

Universidad Carlos III de Madrid  
Escuela Politécnica Superior



Grado en Ingeniería Electrónica Industrial y Automática

**Trabajo de Fin de Grado**

**Control para Navegación de Robot por  
Medio de Visión Artificial**

Autor: Samuel Hernández Bermejo

Tutor: Enrique Pelayo Campillos



**Título:** Control para Navegación de Robot por Medio de Visión Artificial

**Autor:** Samuel Hernández Bermejo

**Tutor:** Enrique Pelayo Campillos

**TRIBUNAL CALIFICADOR**

**Presidente**

**Dr. D** \_\_\_\_\_

**Secretario**

**Dr. D** \_\_\_\_\_

**Vocal**

**Dr. D** \_\_\_\_\_

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el día \_\_\_\_\_  
de \_\_\_\_\_ de 2017 en Leganés, en la Escuela Politécnica Superior de la  
Universidad Carlos III de Madrid, acuerda otorgarse la CALIFICACIÓN de:



## AGRADECIMIENTOS

A todas las personas que me han apoyado y que han confiado en mí durante todo mi periodo universitario.

A mi familia por la ayuda, apoyo y paciencia de cada uno de ellos en los momentos difíciles, gracias a ellos he podido llegar hasta aquí. Les agradezco a todos su ánimo, aguante y su tiempo dedicado a que todo esto fuese un poco más fácil. Especialmente, a mi tío Pedro, que ha confiado en mí durante muchos años, me ha enseñado, y me ha dado muchas facilidades para trabajar y poder realizar mi carrera al mismo tiempo.

A mi tutor Enrique, por transmitirme iniciativa y entusiasmo por este proyecto y, desde su comienzo, apoyarme en la realización del mismo. También le agradezco que haya sabido gestionar mi forma de trabajar y haya sabido guiarme para finalmente conseguir el objetivo.

A mis compañeros de carrera Javi, Iván, Jesus y Bea, por su ayuda, paciencia y buenos momentos durante todos estos años, y por la confianza que me han dado para poder llevar a cabo la realización de este Trabajo de Fin de Grado.



## RESUMEN

En la actualidad, las nuevas tecnologías se están aplicando a la hora de mejorar la conducción de vehículos. Existe tecnología simple, como sensores y cámaras, que nos ayudan a aparcar, hasta los ya conocidos coches de conducción autónoma. Con estos nuevos sistemas de conducción se busca mayor confort y seguridad para el conductor y para los peatones.

El objetivo de este trabajo es el desarrollo de un sistema de conducción autónomo guiado por visión artificial. El sistema estará integrado por un robot con cámara y una parte software que, a partir de las imágenes obtenidas por la cámara, guiará a un robot entre las líneas de un carril bici (simulando un vehículo autónomo en la carretera).

El robot estará compuesto por una cámara, un chasis con cuatro motores y sus respectivos controladores de motores, todo ello integrado en una *Raspberry Pi*. La *Raspberry Pi* soportará un software desarrollado en *Python* y realizado con la ayuda de las librerías *OpenCV*, creadas para el desarrollo de la visión artificial, que cuentan con un gran número de funciones que se adecúan al objetivo.

Una vez perfeccionado el proyecto y desarrollando esta tecnología lo suficiente, podría utilizarse en otro tipo de vehículos de menor tamaño y potencia que un automóvil y en espacios menos abiertos que una carretera, como podrían ser sillas de ruedas motorizadas de personas que no tienen la capacidad para controlarla en la calle o en un hospital. Otra opción podría ser guiar vehículos de menor tamaño dentro de una empresa que transporten a sus trabajadores, transportar viajeros por un aeropuerto o incluso por un campus de una Universidad. Las posibilidades son muchas debido a su bajo coste y su gran rendimiento.



## ABSTRACT

Nowadays, new technologies are been applied to improve vehicle driving. There are simple technologies, such as sensors and cameras, which help us to park, even the well-known self-driving vehicles. These new autonomous driving systems look for comfort and safety for the driver and pedestrians.

The aim of the work is the development of a computer vision-guided autonomous driving system. It will be composed of a robot with camera and software which will guided the robot between the bicycle lane lines from the frames supplied by the camera (like an autonomous vehicle on the road).

The robot will be composed of a camera, a chassis, four DC motors and their respective motor drivers, all of it is supported by a *Raspberry Pi* with a *Python* software and the *OpenCV* library, which is created for computer vision development. This library has many functions to achieve the aim.

When this project has been improved and this technologies has been enoughly developed, it would be used in friendly environments with any type of less powerfull and smaller vehicle than a car, for example to guide a motorized wheelchair of disabled people in a hospital. Another option would be to move passengers at the airport or students in a university campus. There are so many possibilities due to its low cost and high performance.



## CONTENIDO

<b>RESUMEN .....</b>	<b>7</b>
<b>ABSTRACT .....</b>	<b>9</b>
<b>CAPÍTULO 1. INTRODUCCIÓN.....</b>	<b>16</b>
1.1. MOTIVACIÓN.....	18
1.2. OBJETIVO.....	19
1.3. ESTRUCTURA DEL TRABAJO.....	19
<b>CAPÍTULO 2. ESTADO DE LA TÉCNICA.....</b>	<b>21</b>
2.1. NAVEGACIÓN AUTÓNOMA.....	21
2.1.1. TÉCNICAS Y MÉTODOS CLAVES.....	21
2.1.2. APLICACIONES ACTUALES.....	22
2.1.2.1. AÉREAS.....	22
2.1.2.2. ACUÁTICOS.....	23
2.1.2.3. TERRESTRES.....	23
2.2. VEHÍCULOS DE CONDUCCIÓN AUTÓNOMA TERRESTRE.....	24
2.2.1. NIVELES Y EVOLUCIÓN.....	24
2.2.2. TECNOLOGÍA NECESARIA.....	26
2.2.2.1. INTELIGENCIA ARTIFICIAL.....	28
2.2.2.2. VISIÓN POR COMPUTADOR.....	30
2.2.2.3. MICROCONTROLADOR.....	33
2.2.3. DISEÑOS DE VEHÍCULOS AUTÓNOMOS.....	36
2.2.3.1. VEHÍCULOS DE CORTA DISTANCIA.....	36
2.2.3.2. VEHÍCULOS DE MEDIA DISTANCIA.....	37
2.2.3.3. VEHÍCULOS MULTIUSO.....	41
2.2.4. FUTURO DE LA TECNOLOGÍA.....	44
2.3. APLICACIONES Y TECNOLOGÍAS SIMILARES.....	45
2.3.1. ROBOT SIGUE LINEA INFRARROJOS.....	45
2.3.2. ROBOT SIGUE LÍNEA POR CÁMARA.....	45
2.3.3. ROBOT CONTROLADO CON SISTEMA ANDROID.....	46
2.3.4. COCHE AUTÓNOMO RC.....	46
2.3.5. SISTEMA DE DETECCIÓN DE SEÑALES DE TRÁFICO (TSR).....	47
2.3.6. LDW Y LKS.....	47
2.3.7. OTRAS TECNOLOGÍAS.....	49
<b>CAPÍTULO 3. SISTEMA PROUESTO.....</b>	<b>50</b>
3.1. DESCRIPCIÓN GENERAL.....	50
3.2. ARQUITECTURA HARDWARE.....	53
3.2.1. RASPBERRY PI CAMERA BOARD v2.....	53
3.2.2. RASPBERRY PI 3 MODEL B.....	54
3.2.3. MÓDULO L298N.....	56
3.2.4. CHASIS Y MOTORES.....	57
3.2.5. SISTEMA DE ALIMENTACIÓN.....	59
3.3. ARQUITECTURA SOFTWARE.....	60
3.3.1. HERRAMIENTAS SOFTWARE.....	60
3.3.1.1. SISTEMA OPERATIVO RASPBIAN.....	60
3.3.1.2. ANACONDA Y PYTHON.....	61
3.3.1.3. OPENCV.....	61

