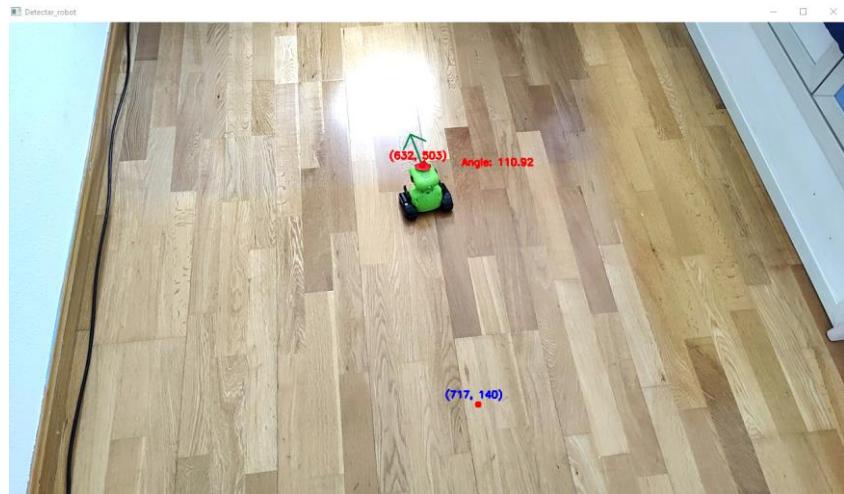


Figura 48. Situación de un punto justo detrás del robot

Robot físico

Se puede observar en la **Tabla 22** y la **Tabla 23** que los resultados son prácticamente idénticos al primer experimento, las medias son de 5 movimientos y de 10 segundos. En el primer experimento con el robot físico eran de 5.2 movimientos y 10.6 segundos. Esto es debido a que el algoritmo de navegación es capaz de detectar si el punto está justo detrás del robot y entonces simplemente tiene que retroceder, convirtiéndose en una situación similar a la planteada en el primer experimento del apartado 5.1.1.

Movimientos	Segundos
5	10
5	11.1
5	10
6	11.7
4	7.3

Tabla 22. Datos recogidos de las pruebas realizadas para llegar a un punto detrás del robot

Medida	Media
Movimientos	5
Tiempo	10s

Tabla 23. Medias recogidas de las pruebas realizadas para llegar a un punto detrás del robot

Gemelo virtual

En este caso, como indican la **Tabla 24** y la **Tabla 25**, el gemelo virtual ha necesitado aproximadamente medio movimiento más de media para llegar al punto y 1.3 segundos más. Además, también se aprecia variabilidad en el número de movimientos en las pruebas realizadas, desde 3 movimientos hasta 6. Esto se debe a que, en algunos casos, el gemelo virtual se ha dirigido marcha atrás hasta llegar al punto, mientras que en otros casos se ha dado la vuelta y ha avanzado hacia delante. Esto es debido a las imprecisiones simuladas, ya que la orientación del robot ha variado mientras el robot retrocedía. Y al variar la orientación el robot deja de estar alineado con el punto y el algoritmo ha decidido girar para luego avanzar recto.

Movimientos	Segundos
4	6.7s
6	10.1s
3	4.6s
6	9.8s
5	8.6s

Tabla 24. Datos recogidos de las pruebas realizadas para llegar a un punto detrás del gemelo virtual

Medida	Media
Movimientos	4
Tiempo	8s

Tabla 25. Medias recogidas de las pruebas realizadas para llegar a un punto detrás del gemelo virtual

5.1.3 Llegar a un punto detrás del robot en diagonal

En el siguiente experimento el punto objetivo estará colocado detrás y en diagonal al robot, como se muestra en la **Figura 49**, ya que se ha considerado que en esta situación se añade dificultad al experimento anterior.

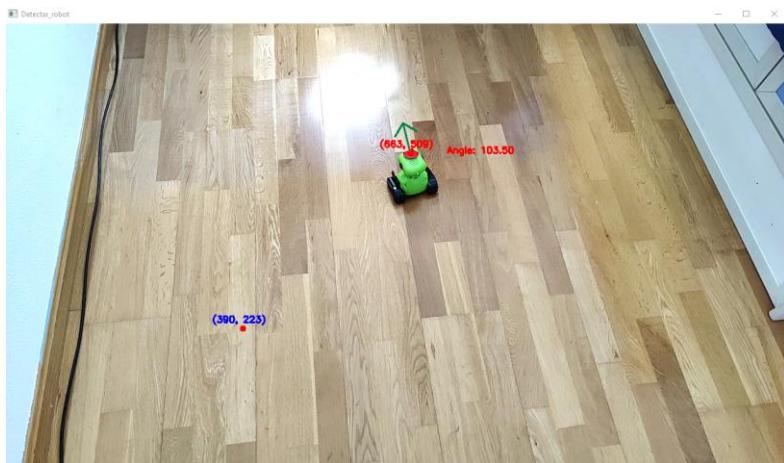


Figura 49. Situación de un punto detrás y en diagonal del robot

Robot físico

Como se muestra tanto en los datos de la **Tabla 26** como en las medias de la **Tabla 27**, los resultados son muy similares a los del primer experimento con el robot físico. Se observa que la media de movimientos es exactamente la misma, 5.2, y que tarda 1 segundo más de media. Que la media de movimientos sea la misma pero que la del tiempo sea de 1 segundo más, es debido a que en este caso el robot primero tiene que girar y luego avanzar. Y el giro es un movimiento doble, porque primero debe girar y luego avanzar un poco para que se pueda determinar su orientación, como se ha explicado en el apartado 4.1.

Movimientos	Segundos
6	14s
6	11.5s
4	8.3s
5	12.3s
5	12.1s

Tabla 26. Datos recogidos de las pruebas realizadas para llegar a un punto detrás y en diagonal del robot

Medida	Media
Movimientos	5.2
Tiempo	11.6s

Tabla 27. Medias recogidas de las pruebas realizadas para llegar a un punto detrás y en diagonal del robot

Gemelo virtual

De igual manera que en el primer experimento se observa que el algoritmo en el gemelo virtual es más eficiente que en el robot físico. Y comparando directamente estos datos y medias de la **Tabla 28** y la **Tabla 29** con el primer experimento del gemelo virtual se ve que, aunque la media de movimientos es de 1 movimiento menos, la media del tiempo empleado es solo 0.3 segundos más, y esto es porque de nuevo el gemelo tiene que usar movimientos de giro.

Movimientos	Segundos
3	6.7s
2	4.2s
2	5.4s
2	4.3s
3	6.6s

Tabla 28. Datos recogidos de las pruebas realizadas para llegar a un punto detrás y en diagonal del gemelo virtual

Medida	Media
Movimientos	2.4
Tiempo	5.4s

Tabla 29. Medias recogidas de las pruebas realizadas para llegar a un punto detrás y en diagonal del gemelo virtual

5.2 Seguir una línea recta

Se realizarán 2 experimentos en los que se comparará el algoritmo de seguir una línea descrito en el apartado 4.8, con el algoritmo de seguir una línea y volver a ella cuando se ha desviado detallado en el apartado 4.10.1, los dos utilizando el robot físico, y además con el primer algoritmo en el gemelo virtual. La situación planteada en cada experimento será progresivamente más compleja, de esta forma se podrá evaluar la eficacia de los algoritmos en las dos posibles situaciones en las que el robot se puede encontrar una línea recta. Estando ya al inicio de esta o que el robot tenga que ir hacia ella.

Para evaluar la eficacia y eficiencia de los algoritmos se recogerán distintos datos. Con el primer algoritmo, se medirán los segundos que tarda en llegar al inicio y los movimientos empleados, los movimientos y segundos que tarda en orientarse una vez está en el punto inicial de la línea, la cantidad de movimientos fuera de línea, los movimientos y el tiempo totales en recorrer la línea.

Con el segundo algoritmo se recogerán los mismos datos y, además, los movimientos y segundos totales empleados para volver a la línea cada vez que se salga, y los movimientos y segundos totales utilizados para volver a alinearse con ella cada vez que la alcanza.

5.2.1 Seguir una línea recta enfrente del robot

Como se ve en la **Figura 50** esta es la situación más sencilla posible, una línea recta justo delante del robot.

Algoritmo de seguir una línea

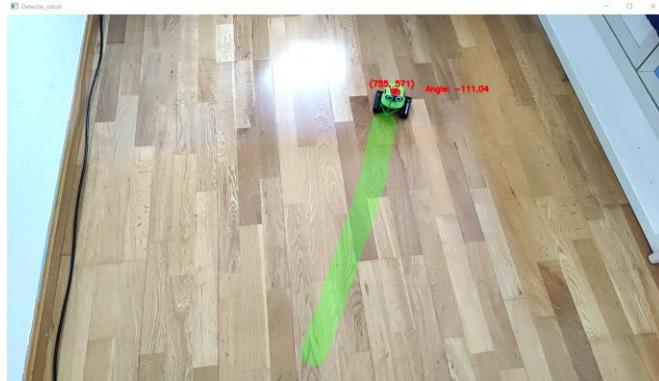


Figura 50. Situación de una línea recta justo delante del robot

En la **Tabla 30** se recogen los datos de las pruebas realizadas para este experimento y en la **Tabla 31** están las medias de estos datos. Los movimientos y segundos en orientarse no son 0 en todos los casos, ya que al tener que colocar el robot manualmente ha debido haber alguna diferencia entre la orientación del robot y la de la línea en algunas pruebas. De las medias obtenidas lo más destacable es que el 18% de los movimientos han sido fuera de la línea, y esto servirá para compararlo con el resto de los experimentos.