3.3.2. DESARROLLO DE LA APLICACIÓN .....	62
3.3.2.1. CONVERSIÓN DE RGB A HSV.....	65
3.3.2.2. SELECCIÓN DE REGIÓN DE INTERÉS.....	66
3.3.2.3. FILTRAR RUIDO .....	67
3.3.2.4. SUAVIZAR CONTORNOS.....	68
3.3.2.5. FILTRAR COLOR BLANCO.....	69
3.3.2.6. FILTRO CANNY. ....	70
3.3.2.7. TRANSFORMADA DE HOUGH.....	71
3.3.2.8. CONTROL DE DIRECCIÓN. ....	73
<b>CAPÍTULO 4. EVALUACIÓN DEL SISTEMA, PRUEBAS Y LIMITACIONES.....</b>	<b>74</b>
4.1. DETERMINACIÓN DE LOS VALORES DE FILTRADO. ....	74
4.1.1. FILTRADO DE LA REGIÓN DE INTERÉS.....	74
4.1.2. FILTRADO POR COLOR BLANCO. ....	75
4.1.3. FILTRADO DE LAS LÍNEAS. ....	76
4.2. DETERMINACIÓN DE LOS PARAMETROS DE LAS FUNCIONES.....	78
4.2.1. FUNCIONES DE FILTRADO DE RUIDO. ....	78
4.2.2. FUNCIONES DE CONTORNO. ....	80
4.3. PRUEBAS Y FUNCIONAMIENTO. ....	82
4.3.1. FUNCIONAMIENTO DEL SOFTWARE DE VISIÓN POR COMPUTADOR. ....	82
4.3.2. FUNCIONAMIENTO CONTROL DE MOTORES. ....	85
4.3.3. FUNCIONAMIENTO GLOBAL. ....	86
4.4. LIMITACIONES Y ERRORES. ....	91
<b>CAPÍTULO 5. PLANIFICACION. ....</b>	<b>94</b>
<b>CAPÍTULO 6. MARCO REGULADOR. ....</b>	<b>97</b>
<b>CAPÍTULO 7. ENTORNO SOCIO-ECONÓMICO.....</b>	<b>99</b>
7.1 PRESUPUESTO. ....	99
7.1.1. COSTE DE PERSONAL.....	99
7.1.2. COSTE DE SOFTWARE Y HARDWARE.....	100
7.1.3. COSTES INDIRECTOS Y AMORTIZACIONES. ....	101
7.1.4. COSTE TOTAL. ....	101
7.2 IMPACTO SOCIO-ECONÓMICO. ....	102
7.2.1. COSTES Y BENEFICIOS DE LA EMPRESA. ....	103
<b>CAPÍTULO 8. LÍNEAS FUTURAS. ....</b>	<b>104</b>
8.1 MEJORAS DE LA CAPACIDAD DE PROCESAMIENTO. ....	104
8.2 MEJORAS VISIÓN POR COMPUTADOR.....	104
8.3 MEJORAS HARDWARE. ....	105
8.4 MEJORAS SOFTWARE. ....	105
8.5 MEJORAS NAVEGACIÓN. ....	106
8.6 OPEN SOURCE.....	106
<b>CAPÍTULO 9. CONCLUSIONES. ....</b>	<b>107</b>
<b>CAPÍTULO 10. BIBLIOGRAFÍA.....</b>	<b>108</b>
ANEXOS A LA MEMORIA .....	112
ANEXO I: CÓDIGO DE PROGRAMACIÓN DEL PROYECTO .....	112
ANEXO II: CATÁLOGOS DE DATOS .....	117

## Índice de Figuras

<i>Figura 1. Estimación global de vehículos autónomos.</i> .....	16
<i>Figura 2. Explicación gráfica de la tecnología necesaria</i> .....	28
<i>Figura 3. Etapas de un proceso de visión por computador.</i> .....	31
<i>Figura 4. Placa Arduino.</i> .....	33
<i>Figura 5. Beaglebone Black.</i> .....	34
<i>Figura 6. Jetson TX2 de Nvidia.</i> .....	35
<i>Figura 7. Vehículo Waymo.</i> .....	37
<i>Figura 8. Olli, desarrollado por IBM.</i> .....	38
<i>Figura 9. EZ10. Primer autobus autónomo.</i> .....	39
<i>Figura 10. Proyecto de GATEway.</i> .....	40
<i>Figura 11. Dispositivos del sistema.</i> .....	41
<i>Figura 12. Visión desde el interior y cámaras en funcionamiento.</i> .....	42
<i>Figura 13. Robot sigue línea infrarrojos.</i> .....	45
<i>Figura 14. Detectando la línea con OpenCV.</i> .....	46
<i>Figura 15. Coche autónomo RC.</i> .....	46
<i>Figura 16. Aviso en el salpicadero de las señales próximas.</i> .....	47
<i>Figura 17. Funcionamiento LDW.</i> .....	48
<i>Figura 18. Funcionamiento LKS.</i> .....	48
<i>Figura 19. Detección de un objetivo con un dron.</i> .....	49
<i>Figura 20. Esquema del sistema propuesto.</i> .....	50
<i>Figura 21. Diagrama del robot: (A) Ruedas; (B) RaspiCam;</i> .....	51
<i>Figura 22. Vista superior y lateral del robot.</i> .....	51
<i>Figura 23. Conexionado eléctrico.</i> .....	52
<i>Figura 24. Conexión de RaspiCam en Raspberry Pi.</i> .....	53
<i>Figura 25. RaspiCam.</i> .....	53
<i>Figura 26. Raspberry Pi 3 Model B+.</i> .....	54
<i>Figura 27. Esquema Raspberry Pi.</i> .....	55
<i>Figura 28. Módulo L298N.</i> .....	56
<i>Figura 29. Esquema simple del conexionado de un motor.</i> .....	57
<i>Figura 30. Primera elección de chasis.</i> .....	58
<i>Figura 31. Sistema de locomoción diferencial.</i> .....	58
<i>Figura 32. Opción elegida de chasis.</i> .....	58
<i>Figura 33. Batería de litio recargable.</i> .....	59
<i>Figura 34. Logo Raspbian.</i> .....	60
<i>Figura 35. Escritorio de la Raspberry Pi.</i> .....	60
<i>Figura 36. Logo de Anaconda.</i> .....	61
<i>Figura 37. Logo de Python.</i> .....	61
<i>Figura 38. Logo de OpenCV.</i> .....	61
<i>Figura 39. Funcionamiento lógico de la aplicación.</i> .....	63
<i>Figura 40. Carril bici.</i> .....	64
<i>Figura 41. Estructura código.</i> .....	64
<i>Figura 42. Representación cilíndrica HSV frente representación cúbica BGR.</i> .....	65
<i>Figura 43. Imagen del carril bici en el espacio de color HSV.</i> .....	66
<i>Figura 44. Arriba región de interés; abajo imagen sin region de interés.</i> .....	66
<i>Figura 45. Imagen anterior y posterior al filtrado.</i> .....	67

<i>Figura 46. Campana de Gauss.</i> .....	68
<i>Figura 47. Máscara o Kernel.</i> .....	69
<i>Figura 48. Imagen después y antes del proceso.</i> .....	69
<i>Figura 49. Imágenes con filtro Canny.</i> .....	70
<i>Figura 50. Representación transformada de Hough para rectas.</i> .....	71
<i>Figura 51. Ejemplos tras aplicar la transformada de Hough.</i> .....	72
<i>Figura 52. Distintos casos de detección.</i> .....	76
<i>Figura 53. Visión del robot del giro hacia la izquierda.</i> .....	77
<i>Figura 54. Filtro 3x3 (izq.) y 5x5 (dcha.).</i> .....	78
<i>Figura 55. Filtro 3x3 (izq.) y 7x7 (dcha.).</i> .....	79
<i>Figura 56. Diferencias entre GaussianBlur (izq.) y MedianBlur (dcha.).</i> .....	79
<i>Figura 57. Programa filtro Canny. Al inicio (izq.) y modificando la foto (dcha.).</i> .....	80
<i>Figura 58. Comparacion de la configuración elegida frente a las no elegidas.</i> .....	81
<i>Figura 59. Resultados primer software.</i> .....	82
<i>Figura 60. Diferencias de resultados entre HoughLinesP (arriba) y HoughLines (abajo).</i> .....	84
<i>Figura 61. Código de comprobación de los motores.</i> .....	85
<i>Figura 62. Robot y ordenador en las pruebas.</i> .....	86
<i>Figura 63. Robot en funcionamiento.</i> .....	86
<i>Figura 64. Visión de la recta desde la cámara del robot.</i> .....	87
<i>Figura 65. Visión de la recta tras el filtro Canny.</i> .....	87
<i>Figura 66. Visión de la recta desde la cámara del robot.</i> .....	88
<i>Figura 67. Distintos puntos de vista del robot.</i> .....	88
<i>Figura 68. Utilización de Filtro (arriba); falsos positivos (abajo).</i> .....	89
<i>Figura 69. Punto de vista del robot curva derecha.</i> .....	89
<i>Figura 70. Falso positivo debido a la sombra de la valla.</i> .....	91
<i>Figura 71. Valores de tiempo (segundos) entre procesos.</i> .....	92
<i>Figura 72. Valores de tiempo (segundos) entre procesado de imágenes.</i> .....	93
<i>Figura 73. Diagrama de Gantt.</i> .....	96

## Índice de Tablas

<i>Tabla 1. Visualización de niveles de automatización.</i> .....	26
<i>Tabla 2. Comparativa de microcontroladores estudiados.</i> .....	35
<i>Tabla 3. Resumen vehículos de media distancia.</i> .....	40
<i>Tabla 4. Resumen de los diseños de vehículos autónomos.</i> .....	43
<i>Tabla 5. Conexiones Raspberry Pi y Módulo L298N.</i> .....	57
<i>Tabla 6. Valores basados en la reolución.</i> .....	74
<i>Tabla 7. Rangos del filtro de color blanco basado en la luminosidad.</i> .....	75
<i>Tabla 8. Valores de tiempo (segundos) entre procesos.</i> .....	92
<i>Tabla 9. Valores de tiempo (segundos) entre procesado de imágenes.</i> .....	93
<i>Tabla 10. Tareas y fechas de la planificación.</i> .....	95
<i>Tabla 11. Coste de personal.</i> .....	99
<i>Tabla 12. Coste de hardware y software.</i> .....	100
<i>Tabla 13. Costes indirectos.</i> .....	101
<i>Tabla 14. Coste total.</i> .....	101
<i>Tabla 15. Costes y beneficios de la empresa.</i> .....	103

# CAPÍTULO 1. INTRODUCCIÓN.

La conducción autónoma está llamada a ser una de las tecnologías más importantes en el futuro, aunque en el presente ya se puede disfrutar de algunas de sus ventajas. Sin embargo, se espera que el impacto en la industria automovilística, hasta el año 2030, sea más o menos progresivo. Sobre ese mismo año se espera que la tecnología mueva en torno a 50.000 millones de euros, siempre y cuando la industria automovilística sea capaz de dominar nuevos campos y modelos de ventas, y así aprovechar esta ventaja.

Además, se estima que a partir del año 2030 la amplia adopción de la conducción autónoma completa va a crear un cambio revolucionario en la compra, propiedad, uso y diseño de los vehículos redefiniendo toda la industria.



Figura 1. Estimación global de vehículos autónomos.

Como se puede observar en el gráfico anterior, la estimación de coches, de cualquier nivel de autonomía, circulando por carretera, aumentará aproximadamente 134% cada año. Esto significa que en 2020 se espera que haya 10 millones de coches circulando y se espera que esta cifra aumente progresivamente.

El aumento de vehículos autónomos traerá consigo un gran número de ventajas a la sociedad, como dijo el CEO de NVIDIA, Jensen Huang, “*The contribution of self-driving cars to society is, arguably, incredible (La contribución de la conducción autónoma a la sociedad es, indiscutiblemente, increíble)*”. La primera y principal ventaja será la seguridad. Por citar un ejemplo, según un informe de la consultora McKinsey & Company, sólo con que existieran un

10-20% de coches con algún nivel de autonomía en Estados Unidos, se podrían evitar aproximadamente 140.000 muertes en un año por accidentes ahora provocados por errores humanos.

Los errores humanos provocan más del 94% de los accidentes de automóvil, además la Organización Mundial de la Salud estima que más de 1,2 millones de personas mueren en el mundo anualmente en accidentes de automóvil. Un número muy alto, “*un número mayor que las personas muertas en guerra*”, como dijo Gill Pratt, CEO del Instituto de Búsqueda de Toyota en 2016, en la Conferencia de Tecnología GPU. A este beneficio hay que sumarle el ahorro de millones de euros debido al descenso de accidentes sin mortalidad. [1]

Otra de las ventajas son las bajas emisiones de dióxido de carbono, ya que reducen el consumo de combustible en un 20%. Además se prevé una mejoría de los atascos en un futuro con vehículos autónomos, que unido a la conducción autónoma proporcionará más tiempo libre a los conductores.

Por último, no debe olvidarse que estos vehículos permitirán el transporte a personas que actualmente no pueden conducir. Este es uno de los puntos donde este proyecto desea explotar todas sus capacidades, ya que es una tecnología de un bajo coste y por lo tanto muy asequible, a diferencia de los futuros vehículos autónomos, que estarán valorados alrededor de 200.000 euros y el coste de reparaciones y mantenimiento será muy alto.

Este proyecto no pretende competir con los vehículos autónomos porque es limitado tanto en sensores, sólo dispone de una cámara, como en el procesador. A esto hay que añadirle que no se puede contar con las tecnologías de inteligencia artificial, las horas de aprendizaje de las mismas y la mayor inversión. Debido a estas limitaciones, este proyecto busca instalarse en vehículos con una movilidad en entornos reducidos, cerrados o en plataformas no tan desarrolladas como los automóviles, es decir, este proyecto busca instalarse en carritos de empresa, de aeropuertos, de campus, sillas de ruedas y sitios donde no es necesario llegar a una tecnología tan desarrollada, ya que son entornos mucho más simples para la navegación.

El proyecto es de código abierto, “*open source*”, lo que significa que cualquier persona puede crear mejoras y así ayudar a mejorar el proyecto. Con esto se pretende poder crecer y desarrollarse, es una opción utilizada en la actualidad por empresas dedicadas a las tecnologías de conducción autónoma, como Tesla con sus patentes.

## 1.1. MOTIVACIÓN.

La conducción autónoma de los vehículos es una de las grandes tecnologías emergentes y en desarrollo. La nueva búsqueda por desarrollar un sistema de control aplicable a situaciones cotidianas y con un mayor rendimiento al menor precio es uno de los campos con mayor desarrollo.

En la actualidad, los vehículos autónomos están lejos de introducirse en la vida cotidiana debido a la dificultad de controlar todo lo que les rodea, además, si esta tecnología se pudiese implantar, tendría un alto precio. Por ello, el uso de este sistema de conducción autónoma más simple permite, y en mayor medida permitirá disponer de vehículos autónomos a cualquier persona en un ámbito más simple. Esto supondrá una gran mejora en confort, calidad de vida y seguridad.