Movimientos en orientarse	Segundos en orientarse	Movimientos fuera de línea	Movimientos totales	Tiempo total
0	0s	14	50	9.7s
0	0s	6	42	11.1s
1	2.2s	0	44	13.8s
0	0s	0	26	7.1s
2	4.6s	23	74	15s

Tabla 30. Datos recogidos en las pruebas realizadas de seguir una línea recta delante del robot utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos en orientarse	0.6
Segundos en orientarse	1.4s
Movimientos fuera de línea	8.6
Movimientos totales	47.2
Porcentaje de movimientos fuera de línea	18.2%
Tiempo total	11.3s

Tabla 31. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot con el algoritmo de seguir una línea

En la **Figura 51** se puede observar el recorrido que ha trazado el robot en azul cuando estaba sobre la línea, en morado cuando estaba lejos de ella y los puntos rojos son la propia línea. Se ve que traza un recorrido en zigzag como ya se vio en el apartado 4.8.

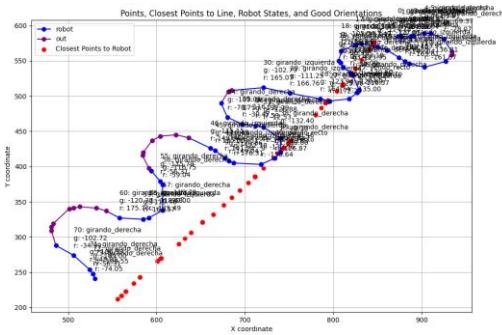


Figura 51. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta enfrente del robot utilizando el algoritmo de seguir una línea

Gemelo virtual

En este caso la **Tabla 32** no se indican movimientos ni segundos en orientarse ya que el gemelo virtual estaba bien colocado y alineado con la línea en todas las pruebas. Además, en las medias de la **Tabla 33** se aprecia que el algoritmo funciona a la perfección ya que, al no existir latencia, las instrucciones son ejecutadas al instante de mandarlas y el robot no se sale ni una sola vez de la línea como se ve en la **Figura 52**.

Movimientos fuera de línea	Movimientos totales	Segundos totales
0	30	7.8s
0	29	8.5s
0	22	6.8s
0	24	7.8s
0	30	8s

Tabla 32. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del gemelo virtual utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos fuera de línea	0
Movimientos totales	27
Porcentaje de movimientos fuera de línea	0%
Tiempo total	7.8s

Tabla 33. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot utilizando el algoritmo de seguir una línea

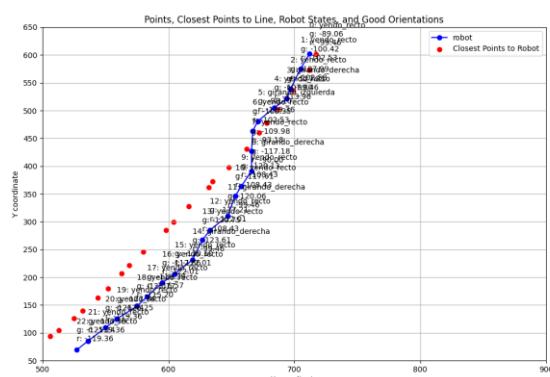


Figura 52. Recorrido del gemelo virtual en una de las pruebas realizadas de seguir una línea recta enfrente del gemelo utilizando el algoritmo de seguir una línea

Seguir línea volviendo a ella

En estas pruebas se cambia el algoritmo utilizado por el robot físico para seguir la línea, empleando en su lugar el algoritmo que detiene el robot y la vuelve a dirigir a ella cuando se ha desviado. En la **Tabla 34** se puede ver los datos de todas las pruebas. De nuevo no ha necesitado ni tiempo ni movimientos en llegar al inicio ni en orientarse por lo que no se indica. En las medias de la **Tabla 35**, se observa que la media de movimientos fuera de línea ha sido de 1 movimiento, lo que indica que el robot sólo se ha salido una vez en cada prueba de media y una vez que volvió y se alineó con ella, ha sido capaz de completar el recorrido sin volver a salirse.

El tiempo total en recorrer la línea ha sido del doble en comparación con el primer algoritmo, pero por el contrario ha recorrido la línea con mayor precisión. El tiempo medio ha sido de 22.3 segundos con un 2.4% de movimientos fuera de línea y con el primer algoritmo 11.3s segundos y 18.2% de movimientos fuera de línea.

Segundos totales volviendo a línea	Movimientos totales volviendo a línea	Segundos totales orientándose a la línea	Movimientos totales orientándose a la línea	Movimientos fuera de línea	Movimientos totales	Segundos totales
11.1s	5	0s	0	1	43	23.7s
7.4s	3	5.5s	2	1	56	25.9s
7.5s	3	0s	0	1	30	19s
7.7s	3	0s	0	1	48	20.9s
5.3s	2	5.4s	2	1	35	21.9s

Tabla 34. Datos recogidos de las pruebas realizadas de seguir una línea recta delante del robot físico utilizando el algoritmo de seguir una línea y volver a ella.

Medida	Media
Movimientos totales volviendo a la línea	3.2
Segundos totales volviendo a la línea	7.8s
Movimientos totales orientándose a la línea	0.8
Segundos totales orientándose a la línea	2.2s
Movimientos fuera de línea	1
Movimientos totales	42.4
Porcentaje de movimientos fuera de línea	2.4%
Segundos totales	22.3s

Tabla 35. Medias recogidas de las pruebas realizadas de seguir una línea recta delante del robot físico utilizando el algoritmo de seguir una línea y volver a ella

En la **Figura 53** se puede ver el recorrido que traza el robot en una de las pruebas. En el punto que está dentro del círculo de color verde se manda la instrucción de detenerse al robot porque se ha detectado que estaba lejos de la línea, como se explicó en el apartado 4.10.1 el robot sigue avanzando un poco hasta que le llega la orden por el problema de la latencia, y en el punto amarillo el robot comienza a redirigirse a la línea para después alinearse con ella de nuevo.

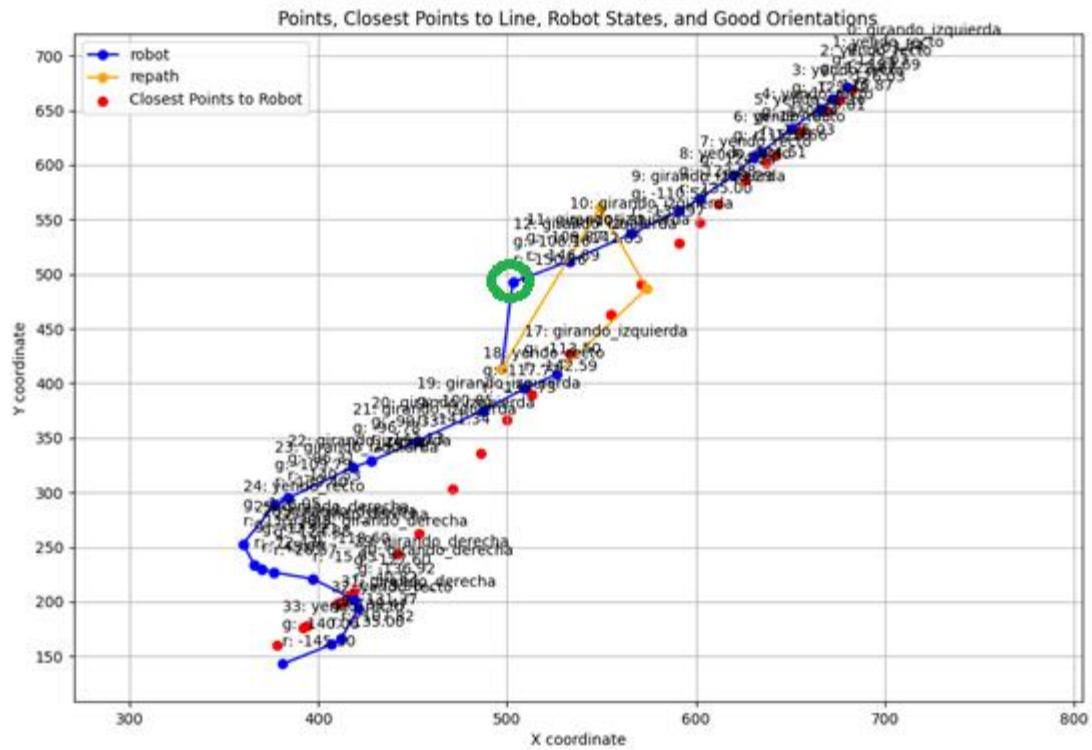


Figura 53. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta enfrente del robot utilizando el algoritmo de seguir una línea y volver a ella

5.2.2 Seguir una línea recta perpendicular al robot

Este experimento es similar al anterior, pero se le añade la dificultad de que el robot tiene que llegar al inicio de la línea como se observa en la **Figura 54**. Los 3 métodos planteados usan el mismo algoritmo para llegar al inicio, y una vez allí se evaluará el comportamiento del robot siguiendo la línea.