Por otro lado, diseñar un robot desde cero es un reto, al diseñar el sistema teniendo como placa principal la Raspberry Pi ha resultado muy interesante debido al amplio ámbito que tiene, no sólo de robótica y automática. Además, el manejo de la Raspberry y sus controladores supone una forma de analizar y entender mejor cómo funcionan los dispositivos, lo que en un futuro puede capacitar para desarrollar nuevos dispositivos o mejorar los ya existentes.

La suma de estas motivaciones hace el trabajo interesante para la formación de un ingeniero de electrónica industrial y automática, ya que la robótica combinada con la automatización de procesos es una de las opciones con mayor futuro de este tipo de grado y en el desarrollo profesional. Además, el diseño y construcción del robot permite aplicar y materializar parte de lo aprendido durante estos años.

Una motivación extra es que la realización de este proyecto implica el aprendizaje y dominio de un nuevo lenguaje de programación de software, *Python*, con una librería muy utilizada: *OpenCV*, utiliza en la actualidad en muchos de los diferentes sistemas de visión artificial, no solo de conducción autónoma.

Estas razones expuestas justifican la realización de este Trabajo de Fin de Grado, con el fin de promover e impulsar distintas formas de conducción autónoma de menor coste en ámbitos con poco tránsito, comenzando por un carril bici.

## 1.2. OBJETIVO.

El presente proyecto consiste en el desarrollo y creación de un robot capaz de guiarse, en un entorno limitado, por un sistema de visión artificial basado en un dispositivo *Raspberry Pi 3 Model B*, una cámara y los actuadores conectados a la *Raspberry Pi*.

El robot deberá tener un coste asequible y un diseño fácil de integrar en una plataforma no autónoma para dotarla de esa capacidad. Para ello es necesario un buen estudio de todos los componentes que necesitamos y una buena elección de estos mismos.

Se pretende realizar el diseño y creación de un robot móvil autónomo simulando un vehículo capaz de circular en una vía y detectando las líneas que la delimitan, y para ello se eligió como vía de circulación un carril bici.

El software estará desarrollado en *Python* y deberá transformar las imágenes que obtiene la cámara en órdenes claras para guiar al robot. Deberá ser un software rápido, dentro de las posibilidades que nos ofrece la *Raspberry Pi*.

## 1.3. ESTRUCTURA DEL TRABAJO.

El primer capítulo se centra en exponer la motivación que ha provocado la realización del presente Trabajo de Fin de Grado. Además, se expone de forma breve lo que se intenta conseguir, y cómo y con qué elementos se ha trabajado para intentar llevarlo a cabo.

Para una mayor comprensión, en el capítulo dos se ha realizado el estado del arte, donde se enumeran y explican algunos conceptos de las tecnologías utilizadas en este proyecto. Se han detallado las tecnologías y elementos que componen la conducción autónoma terrestre, como inteligencia artificial, visión artificial y plataformas embebidas. Por último, se concluye el capítulo analizando distintas aplicaciones ligadas a las tecnologías anteriormente expuestas.

El caso de estudio es planteado en el capítulo tres. Comienza explicando el sistema propuesto, realizando una introducción al sistema y detallando aspectos generales de funcionamiento. Posteriormente se realiza un análisis de los diferentes elementos empleados en el sistema de hardware y se realiza una explicación teórica, acompañada de ejemplos prácticos del proyecto, de cada función utilizada en el software.

En el capítulo cuatro se plantea la evaluación, pruebas y limitaciones del proyecto. Se comienza realizando una explicación de los procedimientos de elección de los parámetros de las funciones del sistema. Una vez terminada esta primera explicación, se profundiza en el funcionamiento del sistema y por último, se explican y justifican las distintas limitaciones del sistema. Todas las explicaciones se acompañan de ejemplos prácticos basados en las pruebas realizadas.

En el capítulo cinco muestra la planificación seguida para realizar el proyecto.

El marco regulador para este proyecto se recoge en el capítulo seis, analizando la legislación aplicable al proyecto y normativa de estandarización aplicable en una futura comercialización.

El capítulo siete se basa en el entorno socio-económico e incluye el presupuesto de realización del presente Trabajo de Fin de Grado y el análisis de una posible comercialización.

En el capítulo ocho se detallan futuras mejoras y líneas a seguir para mejorar el proyecto y en el capítulo nueve se exponen las conclusiones personales del autor acerca del proyecto y del sistema propuesto.

Por último, en el capítulo diez se han incluido una serie de referencias que han servido de apoyo para la realización del presente documento. Además, a continuación se ha añadido una relación de anexos donde se encuentra el código de programación del sistema propuesto.

## CAPÍTULO 2. ESTADO DE LA TÉCNICA.

Para la realización de un proyecto conviene conocer todo el ámbito relacionado con éste, por eso se procederá a explicar los puntos importantes relacionados con la navegación autónoma y la tecnología de conducción autónoma terrestre.

### 2.1. NAVEGACIÓN AUTÓNOMA.

La movilidad autónoma en los robots requiere la capacidad de navegar. Las tecnologías de navegación son necesarias para moverse con éxito a través de los diferentes entornos. En aplicaciones avanzadas, los robots necesitan actuar en entornos no estructurados y dinámicos sin una guía humana continua. En el futuro, las aplicaciones pueden requerir períodos prolongados de funcionamiento continuo, en cuyo caso la provisión de niveles apropiados de rendimiento de navegación autónoma puede convertirse en una barrera crítica para la absorción del mercado.

La navegación puede definirse como la combinación de tres tecnologías fundamentales:

- Autolocalización.
- Mapeo e interpretación del mapa.
- Planificación y ejecución de trayectoria y movimiento.

Por lo general, las dos primeras competencias son facilitadoras de la tercera. No ser capaz de localizar en un entorno conocido o desconocido de una manera fiable inhibe la generación de planes de trayectoria y movimientos exitosos. La autonomía de un robot depende de su capacidad de navegar de forma independiente en el entorno y en cooperación con otros elementos implicados en su función como humanos u otros robots. [2]

#### 2.1.1. TÉCNICAS Y MÉTODOS CLAVES.

Los robots se basan principalmente en sensores de a bordo para localizar y crear mapas fiables del medio ambiente para establecer dónde desplazarse de forma autónoma. Los robots también pueden hacer uso de información ambiental intrínseca para establecer la posición, o confiar en sistemas basados en balizas con el fin de aumentar la robustez. Existen diferentes mecanismos de autolocalización que pueden utilizarse según la escala y la naturaleza del entorno operativo. Estos pueden clasificarse como:

- Satélite y basado en red.
- Basado en sensores (incluyendo basado en la visión).
- Basado en balizas.

Las tecnologías basadas en satélites y en red se utilizan principalmente para la navegación global y el posicionamiento absoluto al aire libre, mientras que las tecnologías basadas en los sensores (sistemas iniciales, odometría, sistemas basados en la distancia, sensores visuales y sensores

de profundidad 3D, WiFi, RFID, NFC, etc) se utilizan principalmente para una navegación en interiores y posicionamiento relativo. En casos prácticos, en escenarios al aire libre, se utiliza un enfoque mixto para aprovechar plenamente la información global y local procedente de diferentes fuentes. Por ejemplo, los vehículos aéreos ligeros realizan autolocalización mediante la fusión de datos de posicionamiento global, sensor de inercia y datos de cámaras a bordo. El mapeo genera un marco de referencia absoluto o relativo para la navegación. La técnica más común consiste en localización y asignación simultánea (*SLAM*) donde el mapeo y localización ocurren momento a momento mientras un robot explora su entorno.

En este Trabajo de Fin de Grado se utiliza una autolocalización basada en sensores, concretamente se basa en la visión artificial.

## 2.1.2. APPLICACIONES ACTUALES.

Gracias a la navegación autónoma los robots móviles pueden desplazarse de un lugar a otro sin necesidad de un control humano. Dependiendo de su entorno de trabajo podemos distinguir varios tipos de robot móviles autónomos por lo que cada tipo tendrá una serie de aplicaciones.

### 2.1.2.1. AÉREAS.

Estas aplicaciones se realizan gracias a los UAV (*Unmanned Aerial Vehicle*), en castellano VANT (Vehículo Aéreo No Tripulado), también es muy utilizada la denominación UAS (*Unmanned Aerial System*). Son vehículos sin tripulación, capaces de mantener de manera autónoma un nivel de vuelo controlado y sostenido. Dependiendo del tipo de sistema pueden alcanzar desde los 2.000 pies (610 metros) de altura hasta los 50.000 pies (15.240 metros). Gracias a la ventaja de que su duración máxima de vuelo sólo está limitada por su combustible y por su sistema de vuelo, tienen multitud de aplicaciones y posibilidades en el mercado civil y profesional. [3]

La aplicación más conocida y motivo de creación de este tipo de robot fue la defensa y seguridad (con fines militares). En cambio, una aplicación desconocida son las comunicaciones, pudiendo funcionar como plataforma de enlace de comunicaciones segura y fiable, pues además son fácilmente reconfigurables para adaptarse a los diferentes escenarios y plataformas en los que tenga que operar. Así mismo, son ideales para realizar trabajos de actualización frecuente de bases de datos cartográficas y catastrales, y están capacitados para actuar como soporte a excavaciones y prospecciones arqueológicas.

Este tipo de robot móvil es una herramienta ideal para realizar tareas de vigilancia e inspección de infraestructuras, tales como líneas de alta y media tensión, oleoductos, gaseoductos, cierres, ferrocarriles, y obra civil. Gracias a los avances se ha abierto otro campo donde hay multitud de aplicaciones, la agricultura (vigilancia de cultivos, frecuencia de riego, detección temprana de enfermedades y plagas, etc.). Para finalizar, resaltar que pueden ser muy útiles para el medio ambiente, ya que pueden detectar incendios y en caso de incendio poder realizar una búsqueda y rescate, realizar un seguimiento de áreas boscosas, comprobar el estado de la atmósfera y controlar el estado del agua y del terreno en diferentes zonas. [4]

### 2.1.2.2. ACUÁTICOS.

En cuanto a las aplicaciones acuáticas se debe explicar que existe menor variedad que las aéreas, ya que llevan menor tiempo en desarrollo. Existen dos tipos de robot móvil acuático dependiendo de si realizan un trabajo submarino, UUV (*Unmanned Underwater Vehicles*) o sobre la superficie, USV (*Unmanned Surface Vehicle*). Por tanto, se tendrán dos grupos de aplicaciones.

Los UUV tienen aplicaciones de tipo comercial industrial, donde las empresas de petróleo y gas usan estos vehículos para tomar mapas detallados del fondo del mar antes de comenzar a crear sus plantas petrolíferas. De esta manera saben donde deben colocar sus plantas para sacar el mayor provecho y ganancias al fondo marino.

Otra aplicación centrada en el fondo marino es la científica, que analiza la absorción de luz, qué elementos y componentes se encuentran en el agua o la existencia de microorganismos. Además de las aplicaciones mencionadas, los UUV pueden ser utilizados para realizar misiones de rescate por accidentes aéreos o navíos, en trabajos de catástrofes medioambientales y misiones militares. [5]

Por último, las aplicaciones de los vehículos autónomos de superficie son mucho más limitadas y en la actualidad los USV sólo son destinados al ocio personal y a la oceanografía.

### 2.1.2.3. TERRESTRES.

Los avances tecnológicos están aumentando el desarrollo del sector de los vehículos no tripulados terrestres o UGV (*Unmanned Ground Vehicle*), observándose que, tecnológicamente, este tipo de sistemas autónomos, y en consecuencia sus aplicaciones, avanza más rápido de lo que se esperaba hace unos años. [6]

Hoy en día existe una gran variedad de usos de los UGV. Principalmente, estos vehículos se utilizan para remplazar a las personas en situaciones muy duras y peligrosas, como pueden ser manejo de explosivos y desactivación de bombas; en situaciones donde se necesita mucha fuerza, se ha de operar en espacios reducidos o donde una persona no podría escapar fácilmente. Estas tecnologías inicialmente fueron creadas para uso militar y espacial, por lo que estos robots son muy utilizados en misiones de supervivencia, reconocimiento, adquisición de información y situación del objetivo.

En cuanto a las aplicaciones civiles, son utilizados en la agricultura, tractores autónomos son capaces de sembrar y recoger la cosecha. Con aplicación en la minería, se han desarrollado camiones capaces de trabajar y desplazarse bajo tierra de forma autónoma. Además, en las cadenas de suministros, pueden ser utilizados en el sistema de gestión de almacenes, desde el transporte de mercancías hasta el escaneo de existencias y la toma de inventario. [7]

Este proyecto se engloba dentro de las aplicaciones terrestres, ya que su funcionamiento se basa en la detección de las líneas que delimitan determinadas vías terrestres, similar al de los vehículos autónomos terrestres. Sus posibles aplicaciones podrían estar relacionadas con el transporte de personas o mercancías dentro de entornos con condiciones simples de tráfico.

## 2.2. VEHÍCULOS DE CONDUCCIÓN AUTÓNOMA TERRESTRE.

### 2.2.1. NIVELES Y EVOLUCIÓN.

Recientemente, grandes empresas han apostado por la conducción autónoma, comenzando planes para futura venta de vehículos autónomos. Por lo tanto, la conducción autónoma es una realidad y, aunque aún no ha llegado a su plenitud máxima y es minoritaria entre la población, se espera una introducción gradual de su funcionalidad en los próximos 15 años. [8]

Se pueden distinguir varios niveles para llegar al objetivo final de la completa conducción autónoma, exactamente seis.

- **Nivel 0 - No automatización:** En el primer nivel no existe automatización, los conductores tienen completo control sobre todos los movimientos, tanto longitudinales como laterales. En esta etapa, los conductores tienen la responsabilidad de realizar maniobras seguras tanto para su propia integridad como para los conductores que le rodean, así como estar al tanto de los posibles cambios del tráfico, del medioambiente y en consecuencia de las posibles amenazas que estos cambios generen. Lógicamente, esta etapa ya existe.
- **Nivel 1 - Automatización de funciones específicas:** En esta segunda etapa, hay un gran número de funciones específicas que están automatizadas pero tienen un control individual, es decir, no están relacionadas entre si y no comparten información entre ellas, como por ejemplo el control de crucero (mantendrá una velocidad fija), el aviso de salida de carril o ayuda en el aparcamiento. El conductor sigue siendo responsable de realizar una conducción segura y sigue teniendo el control de todos los movimientos, debe tener el control del volante y de los pedales en todo momento, pero puede delegar ciertas acciones del control principal al sistema de “autopiloto” como la velocidad de crucero. Como el nivel anterior, este nivel ya está en funcionamiento.
- **Nivel 2 - Automatización de funciones combinadas:** Etapa en la que un mínimo de dos sistemas principales de conducción están automatizados y operan simultáneamente, como por ejemplo control de crucero adaptativo con asistente de mantenimiento de carril. En esta ocasión, los conductores podrán delegar el control del coche en estas funciones bajo unas circunstancias específicas, supervisando en todo momento las acciones y estando preparado para poder tomar el control. De todos modos, el conductor sigue siendo responsable de realizar una conducción segura fuera de estas circunstancias específicas y de ir controlando todo lo que sucede a su alrededor. Como ocurre con los dos niveles anteriores, también está ya en funcionamiento.
- **Nivel 3 - Conducción autónoma limitada:** En la cuarta etapa, se diseñará un sistema en el que el coche sea capaz de operar de forma segura mientras se encuentra en modo de conducción autónoma. Los conductores pueden ceder todas las funciones críticas de seguridad bajo algunas condiciones y no necesitan estar

permanentemente controlando el sistema y el tráfico, mientras el sistema está activo. En el momento en que el sistema no fuese capaz de mantener la conducción autónoma, el conductor podría retomar el control del vehículo de forma segura. Este sistema sería utilizado exclusivamente en autopistas. Según los avances realizados esta etapa debería implementarse entre los próximos años (2018-2020).