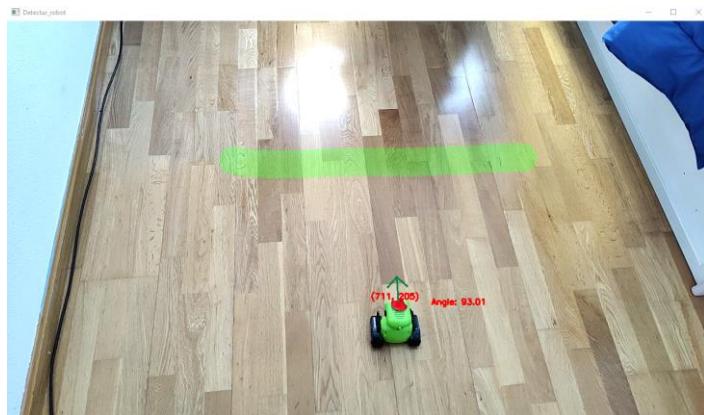


Figura 54. Situación de una línea recta perpendicular al robot

Algoritmo de seguir una línea

En los datos reflejados en la **Tabla 36**, se puede observar una clara diferencia con el primer experimento, y es que, en este, el robot ha necesitado tiempo y movimientos para llegar al inicio de la línea. Después de llegar al inicio, a excepción de la primera prueba,

el robot apenas se ha salido la línea con solo 0 o 1 movimiento fuera de línea. Esto da a entender que el robot se ha alineado con la línea de mejor forma que cuando se ha colocado de manera manual en el primer experimento porque una vez empieza a recorrer la línea no se ha salido de ella. Esto es debido al umbral de error que soporta el programa a la hora de alinearse con la línea ya que al colocarlo de manera manual el programa ha permitido que, aunque no estuviera perfectamente alineado, haya avanzado.

Movimientos en llegar al inicio	Segundos en llegar al inicio	Movimientos en orientarse	Segundos en orientarse	Movimientos fuera de línea	Movimientos totales	Segundos totales
8	18.3s	1	2.4s	17	62	29.9s
10	22.32s	3	6.83s	0	14	34.2s
7	15s	2	4.4s	1	36	26.5s
7	16.8s	2	4.6s	0	18	26.7s
8	18.6s	2	4.5s	0	43	30.5s

Tabla 36. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea

En cuanto a las medias de la **Tabla 37**, se observa que el tiempo es significativamente superior al primer experimento por una razón, y es que se ha añadido el tiempo en llegar al inicio de la línea. La media de tiempo es de 29.6 segundos y en el primer experimento con el primer algoritmo era de 11.3 segundos.

También es destacable que ha tardado 8 movimientos de media en llegar al inicio, lo cual son 3 movimientos más de lo que tardaba de media en llegar al punto objetivo en los primeros experimentos. Esto se debe a que ahora el umbral de distancia con el que el algoritmo detecta que el robot ha llegado al punto se ha reducido de 50 píxeles a 25 píxeles, ya que como el robot tiene que seguir la línea después de llegar al punto, se necesita que este lo mejor posicionado posible. Esta reducción del umbral hace que necesite más movimientos en llegar al punto objetivo, porque tiene que ajustarse con mayor precisión.

Medida	Media
Movimientos en llegar al inicio	8
Segundos en llegar al inicio	18.2s
Movimientos en orientarse	2
Segundos en orientarse	4.6s
Movimientos fuera de línea	3.6
Movimientos totales	34.6
Porcentaje fuera de línea	10.4%
Tiempo total	29.6s

Tabla 37. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea

En la **Figura 55** se muestra el recorrido que ha seguido el robot. Se observa que en sus primeros movimientos el robot comienza correctamente alineado y poco a poco se desvía hacia la derecha, en ese momento el algoritmo le envía las instrucciones de giro y a partir de ahí empieza a realizar zigzag por la latencia.

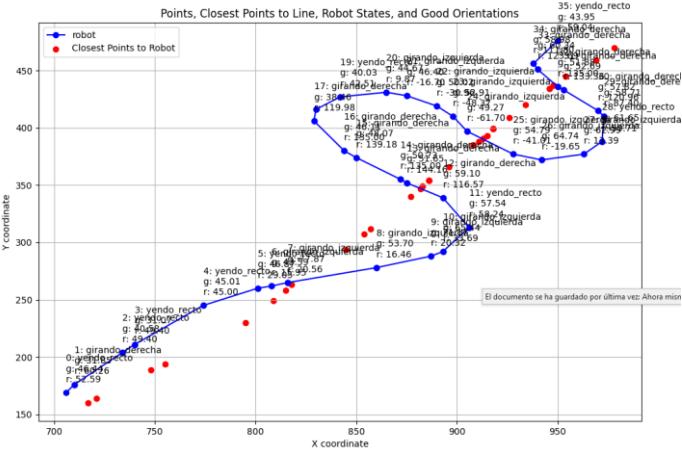


Figura 55. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta perpendicular al robot utilizando el algoritmo de seguir una línea

Gemelo virtual

En todas las pruebas realizadas reflejadas en la **Tabla 38**, el robot no se ha salido de la línea ni una sola vez lo que indica de nuevo que el algoritmo inicial funciona bien cuando no existe latencia. Lo único que varía respecto al primer experimento del apartado 5.2.1 es que se añade tiempo y movimientos que necesita en llegar al inicio. Comparándolo con el algoritmo de seguir una línea con el robot físico en este mismo experimento, se ve en las medias de la **Tabla 39** que es mucho más eficiente en recorrer la línea, ya que únicamente tiene 14.2 movimientos de media frente a los 34.6 movimientos de media en el robot físico. Esta cantidad de movimientos de más se debe a que el robot físico realiza los recorridos en zigzag por la latencia y en este caso como se observa en la **Figura 56** el gemelo virtual traza un recorrido prácticamente sobre la línea.

Movimientos en llegar al inicio	Segundos en llegar al inicio	Movimientos en orientarse	Segundos en orientarse	Movimientos fuera de línea	Movimientos totales	Segundos totales
4	7.1s	2	4.1s	0	15	17.4s
7	13.3s	1	2.3s	0	12	21s
4	7s	1	2.2s	0	16	15.1s
7	13.4s	2	4.3s	0	12	23.2s
6	12.2s	1	2.3s	0	16	20.4s

Tabla 38. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al gemelo virtual utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos en llegar al inicio	5.6
Segundos en llegar al inicio	10.6s
Movimientos en orientarse	1.4
Segundos en orientarse	3s
Movimientos fuera de línea	0
Movimientos totales	14.2
Porcentaje de movimientos fuera de línea	0%
Tiempo total	19.4s

Tabla 39. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al gemelo virtual utilizando el algoritmo de seguir una línea

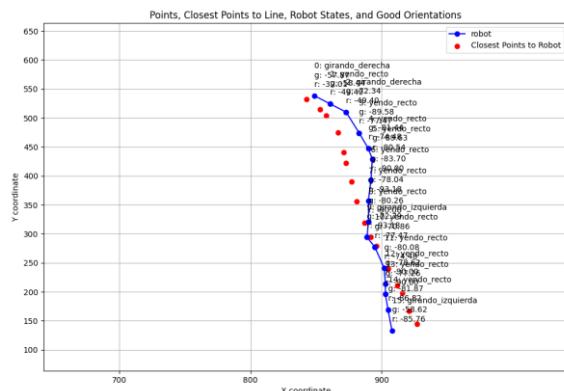


Figura 56. Recorrido del gemelo virtual en una de las pruebas realizadas de seguir una línea recta perpendicular al gemelo utilizando el algoritmo de seguir una línea

Algoritmo de seguir línea volviendo a ella

En este caso se cambia de nuevo al algoritmo de seguir una línea, detener y volver a ella cuando el robot se ha desviado. Se observan pocos cambios con respecto al primer algoritmo ya que las medias de la **Tabla 40** son similares a las recogidas en el primer algoritmo. En el primer algoritmo, se obtenía una media de 34.6 movimientos y 29.4 segundos, mientras que, con este algoritmo, la media fue de 35.6 movimientos y 35.1 segundos.

Aunque las medias de movimientos y tiempo son parecidas, se observa mayor variabilidad de estos datos en las pruebas realizadas, como se muestra en la **Tabla 39**, con resultados que van desde 23 movimientos y 18.6 segundos, hasta 47 movimientos y 50.6 segundos. En las pruebas donde el robot tiene 0 movimientos fuera de línea, el robot no ha tenido que detenerse y volver a la línea, aplicando el primer algoritmo durante todo el recorrido. En las pruebas donde se ha salido de la línea, el robot se ha detenido y ha vuelto a la línea, sumando más tiempo al recorrido.

Al igual que con el primer algoritmo, en este experimento como el robot se ha alineado automáticamente con la línea hay algunas pruebas en las que el robot no se ha salido en ningún momento de ella. El porcentaje de movimientos en los que el robot ha estado fuera de la línea es de 1.7% ya que al igual que en el primer experimento, tan solo se ha salido una vez de la línea como mucho a diferencia del 10.4% que se obtiene del primer experimento.