- **Nivel 4 - Conducción autónoma bajo condiciones específicas:** Se avanza hacia una conducción autónoma semicompleta, ya que es completo pero sólo bajo algunas condiciones. El vehículo puede realizar todas las funciones de conducción de forma autónoma pero bajo unas buenas condiciones de tráfico, climatología, bajas velocidades y en un entorno cerrado, como podrían ser aparcamientos y zonas urbanas de baja velocidad. En esta etapa, el conductor solo tiene que ser un pasajero más mientras el sistema esté activo, teniendo como único objetivo introducir el destino a donde desea llegar. Se estima que esta etapa se pueda llegar a conseguir entre los años 2020 y 2025.
- **Nivel 5 – Conducción autónoma completa:** Una vez que se hayan realizado con éxito todas las etapas, se conseguiría tener un sistema donde todas las funciones de conducción están automatizadas y se realizan de forma segura, sin importar las condiciones en las que se encuentre y detectando todos los posibles riesgos. Además, este sistema será capaz de crear varios escenarios y escoger aquél con menos riesgo para evitar cualquier peligro. El conductor será un mero pasajero en todo momento, teniendo que realizar al inicio del viaje una única función: elegir destino. Se piensa que se podrá realizar una conducción autónoma completa en ciudades entre los años 2025 y 2030, y para conseguir realizar una conducción autónoma de “puerta a puerta” se debería esperar hasta el año 2040.

Una vez explicadas las distintas etapas o niveles se estima que este proyecto se encontraría en la etapa 3, donde se encuentran vehículos con conducción autónoma limitada, ya que es un prototipo muy sencillo que funciona en un entorno limitado. Para poder optar a los siguientes niveles debería aumentar sus funciones, para poder circular por cualquier entorno. Además, se deberían añadir elementos tanto de hardware como de software. [9]

	Nivel 0	Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5
<b>Grado de automatización</b>	No automatización	Automatización de funciones específicas	Automatización de funciones combinadas	Conducción autónoma limitada	Conducción autónoma bajo condiciones específicas	Conducción autónoma completa
<b>Características</b>	El conductor tiene control total	Existen funciones automatizadas controladas individualmente	Existen varias funciones automatizadas que funcionan simultáneamente	El coche opera de forma segura en autopistas	El coche opera de forma autónoma bajo condiciones favorables y en entornos cerrados	El coche opera de forma autónoma en cualquier ocasión
<b>Responsabilidad del conductor</b>	El conductor es responsable de una conducción segura	El conductor es responsable de una conducción segura.	El conductor puede delegar el control del coche bajo ciertas circunstancias, siempre vigilándolo	El conductor puede delegar el control del coche sin tener que estar vigilando sus acciones.	El conductor puede delegar el control del coche sin tener que estar vigilando sus acciones.	El conductor puede delegar el control del coche y relajarse.
<b>Timeline</b>	Existente	Existente	Existente	2018-2020	2020-2025	2030-2040

Tabla 1. Visualización de niveles de automatización.

## 2.2.2. TECNOLOGÍA NECESARIA.

Para sustituir el elemento humano en la conducción se deben utilizar diferentes tecnologías desarrolladas de forma simultánea, para poder sustituir funciones humanas sensoriales realmente importantes.

Una de las principales funciones es la capacidad para tomar decisiones mientras se conduce, que tienen los seres humanos, la cual se debería sustituir por una máquina capaz de aprender multitud de distintos algoritmos. Este ordenador central será el cerebro del coche autónomo, actuará en tiempo real gracias a un software que procesará la información y tomará las decisiones oportunas. En el proyecto, este ordenador central será el microcontrolador elegido: la *Raspberry Pi 3 Model B*. Atendiendo a las capacidades cerebrales de los humanos, se debe sustituir la memoria por una serie de mapas (sistema GPS/IMU y mapas especiales) y modelos y señales del entorno. Debido a las limitaciones de velocidad de procesamiento de la *Raspberry Pi* (expuestas en el apartado 4.4. *Limitaciones y Errores*), será imposible desarrollarlo.

En cuanto a los sentidos de la vista y el oido, serán sustituidos por todo tipo de sensores (cámaras, sensores infrarrojos, sensores de proximidad, sensores térmicos, radar, sensores ultrasonido, receptores de sonido y láser) capaces de operar en variedad de condiciones (lluvia, nieve, carreteras no pavimentadas, túneles, etc.) y que darán grandes cantidades de datos e información al sistema central para que controle el vehículo. El robot diseño para este proyecto cuenta con una cámara, los demás sensores han sido eliminados debido al bajo presupuesto.

Las cámaras sustituyen al sentido de la vista y reconocerán a otros vehículos, peatones, colores, luces, señales de tráfico, etc. Para ello, se necesitará colocar un gran número de cámaras a lo largo del perímetro del vehículo. Por si solas, no ofrecen grandes funcionalidades pero combinadas con algoritmos y técnicas de visión artificial, se dota al vehículo del sentido de la visión. Por otro lado, para sustituir el sentido auditivo los sensores más importantes en esta función serán los micrófonos y sensores de sonido, que recogerán todo lo que suceda a su alrededor a nivel acústico. [10]

Para suplir la capacidad de comunicación, utilizada para intercambiar distinta información sobre carreteras, atascos, etc, el vehículo autónomo necesitará tener un sistema de comunicación con otros vehículos, para estar informado en todo momento de lo que ocurre a su alrededor, por lo que se necesitará una conexión inalámbrica de corto alcance para la comunicación vehículo-vehículo y otra conexión de largo alcance para acceder a los mapas, mejoras de software del sistema, para conocer el estado de las carreteras y para mensajes de emergencia.

La conducción, que requiere de las personas realizar una serie de acciones y movimientos coordinados con su cuerpo, se transformará en la perfecta coordinación de una serie de controladores y actuadores. Para realizar las distintas acciones se necesitará tener una transmisión automática y controles automatizados (frenar, giros, señales, etc.). Aunque sea obvio, no se debe olvidar que para el correcto funcionamiento de este sistema es necesario tener servidores, software y suministros de potencia con una gran fiabilidad y rendimiento.

Por último, se debe dar tanta o mayor importancia a las pruebas, mantenimiento y reparación de los distintos componentes y sensores que se encuentran en el vehículo, ya que un fallo en uno de ellos puede llevar a graves consecuencias.

Para visualizar lo anteriormente expuesto se presenta la siguiente imagen:

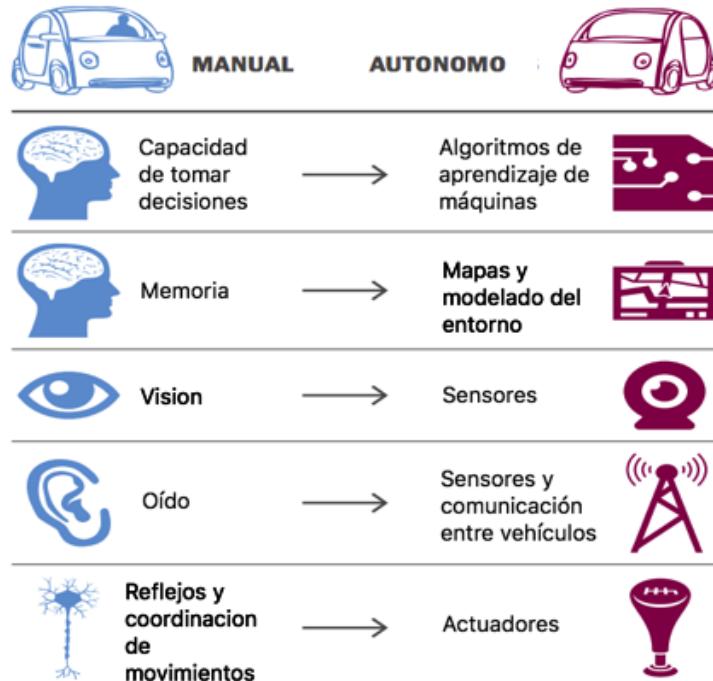


Figura 2. Explicación gráfica de la tecnología necesaria

Como se puede apreciar, muchas de las tecnologías mencionadas anteriormente ya existen a día de hoy. Sin embargo, para facilitar y mejorar la conducción en condiciones adversas y de mucho tráfico, se requiere llegar a dominar varias áreas. En este proyecto no serán necesarias debido a la simplicidad del mismo, comparado con un vehículo autónomo. En este proyecto se usará la inteligencia artificial, la visión por computador y un microcontrolador central que será el encargado de tomar decisiones.

### 2.2.2.1. INTELIGENCIA ARTIFICIAL.

En el ámbito de las ciencias de la computación se denomina como inteligencia artificial (IA) a la facultad de razonamiento que ostenta un agente que no está vivo, es una de las ramas de la Informática, con fuertes raíces en distintas áreas como las ciencias cognitivas, la filosofía o la lógica.

Inicialmente nació como un estudio filosófico que buscaba imitar la inteligencia humana. El término de inteligencia artificial se le debe a John McCarthy, en una reunión celebrada en 1956 en Dartmouth (Estados Unidos), en la que participaban los principales investigadores del área, y donde se hicieron previsiones triunfalistas a diez años que jamás se cumplieron, lo que provocaría el abandono casi total de las investigaciones durante quince años. A mediados de los años 60, aparecen los sistemas expertos, los cuales predicen la probabilidad de una solución bajo unas condiciones preestablecidas. Sin embargo, se tendría que llegar a las décadas de 1970 y 1980 para que creciese el uso de los sistemas expertos. A partir de 1990 se conoce como la era moderna, consiguiendo que la computadora autónoma Deep Blue ganase una partida de ajedrez a Gari Kasparov. En la actualidad, la inteligencia artificial está en una etapa de redefinición dando prioridad a la comprensión de la inteligencia humana. [11]

Se puede pensar en la inteligencia artificial como en aquella ciencia que incorpora conocimientos a los procesos o actividades para que éstos tengan éxito, desarrollando software informático capaz de ejecutar trabajos de forma inteligente. Tiene como principales objetivos, estudiar el comportamiento inteligente de las personas y crear programas computacionales capaces de imitar el comportamiento humano. [12]

Desde su creación son muchas las ramas que surgen del tronco común de la inteligencia artificial. Las Ciencias de la Computación han asistido continuamente al nacimiento de nuevas ramas y se habla de sistemas expertos, vida artificial, algoritmos genéticos, computación molecular o redes neuronales. En algunas de estas ramas los resultados teóricos van muy por encima de las realizaciones prácticas. [13]

Dentro de las ramas se pueden distinguir dos tipos, el primer tipo son las áreas clásicas de la inteligencia artificial donde se encuentran: [14]

- **Sistemas Expertos (Sistemas basados en conocimiento):** Programas que resuelven problemas que normalmente requieren del conocimiento de un especialista o experto humano. Es un sistema capaz de tomar decisiones inteligentes interpretando grandes cantidades de datos sobre un dominio específico de problemas.
- **Aprendizaje y razonamiento automático:** Máquinas capaces de planificar, tomar decisiones, plantear y evaluar estrategias, aprender a partir de la experiencia, autoreprogramables, etc
- **Robótica:** Artefactos autónomos capaces de llevar a cabo diversas tareas mecánicas de manera flexible e inteligente, cumpliendo con un objetivo y ajustándose al entorno cambiante.
- **Procesamiento de lenguaje natural:** Sistemas capaces de reconocer, procesar y emular el lenguaje humano.
- **Visión artificial (Reconocimiento de patrones):** Reconoce y procesa señales, caracteres, patrones, objetos y escenas. Además toman decisiones en base a lo que ve.

El otro tipo son ramas más recientes, llamadas áreas de vanguardia, y están basadas en la inteligencia natural. Estas ramas son:

- **Redes neuronales:** Crear elementos de procesamiento y organizarlos de acuerdo a un modelo basado en las células del cerebro humano (neuronas). Estos sistemas no se programan, se entrenan, ya que tienen la capacidad de aprendizaje, el cual irá aumentando cuanto mayor sea el número de muestras.
- **Lógica difusa:** Basado en los principios del razonamiento aproximado y el "cálculo con palabras", éstos sistemas logran simplificar y aproximar la descripción del problema de una manera natural, eficiente y robusta. La lógica difusa va más allá de la lógica booleana en cuanto a que acepta valores parciales.

- **Algoritmos genéticos:** En esta área se intenta replicar comportamientos biológicos, con los avances científicos que ello implica, mediante la computación. Se trata de algoritmos de búsqueda basados en la mecánica de la selección natural y de la genética. Utilizan la información histórica para encontrar nuevos puntos de búsqueda de una solución óptima del problema planteado, con la esperanza de mejorar los resultados. Algunas de las aplicaciones son el reconocimiento de palabras habladas, conducción automática de un vehículo, aprendizaje de máquinas y en sistemas de economía. [15]
- **Agentes (wizards):** Son programas "invisibles" tipo espía, que analizan las tareas que está llevando a cabo un usuario y que, dependiendo de las preferencias, costumbres y nivel del usuario, en cuanto se detecte alguna anomalía, el agente "aparece" ante el usuario para ayudarle (dando información), sugiriendo una solución o para ejecutar un conjunto de tareas rutinarias de manera automática.

Este proyecto está desarrollado dentro de las ramas de robótica y visión por computadora, ya que el robot es capaz de tomar diferentes decisiones dependiendo de los patrones reconocidos mediante la visualización (detección de líneas) en un entorno cambiante.

La inteligencia artificial produce un gran número de ventajas comenzando en el ámbito empresarial y laboral reduciendo tiempos, costes y salarios adicionales, sin olvidar el aumento de producción, permitiendo un incremento de ingresos. Gracias a la inteligencia artificial se han conseguido desarrollar aplicaciones que realizan tareas que el hombre nunca hubiera podido llegar a realizar debido a su complejidad. Por último, pueden predecir situaciones o complicaciones a largo plazo, algo que resulta muy útil, sin olvidar que en un futuro pueden llegar a lograr grandes hallazgos y avances.

Por otra parte, como todo sistema informático, requiere de un mantenimiento y constantes actualizaciones, lo que conlleva un coste añadido al dinero y tiempo que se requiere para poder realizar estos sistemas expertos. Toda innovación tiene su riesgo, y la posible creación de máquinas autosuficientes, capaz de realizar multitud de tareas, puede crear rechazo por el posible uso irracional de esta tecnología.

### **2.2.2.2. VISIÓN POR COMPUTADOR**

Es un campo de la inteligencia artificial enfocado a que las computadoras puedan extraer información a partir de imágenes, ofreciendo soluciones a problemas del mundo real a partir de ellas. Un ejemplo que se puede ver en el día a día, es la detección de caras que realizan al hacerse una foto los Smartphones.

La visión por computador tiene como principal objetivo obtener la información explícita y el significado de la realidad de la misma forma que lo haría un ser biológico. [16]

Hay muchas tecnologías que utilizan la visión por computador, entre las cuales están: el reconocimiento y seguimiento de objetos, detección de eventos, reconstrucción de una escena (mapping), restauración de imágenes o actuaciones de control de calidad, por lo que puede aplicarse a numerosos campos como pueden ser el procesamiento de señales, reconocimiento de patrones y la robótica, entre otros.