Segundos en llegar al inicio	Movimientos en llegar al inicio	Movimientos en orientarse	Segundos en orientarse	Segundos totales volviendo a línea	Movimientos totales volviendo a línea	Segundos totales orientándose a la línea	Movimientos totales orientándose a la línea	Movimientos fuera de línea	Movimientos totales	Segundos totales
8.4s	4	2	5.6s	0	0	0	0	0	23	18.6s
15.3s	8	3	8.1s	0	0	0	0	0	31	28.6s
9.8s	5	2	5.6s	4.9s	2	7.9s	3	1	33	37.5s
17.6s	8	2	5.5s	13.2s	5	3.2s	1	1	47	50.6s
12.2s	6	2	5.7s	7.1s	3	5.6s	2	1	44	40.1s

Tabla 40. Datos recogidos de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea volviendo a ella

Medida	Media
Movimientos en llegar al inicio	6.2
Segundos en llegar al inicio	12.7s
Movimientos en orientarse	2.2
Segundos en orientarse	6.1s
Movimientos totales volviendo a la línea	2
Segundos totales volviendo a la línea	5s
Movimientos totales orientándose a la línea	1.2
Segundos totales orientándose a la línea	3.3s
Movimientos fuera de línea	0.6
Porcentaje fuera de línea	1.7%
Movimientos totales	35.6
Segundos totales	35.1s

Tabla 41. Medias recogidas de las pruebas realizadas de seguir una línea recta perpendicular al robot físico utilizando el algoritmo de seguir una línea volviendo a ella

Como se muestra en la **Figura 57** el robot se sale de la línea y en amarillo aparece el movimiento del robot para volver a ella. En esta prueba se observa claramente las imprecisiones del robot para llegar al punto objetivo, que en ese momento era el punto más cercano de la línea.

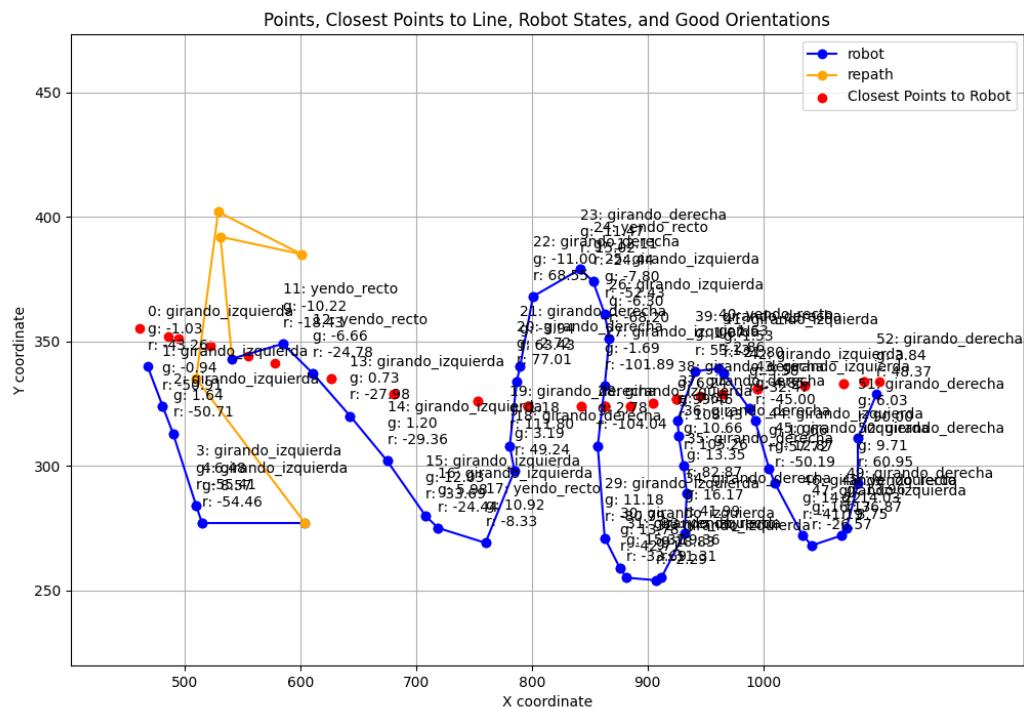


Figura 57. Recorrido del robot en una de las pruebas realizadas de seguir una línea recta perpendicular al robot utilizando el algoritmo de seguir una línea

5.3 Seguir una línea curva

Los siguientes experimentos serán relativos a líneas con curvas. Serán 3 experimentos en los que se complicará la línea a recorrer de menos a más. El primero será una línea con una curva abierta, el segundo con una curva cerrada y el tercero con dos curvas. De esta forma se verá cómo responde el robot ante diferentes situaciones.

5.3.1 Seguir una línea con una curva abierta

En la **Figura 58** se observa la situación del primer experimento, el robot sobre el inicio de la línea la cual tiene una curva abierta.

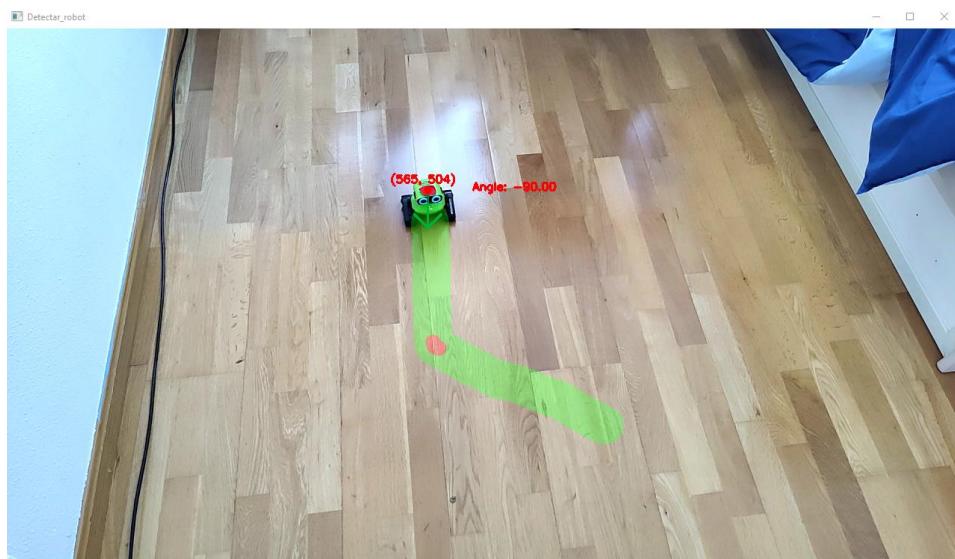


Figura 58. Situación de una línea con una curva abierta

Algoritmo de seguir línea

En este primer experimento con una curva abierta, se recogen datos relativos a 5 pruebas realizadas que se muestran en la **Tabla 42**. En una de las pruebas, el robot ha necesitado alinearse con la línea ya que al colocarlo manualmente no debía estar alineado correctamente. En la **Tabla 43** se observan las medias de estas pruebas. Un dato significativo es que el 23.1% de los movimientos han sido fuera de la línea, lo cual es un aumento respecto a los 18.2% y 10.4% de los dos primeros experimentos con este mismo algoritmo.

Movimientos en orientarse	Segundos en orientarse	Movimientos fuera de línea	Movimientos totales	Segundos totales
0	0	21	50	7.7s
1	2.2s	9	73	12.5s
0	0	29	116	15.2s
0	0	8	54	8.5s
0	0	10	40	7.5s

Tabla 42. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos en orientarse	0.2
Segundos en orientarse	0.4s
Movimientos fuera de línea	15.4
Porcentaje fuera de línea	23.1%
Movimientos totales	66.6
Tiempo total	10.3s

Tabla 43. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea

En la **Figura 59** se puede ver el recorrido que ha trazado el robot en una de las pruebas realizadas. En morado se ve que el robot se ha salido de la línea en la curva. Aunque el robot giró a la izquierda como debía, lo hizo muy tarde, una vez más, por el problema de la latencia.

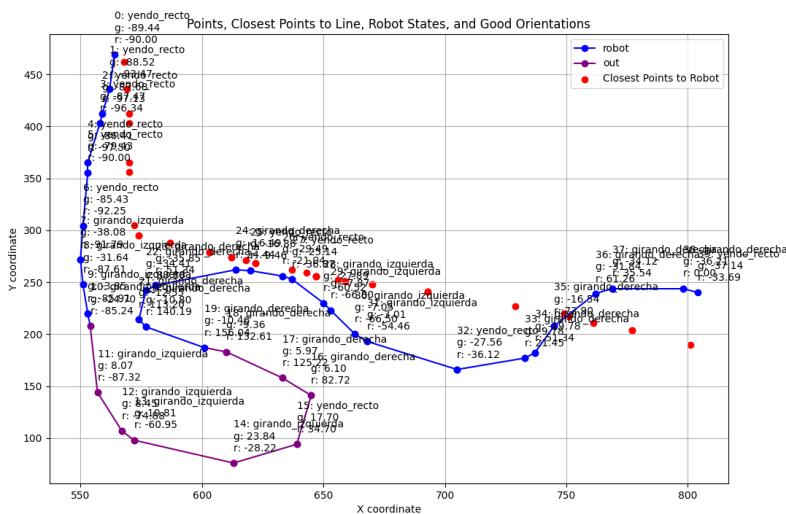


Figura 59. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea

Gemelo virtual

Nuevamente se comprueba con los datos y medias recogidas en la **Tabla 44** y la **Tabla 45** que el algoritmo funciona a la perfección, cuando se elimina la latencia en las instrucciones, aun habiendo simulado los errores de precisión en los movimientos del robot. El gemelo virtual no se sale de la línea en ningún momento en ninguna de las pruebas realizadas.

Movimientos fuera de línea	Movimientos totales	Segundos totales
0	19	6.7s
0	17	6.4s
0	15	5.8s
0	22	6.8s
0	21	7s

Tabla 44. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos fuera de línea	0
Porcentaje de movimientos fuera de línea	0%
Movimientos totales	18.8
Tiempo total	6.5s

Tabla 45. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea

En la **Figura 60** se muestra uno de los recorridos que ha trazado el gemelo virtual en una de las pruebas. Se puede observar que el gemelo traza bien el recorrido de la línea sin desviarse.

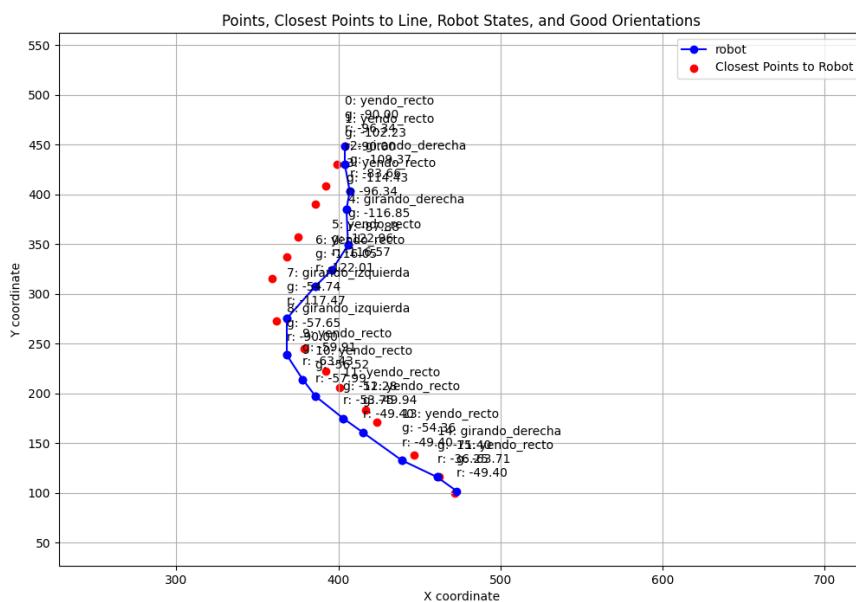


Figura 60. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva abierta con el gemelo virtual utilizando el algoritmo de seguir una línea

Algoritmo de seguir línea volviendo a ella

En este caso se vuelve a usar el robot físico con el algoritmo de seguir una línea volviendo a ella cuando se ha desviado. Los datos recogidos en la **Tabla 46** y las medias de la **Tabla 47**, indican que con este algoritmo se tarda más del doble de tiempo en recorrer la línea que con el primer algoritmo, 26.4 segundos frente a 10.3 segundos. Sin embargo, permanece en la línea mucho más tiempo. Al comparar este experimento con los de la línea recta se ve que cuando se desvía de la línea curva, tiene que emplear más tiempo y movimientos de media para alinearse correctamente con ella.

En los experimentos de la línea recta usando este algoritmo el robot empleaba 7.8 segundos y 5 segundos de media volviendo a la línea y 2.2 segundos y 3.3 segundos alineándose con ella. En este experimento el robot tarda 6.2 segundos en volver a la línea y 8.8 segundos en orientarse, sumando un total de 15 segundos para volver a la línea, lo que es más que en los anteriores experimentos.

Segundos totales volviendo a línea	Movimientos totales volviendo a línea	Segundos totales orientándose a la línea	Movimientos totales orientándose a la línea	Movimientos fuera de línea	Movimientos totales	Segundos totales
3.1s	1	7.8s	3	1	27	20.7s
15.4s	6	12.3s	5	1	30	38.1s
6.8s	3	14.5s	6	1	41	32.2s
5.9s	3	9.2s	3	2	60	32.8s
0	0	0	0	0	48	8.1s

Tabla 46. Datos recogidos de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea volviendo a ella

Medida	Media
Movimientos en llegar al inicio	-
Segundos en llegar al inicio	-
Movimientos en orientarse	-
Segundos en orientarse	-
Movimientos totales volviendo a la línea	2.6
Segundos totales volviendo a la línea	6.2s
Movimientos totales orientándose a la línea	3.4
Segundos totales orientándose a la línea	8.8s
Movimientos fuera de línea	1
Porcentaje de movimientos fuera de línea	2.4%
Movimientos totales	41.2
Segundos totales	26.4s

Tabla 47. Medias recogidas de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea volviendo a ella

En la **Figura 61** se observa uno de los recorridos trazados por el robot durante las pruebas. Se puede ver en amarillo que el robot se sale dos veces de la línea, pero, en este caso, vuelve rápidamente a ella. Hasta ahora, en ninguna de las pruebas realizadas, el robot se había salido dos veces de la línea, pero en este caso, con la curva abierta, si lo ha hecho.

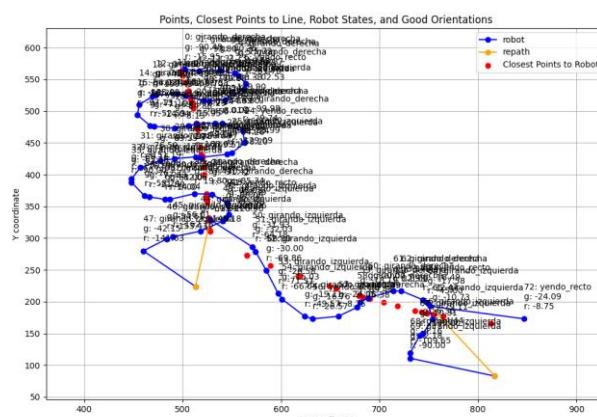


Figura 61. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva abierta con el robot utilizando el algoritmo de seguir una línea

5.3.2 Seguir una línea con una curva cerrada

Como se muestra en la **Figura 62**, en esta situación el robot debe seguir una línea con una curva más cerrada que el último experimento.

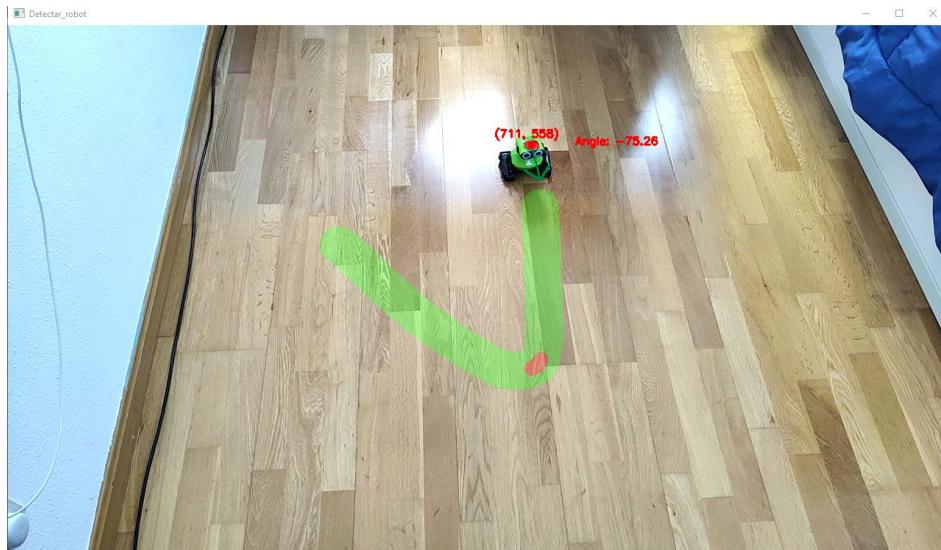


Figura 62. Situación de una línea con una curva cerrada

Algoritmo de seguir una línea

De nuevo se emplea el algoritmo de seguir una línea y se recogen los datos de las pruebas realizadas en la **Tabla 48**. Se observa que en las dos primeras pruebas el programa ha detectado que no estaba correctamente en el inicio de la línea ya que ha necesitado tiempo y movimientos en llegar al inicio. En la segunda prueba ha tardado bastante tiempo en colocarse correctamente, en el resto esto no ha sucedido. En cuanto a los movimientos totales que ha necesitado para seguir la línea, hay mucha variedad, siendo el mínimo 64 y el máximo 220. Esto también afecta al tiempo total, que varía desde 9.6 segundos hasta 27.9 segundos.

Movimientos en llegar al inicio	Segundos en llegar al inicio	Movimientos fuera de línea	Movimientos totales	Segundos totales
1	1.6s	9	72	12s
6	13.6s	5	116	27.9s
0	0s	5	64	9.6s
0	0s	88	220	20.2s
0	0s	11	85	11.7s

Tabla 48. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea

En las medias de la **Tabla 49** se observa un aumento significativo en el número de movimientos de media en recorrer la línea respecto al experimento de seguir una línea con una curva abierta aumentando de 66.6 movimientos a 111.4. Sin embargo, el porcentaje de movimientos fuera de línea se mantiene, 21.2% frente a 23.1% en la curva abierta.

Medida	Media
Movimientos en llegar al inicio	1.4
Segundos en llegar al inicio	3s
Movimientos fuera de línea	23.6
Porcentaje de movimientos fuera de línea	21.2%
Movimientos totales	111.4
Tiempo total	16.3s

Tabla 49. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea

Este incremento en el número de movimientos es debido a que la curva del robot es más pronunciada. Como se muestra en la **Figura 63**, los primeros zigzags que hace tras la curva son tan pronunciados que el robot apenas avanza. En otras figuras como la **Figura 55** o la **Figura 57**, se puede observar que los zigzags no son tan pronunciados, hasta el punto en el que, tras realizar los giros, el robot vuelve a un punto donde ya ha estado.

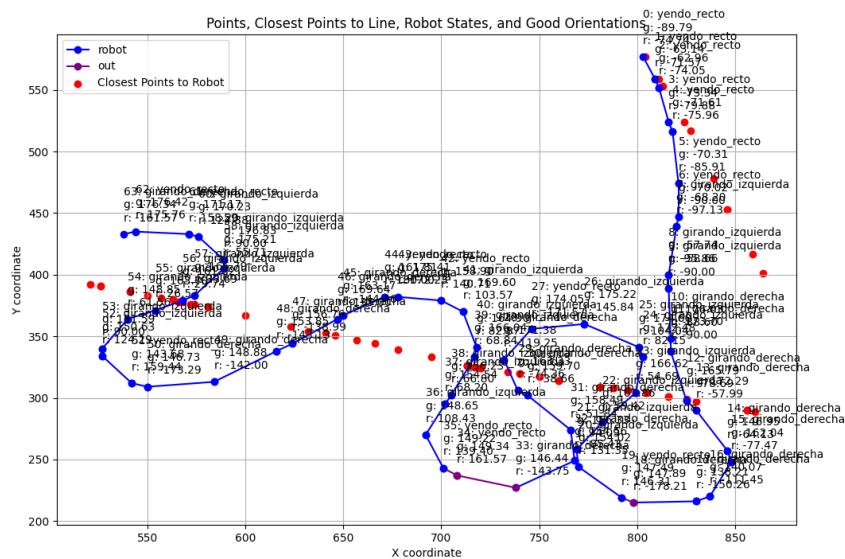


Figura 63. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea

Gemelo virtual

Una vez más, el gemelo virtual con el algoritmo de seguir una línea es impecable, y se mantiene en la línea todo el tiempo, como demuestran los datos y medias de la **Tabla 50** y la **Tabla 51**.

Movimientos fuera de línea	Movimientos totales	Segundos totales
0	28	8s
0	32	8.6s
0	30	8.3s
0	23	7s
0	27	7.6s

Tabla 50. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos fuera de línea	0
Porcentaje de movimientos fuera de línea	0%
Movimientos totales	28
Tiempo total	7.9s

Tabla 51. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea

En la **Figura 64** aparece el recorrido del gemelo virtual en una de las pruebas, en las que se observa que toma la curva correctamente manteniéndose en todo momento cerca de ella.

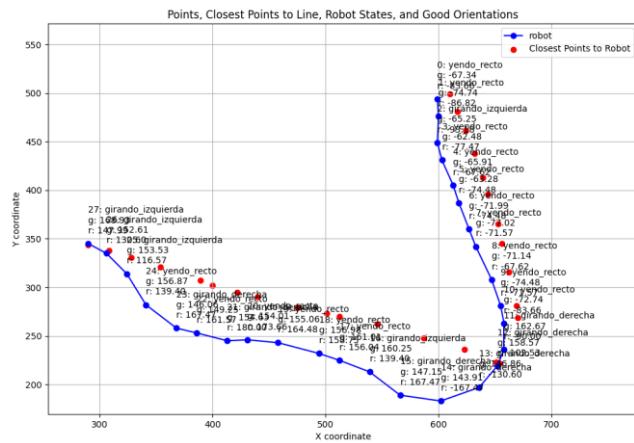


Figura 64. Recorrido del robot en una de las pruebas realizadas de una línea con una curva cerrada con el gemelo virtual utilizando el algoritmo de seguir una línea

Algoritmo de seguir línea volviendo a ella

En este caso en las pruebas reflejadas en la **Tabla 52**, se puede ver que en una de las pruebas el robot se ha salido 4 veces de la línea, ha necesitado 12 movimientos en total en volver a ella y 5 alineándose, lo que suma bastante tiempo. Sin embargo, en otras dos pruebas el robot ha conseguido seguir la línea sin desviarse, simplemente con el algoritmo original. A pesar de estas 3 pruebas con resultados extremos la media de tiempo total de la **Tabla 53** es 27.3 segundos, similar a las dos primeras pruebas. Y el porcentaje de movimientos fuera de línea es mínimo, un 3%.

En comparación con el primer algoritmo necesita más tiempo en recorrer la línea, de 16.3 segundos a 27.3 segundos. Pero el porcentaje de movimientos fuera de línea vuelve a ser muy inferior, de 21.2% a 3%.

Segundos totales volviendo a línea	Movimientos totales volviendo a línea	Segundos totales orientándose a la línea	Movimientos totales orientándose a la línea	Movimientos fuera de línea	Movimientos totales	Segundos totales
5.8s	3	9.1s	3	2	29	28.8s
9.5s	4	8.1s	3	1	42	29.1s
0	0	0	0	0	60	9.3s
0	0	0	0	0	59	9.4s
21.9s	12	15.9s	5	4	40	60.1s

Tabla 52. Datos recogidos de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella

Medida	Media
Movimientos totales volviendo a la línea	3.8
Segundos totales volviendo a la línea	7.4s
Movimientos totales orientándose a la línea	2.2
Segundos totales orientándose a la línea	6.6s
Movimientos fuera de línea	1.4
Porcentaje de movimientos fuera de línea	3%
Movimientos totales	46
Segundos totales	27.3s

Tabla 53. Medias recogidas de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella

En la **Figura 65** se muestra el recorrido de la prueba realizada en la que el robot se ha salido 4 veces de la línea. Se observa que a pesar de salirse de la línea las 4 veces el robot realiza un recorrido completo de la línea.

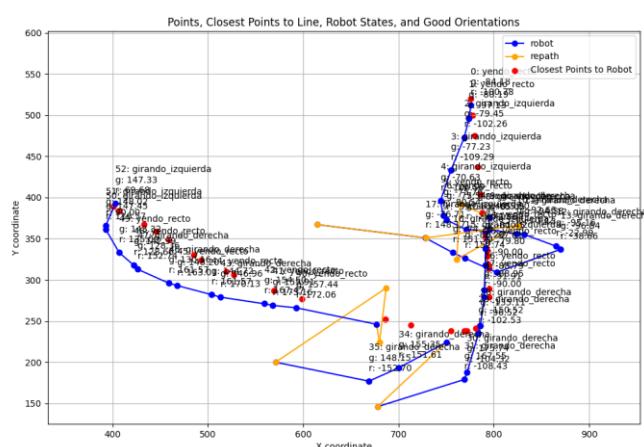


Figura 65. Recorrido del robot en una de las pruebas realizadas de seguir una línea con una curva cerrada con el robot utilizando el algoritmo de seguir una línea volviendo a ella

5.3.3 Seguir línea con dos curvas

En la **Figura 66** se muestre el experimento con la situación más complicada planteada. Una línea con dos curvas para ver cómo responde el robot.

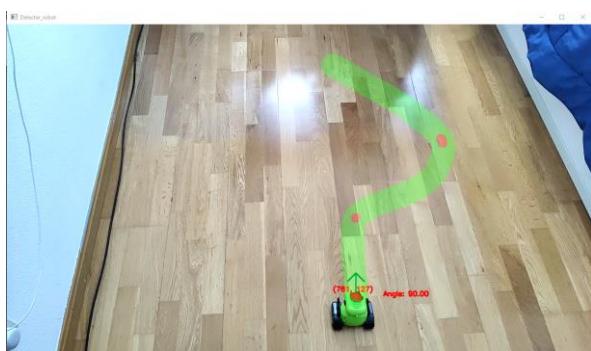


Figura 66. Situación de una línea con dos curvas

Algoritmo de seguir una línea

En estas primeras pruebas con el primer algoritmo en el robot físico recogidas en la **Tabla 54**, lo más destacable es el número de movimientos que necesita el robot para llegar al final, ya que a pesar de que la línea es algo que con solo una curva el número de movimientos respecto al anterior experimento aumenta de 111.4 movimientos a 165.6. Esto sucede porque el robot recorre la línea en zigzag como se muestra en la **Figura 67** y aunque la línea no sea mucho más larga, al no recorrerla en línea recta, aumentan mucho los movimientos.

En cuanto a las medias de la **Tabla 55** se puede ver que el porcentaje de movimientos fuera de línea es del 20.7%, por lo que se mantiene en torno al 20% como en los dos experimentos anteriores.

Movimientos fuera de línea	Movimientos totales	Segundos totales
9	121	14.8s
42	190	21.4s
49	164	18.5s
41	201	22.1s
30	153	17.8s

Tabla 54. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos fuera de línea	34.2
Porcentaje de movimientos fuera de línea	20.7%
Movimientos totales	165.6
Tiempo total	18.9s

Tabla 55. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea

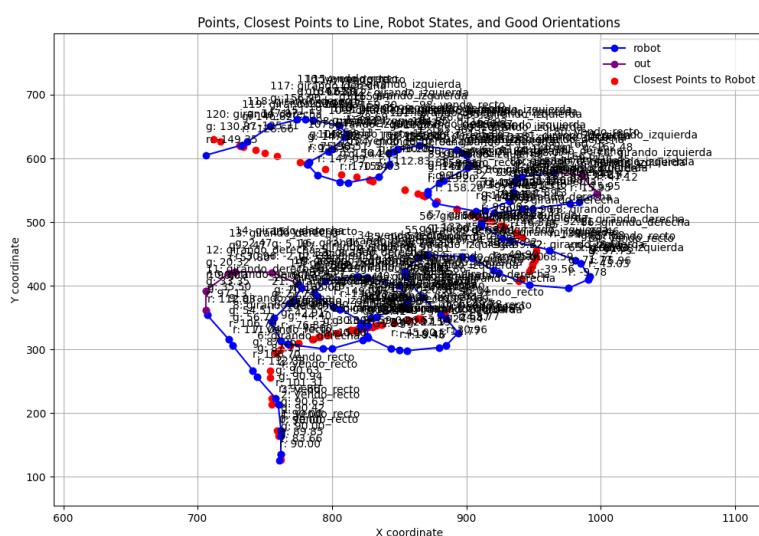


Figura 67. Recorrido del robot en una de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea

Gemelo virtual

El gemelo virtual con el algoritmo original sigue respondiendo perfectamente ante la esta situación y como se observa en la **Tabla 56** y la **Tabla 57** no se desvía en ningún momento.

Movimientos fuera de línea	Movimientos totales	Segundos totales
0	37	9.35s
0	41	10.5s
0	30	8.1s
0	34	8.2s
0	29	8.2s

Tabla 56. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea

Medida	Media
Movimientos fuera de línea	0
Movimientos totales	34.2
Tiempo total	8.9s

Tabla 57. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea

En la **Figura 68** se muestra el recorrido trazado por el gemelo virtual recorriendo la línea con dos curvas. Como se ve la sigue con precisión.

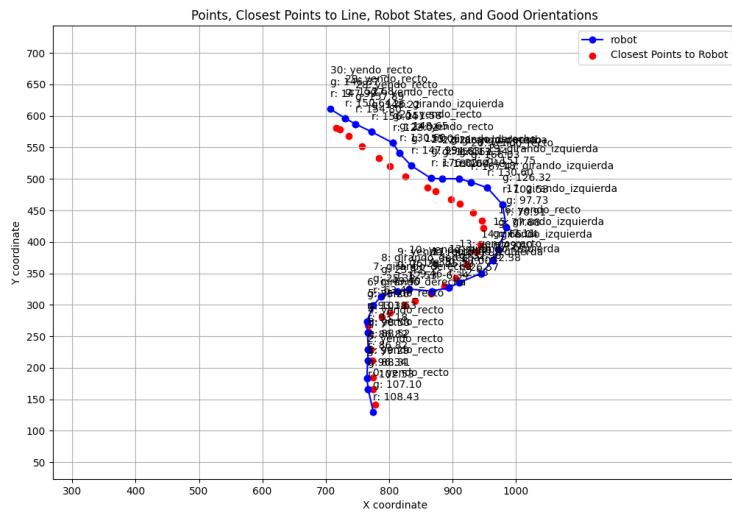


Figura 68. Recorrido del robot en una de las pruebas realizadas de seguir una línea con dos curvas con el gemelo virtual utilizando el algoritmo de seguir una línea

Algoritmo de seguir línea volviendo a ella

En este caso en los datos recogidos en la **Tabla 58** se observa que, en comparación con el primer algoritmo necesita muchos menos movimientos para llegar a la línea, únicamente 61 de media frente a los 165.4 del primer algoritmo. Sin embargo, pasa de 18.9 segundos a 48.9 segundos con este algoritmo. Esto sucede porque a diferencia de los primeros experimentos en los que usando este algoritmo el robot se salía apenas 1 vez de

media, en este experimento el robot se sale 2.6 veces de media como se muestra en la **Tabla 59**. Esto se debe a que al haber dos curvas el robot se desvía con mayor facilidad. De todas formas, el robot sigue teniendo un porcentaje muy bajo de movimientos fuera de línea, un 4.3%.

También se observa que, en comparación con el experimento de seguir una línea con una curva cerrada, se incrementan casi en el doble los movimientos en volver a la línea y en orientarse con ella. Esto se debe a que el robot se sale 2.6 veces de media y en el anterior experimento se salía 1.4 veces. Se pasa de 3.8 movimientos en volver a la línea y 2.2 movimientos en orientarse, a 7.8 movimientos en volver a la línea y 4.8 movimientos en orientarse. Por lo que los movimientos en volver a la línea y en orientarse con ella siguen una relación con el número de veces que se sale de la línea.

Movimientos en orientarse	Segundos en orientarse	Segundos totales volviendo a línea	Movimientos totales volviendo a línea	Segundos totales orientándose a la línea	Movimientos totales orientándose a la línea	Movimientos fuera de línea	Movimientos totales	Segundos totales
-	-	7.5s	4	9s	3	2	38	31.4s
-	-	15.1s	8	11.2s	4	2	70	43.9s
-	-	10.6s	5	13.6s	5	2	67	42.2s
2	5,5s	23.3s	12	14.7s	5	3	89	65s
-	-	18.9s	10	20.6s	7	4	41	62.1s

Tabla 58. Datos recogidos de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea volviendo a ella

Medida	Media
Movimientos en llegar al inicio	-
Segundos en llegar al inicio	-
Movimientos en orientarse	0.4
Segundos en orientarse	1.1s
Movimientos totales volviendo a la línea	7.8
Segundos totales volviendo a la línea	15.1s
Movimientos totales orientándose a la línea	4.8
Segundos totales orientándose a la línea	13.8s
Movimientos fuera de línea	2.6
Porcentaje de movimientos fuera de línea	4.3%
Movimientos totales	61
Segundos totales	48.9s

Tabla 59. Medias recogidas de las pruebas realizadas de seguir una línea con dos curvas con el robot utilizando el algoritmo de seguir una línea volviendo a ella

En la **Figura 69** se muestra el recorrido del robot siguiendo la línea en una de las pruebas. Se observa que se eliminan los recorridos en zigzag y que se recorre la línea con mayor precisión.

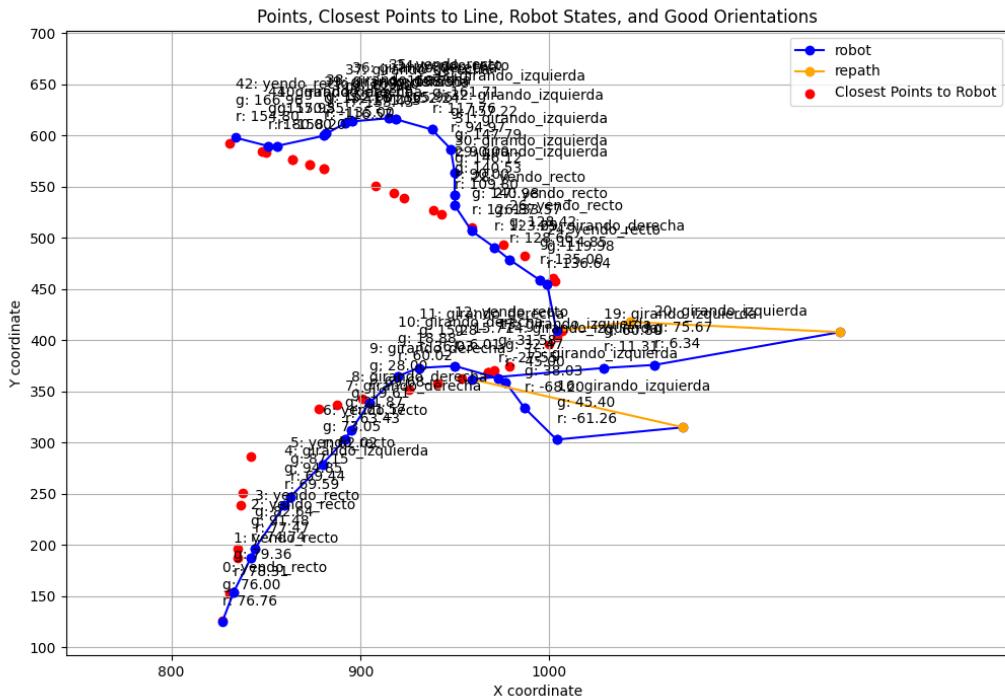


Figura 69. Recorrido del robot en una de las pruebas realizadas de seguir una linea con dos curvas con el robot utilizando el algoritmo de seguir una linea volviendo a ella

6. Conclusiones y trabajos futuros

El presente trabajo de fin de grado ha abordado el desarrollo y la implementación de una plataforma autónoma que, utilizando visión artificial, es capaz de moverse hacia un punto y seguir una línea. Para ello, se han desarrollado dos algoritmos que funcionan con retroalimentación visual.

En el montaje del hardware no se encontraron complicaciones, sin embargo, en el desarrollo de los algoritmos, el principal problema que se encontró fue el de la latencia, ya que con el hardware disponible no se podía eliminar. A pesar de ello se plantearon varias soluciones, pero ninguna con éxito total.

El algoritmo de llegada a un punto cualquiera funciona correctamente ya que los resultados de los experimentos realizados demuestran que el robot ha conseguido llegar al punto en 5 movimientos de media en 3 situaciones diferentes, con el punto situado a la misma distancia siempre, pero en distintas posiciones respecto a la orientación del robot. Esto demuestra que el programa es capaz de detectar correctamente la orientación del robot y el ángulo del robot al punto, para así enviar las instrucciones adecuadas para que el robot llegue al punto correctamente, independientemente de su posición inicial. Este algoritmo funciona bien ya que el robot se detiene entre instrucción e instrucción, por lo que no afecta la latencia.

En cuanto al algoritmo de seguir una línea en el robot físico, la latencia estuvo presente en todo momento, ya que en este caso el robot no debía detenerse entre instrucción e instrucción, lo que ha dificultado mucho el correcto funcionamiento del algoritmo.

Para intentar solucionar el problema de la latencia, se plantearon diferentes soluciones. Una opción fue un algoritmo que detectara cuando el robot estuviera fuera de la línea, deteniéndolo en ese momento para que volviera a dirigirse hacia ella para continuar con el recorrido. Otra solución propuesta fue un algoritmo capaz de estimar la posición del robot en la que va a recibir la instrucción tras la latencia pero que no funcionó correctamente ya que dependía de muchas más variables de las que se había tenido en cuenta en un principio. También se intentó ralentizar el robot en las curvas para que las tomara mejor, pero se descartó esta opción ya que se volvieron a encontrar los mismos problemas que con el algoritmo original.

En los experimentos de seguir una línea se probaron el algoritmo de seguir una línea tanto en el robot físico como en el gemelo virtual, y el algoritmo de seguir una línea volviendo a ella si se desvíá.

Se realizaron 5 experimentos con distintos tipos de líneas y distintas situaciones del robot para evaluar la eficiencia de ambos algoritmos. Con el primer algoritmo el robot en 4 de los 5 experimentos el robot permanece fuera de la línea mientras traza el recorrido alrededor de un 20%. En el experimento restante disminuye a un 10%, esto es debido a que en este experimento el robot tenía que dirigirse hacia la línea y una vez llegaba a ella se alineaba con ella automáticamente, en los otros 4 el robot era colocado manualmente al inicio de la línea. Debido al umbral de error permitido, en los experimentos en los que

se colocaba manualmente el robot, este avanzaba sin estar perfectamente alineado con la línea, en el otro experimento al alinearse automáticamente, a pesar de existir ese umbral de error con las orientaciones el robot se alineaba de mejor manera. Este umbral de error no se puede disminuir porque, si no en muchas ocasiones en las que se alinea automáticamente, tardaría demasiado en orientarse correctamente y se acabaría saliendo de la línea antes de empezar.

El gemelo virtual es un programa desarrollado aparte que simula el robot como un punto, en el que están implementados todos sus movimientos y todos los errores de precisión que comete el robot al desplazarse o girar. Este gemelo ha sido probado con los 5 experimentos de seguir la línea con el algoritmo de seguir una línea. En los 5 experimentos en lo que se ha ido añadiendo dificultad a la línea para ser recorrida, el gemelo virtual ha permanecido en ella el 100% de movimientos realizados, por lo que el algoritmo planteado funciona perfectamente si se eliminara la latencia de envío de instrucciones entre el ordenador y el robot físico.

También se ha probado el algoritmo de seguir una línea y volver a ella cuando el robot se desvía en los 5 experimentos. Los resultados muestran que en comparación con el primer algoritmo el robot permanece mucho más tiempo en la línea, todas las pruebas por debajo del 5%. Pero, por el contrario, el robot tarda de media aproximadamente el doble de tiempo en recorrer la línea, ya que se suma mucho tiempo al detenerse, volver a la línea y alinearse con ella.

Como trabajos futuros, se propone modificar el hardware de la plataforma para eliminar el mando como intermediario entre el ordenador y el robot. Para ello, se sugiere integrar un receptor bluetooth directamente en el robot, para así tener dos conexiones menos que harán que, en caso de existir latencia sea mucho menor. De este modo, se elimina la conexión del ordenador al gpio, del gpio al mando y del mando al robot, enviando directamente las instrucciones del ordenador al robot.

Referencias

- [1] Velázquez Cruz, E. (2011). *Robot seguidor de línea con retroalimentación visual* [Reporte final de residencia profesional]. Instituto Tecnológico de Tuxtla Gutiérrez. Disponible en
<http://repositoriodigital.tuxtla.tecnm.mx/xmlui/bitstream/handle/123456789/2786/MDRPIEL2011030.pdf?sequence=1&isAllowed=y> (tecnm.mx)
- [2] Peña Uriarte, D. (2019). *Automóvil teledirigido mediante microcontrolador Arduino* [Trabajo Fin de Grado]. Universidad del País Vasco. Disponible en
<http://repositoriodigital.tuxtla.tecnm.mx/xmlui/bitstream/handle/123456789/2786/MDRPIEL2011030.pdf?sequence=1&isAllowed=y>
- [3] García Villanueva, M., Ramírez Zavala, S., & Ortega Reyes, H. (2016). *Robot móvil autónomo seguidor de línea con Raspberry Pi y sistema de visión artificial*. Universidad Michoacana de San Nicolás de Hidalgo. Disponible en
https://www.researchgate.net/publication/312489611_Robot_Movil_Autonomo_Seguidor_de_Linea_con_Raspberry_Pi_y_Sistema_de_Vision_Artificial
- [4] Tosini, M., Acosta, N., Aciti, C., & Iarar, S. (2004). *Identificación por hardware de posición de jugadores en futbol de robots*. INTIA/INCA - Facultad de Ciencias Exactas - Universidad Nacional del Centro de la Provincia de Buenos Aires. Disponible en
https://sedici.unlp.edu.ar/bitstream/handle/10915/21316/Documento_completo.pdf?sequence=1&isAllowed=y
- [5] Bohórquez Pinargote, T.A, & Martillo García, D.J. (2023). *Diseño e implementación de un robot seguidor de línea mediante visión artificial* {Trabajo de titulación}. Universidad Politécnica Salesiana. Disponible en
<https://dspace.ups.edu.ec/bitstream/123456789/24933/1/UPS-GT004360.pdf>
- [6] García Villanueva, M., Ramírez Zavala, S., & Ortega Reyes, H. (2016). *Robot móvil autónomo seguidor de línea con Raspberry Pi y sistema de visión artificial*. Universidad Michoacana de San Nicolás de Hidalgo. Disponible en
https://www.researchgate.net/publication/312489611_Robot_Movil_Autonomo_Seguidor_de_Linea_con_Raspberry_Pi_y_Sistema_de_Vision_Artificial
- [7] Velázquez Cruz, E. (2011). *Robot seguidor de línea con retroalimentación visual* [Reporte final de residencia profesional]. Instituto Tecnológico de Tuxtla Gutiérrez. Disponible en
<http://repositoriodigital.tuxtla.tecnm.mx/xmlui/bitstream/handle/123456789/2786/MDRPIEL2011030.pdf?sequence=1&isAllowed=y> (tecnm.mx)
- [8] Sánchez López, J. F. (2013). *Aplicación para Android para el control de un vehículo a través de un trazo* [Proyecto Fin de Carrera, Universidad Politécnica de Cartagena]. Disponible en <http://repositorio.upct.es/entities/publication/9b7829da-c97f-46bc-a617-38791f52fb9f>

- [9] Galeano, P. A., Torres, I. D., & Álvarez, J. F. (2014). *Robot móvil autónomo seguidor de línea*. Investigación e Innovación en Ingeniería, 2(3), 56-62. Universidad Simón Bolívar. Disponible en
<https://revistas.unisimon.edu.co/index.php/innovacioning/article/view/2058/1950>
- [10] Velázquez Cruz, E. (2011). *Robot seguidor de línea con retroalimentación visual* [Reporte final de residencia profesional]. Instituto Tecnológico de Tuxtla Gutiérrez. Disponible en
[http://repositoriodigital.tuxtla.tecnm.mx/xmlui/bitstream/handle/123456789/2786/MDRPIEL2011030.pdf?sequence=1&isAllowed=yMDRPIEL2011030.pdf \(tecnm.mx\)](http://repositoriodigital.tuxtla.tecnm.mx/xmlui/bitstream/handle/123456789/2786/MDRPIEL2011030.pdf?sequence=1&isAllowed=yMDRPIEL2011030.pdf (tecnm.mx))
- [11] DroidCam by Dev47Apps - Official Website. (s. f.-d). <https://droidcam.app/>
- [12] Industries, A. (s. f.). Adafruit FT232H Breakout - General Purpose USB to GPIO, SPI, I2C. <https://www.adafruit.com/product/2264>
- [13] Remote Control Robot 2.4GHz RC Robot Toy for Kids - GILOBABY. (s. f.). GILOBABY. <https://gilobaby.com/collections/on-sale/products/remote-control-robot-2-4ghz-rc-robot-toy-for-kids-gilobaby?variant=40346220331190>
- [14] CircuitPython Libraries on any Computer with FT232H. (2019, 29 septiembre). Adafruit Learning System. <https://learn.adafruit.com/circuitpython-on-any-computer-with-ft232h/windows>
- [15] digitalio – Basic digital pin support — Adafruit CircuitPython 9.1.0-beta.3 documentation. (s. f.). <https://docs.circuitpython.org/en/latest/shared-bindings/digitalio/index.html>
- [16] OpenCV: OpenCV modules. (s. f.). <https://docs.opencv.org/3.4/index.html>
- [17] OpenCV: OpenCV modules. (s. f.). <https://docs.opencv.org/3.4/index.html>