El objetivo de la visión por computadora es, como ya se ha mencionado anteriormente, extraer información a partir de imágenes. Para poder extraer esta información se debe realizar un proceso que se puede dividir en seis áreas principales: [17]

1. Adquisición de la imagen: proceso que nos lleva a la obtención de una imagen visual.
2. Preprocesamiento: técnicas de reducción de ruido y enriquecimiento de detalles en la imagen.
3. Segmentación: proceso que divide una imagen en objetos de interés.
4. Extracción de características: se diferencian unos objetos de otros a partir de un conjunto de características útiles.
5. Reconocimiento: proceso que identifica esos objetos.
6. Interpretación: asigna un significado a un conjunto de objetos reconocidos.

La figura 3 muestra un diagrama de las etapas a considerar, de forma general, en un proceso de visión artificial:

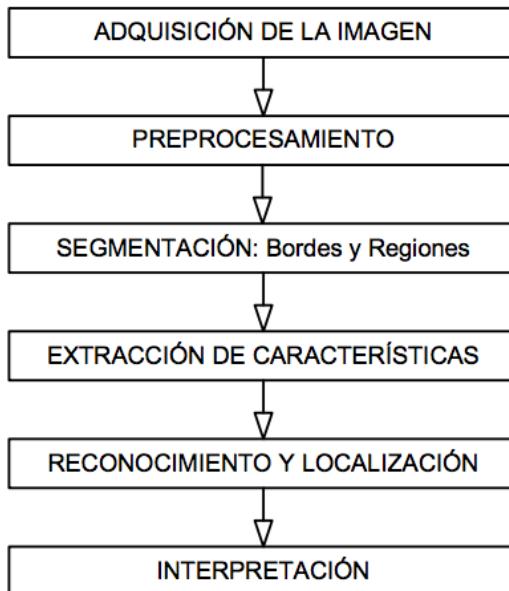


Figura 3. Etapas de un proceso de visión por computador.

Mientras que la percepción visual es algo innato y cotidiano para los seres humanos, la visión artificial es muy compleja y encuentra muchas dificultades mientras se desarrolla. Entre las principales dificultades, destacan: [18]

- **Mundo Tridimensional:** mientras que las imágenes que se obtienen con una cámara son bidimensionales, el entorno que nos rodea no lo es, por ello es necesario realizar las transformaciones necesarias para obtener los valores deseados.

- **Zona de interés:** es necesario extraer elementos sutiles en imágenes complejas, es decir, al tener mucha información es necesario reconocer formas, colores, posiciones, etc.

En este proyecto se ha realizado una selección de zona de interés, que capta únicamente las líneas a detectar, reduciendo el ruido y los falsos positivos. La citada selección se realiza en el apartado 4.1.1. *Filtrado por dimensiones de la región de interés.*

- **Carácter dinámico de las escenas:** el mundo no es una imagen, está vivo, por lo que en las imágenes que se toman muchos elementos están en movimiento. Por otro lado, también influyen otros factores como los cambios de iluminación o de contraste, que pueden cambiar por completo una imagen y marcar una gran diferencia, y por desgracia estos factores no se pueden controlar, ya que son variables.

Para este trabajo se ha transformado la imagen a otra escala de color y se ha realizado un filtro del color blanco para tratar de reducir al máximo posible los cambios de iluminación. Este proceso se explica en el apartado 4.1.2. *Filtrado por color blanco.*

- **Ambigüedad en la definición de un concepto:** las ambigüedades son muy frecuentes, y se basan en que una figura o concepto puede tener más de una posible interpretación, pueden coexistir múltiples interpretaciones o una puede dominar sobre las otras.

Para el proyecto se han reducido las ambigüedades limitando la detección de rectas por su ángulo de inclinación y distancia respecto a un punto. Esta limitación se detalla en el apartado 4.1.3. *Filtrado de las líneas.*

En el sistema de visión por computador se encuentran los siguientes componentes principales de los que dependerá en gran medida el resultado final de este proyecto: [19]

- **La iluminación** tiene una función fundamental dentro de un sistema de percepción, ya que es la característica que utilizan las técnicas de visión artificial para extraer información de los objetos de una escena. La iluminación de una escena se puede realizar con luz natural, como en este proyecto, o mediante la utilización de lámparas. No prestar suficiente atención a la iluminación puede suponer una mayor complicación en el análisis y obtención de resultados a partir de la imagen captada, provocando errores en el control de dirección del vehículo.

- **La cámara** de un sistema de visión artificial es el dispositivo encargado de recibir la luz reflejada por la escena y la utiliza para generar imágenes. El elemento más importante de la cámara es su sensor, que está formado por una capa de material fotosensible que transforma la luz incidente en señales eléctricas.

Algunas características importantes del sensor de la cámara son su sensibilidad, las dimensiones, que indican el número de píxeles con que genera la imagen (640x480 en el caso de este proyecto); la configuración de la ganancia (contraste de la imagen) y del desplazamiento (brillo); y finalmente el ruido que introduce el sensor en la imagen generada. Debido a la evolución de las cámaras y la mejora de las características

anteriormente mencionadas, las prestaciones de los sistemas de visión actuales han aumentado significativamente.

- **La óptica** es uno de los elementos más importantes de cualquier sistema de visión. Su objetivo es concentrar la luz reflejada por los objetos de la escena en el sensor de la cámara para generar la imagen. [20]

- Generalmente, en un sistema de visión artificial, el sistema encargado de realizar el **procesamiento de las imágenes** necesita un dispositivo hardware para capturar las imágenes que le envía la cámara. Además de tareas de captura, este dispositivo puede realizar algunas tareas de procesamiento de las imágenes, liberando de esta tarea al procesador principal. En este proyecto no se tendrá este tipo de sistema por lo que las tareas de procesamiento las realiza la *Raspberry Pi*.

### 2.2.2.3. MICROCONTROLADOR.

En este apartado se enumeran y explican las ventajas y desventajas de las distintas opciones que se encuentran en el mercado actualmente y cuáles han sido los motivos por los que se ha elegido la *Raspberry Pi*.

- **Arduino** es una plataforma de electrónica “*open source*”, o de código abierto, cuyos principios son poder ofrecer software y hardware fácil de entender y utilizar. Puede verse una de sus plataformas en la figura 4. El hardware consiste en una placa con un microcontrolador *Atmel AVR*, puertos digitales y analógicos de entrada y salida. Por otro lado, el software, libre, consiste en un entorno de desarrollo basado en el lenguaje de programación propio. [21]



Figura 4. Placa Arduino.

Esta plataforma tiene grandes ventajas, ya que es de pequeño tamaño y ligera, por lo que puede introducirse en multitud de estructuras, y al ser de hardware y software libre se reduce considerablemente el precio y da multitud de opciones. Además, la programación en este entorno es realmente sencilla. Se ha descartado esta opción porque el precio es ligeramente superior y su rendimiento es más bajo que el ofrecido por la *Raspberry Pi*.

- **Beaglebone Black** es un ordenador de placa reducida, en este caso, diseñado y desarrollado por Texas Instruments. Tiene características muy similares a *Arduino*, al igual que su aspecto físico, al incluir sus pines hembra en sus laterales, como se puede observar en la figura 5. Consta de una *CPU ARM Cortex-A8* de 1 GHz, 512 GB de RAM, acelerador gráfico 3D y 2 GB de almacenamiento interno, además de una ranura MicroSD. Así mismo, incluye una conexión USB, Ethernet, salida micro-HDMI y dos conectores de 46 pines. En cuanto al sistema operativo, por defecto, es *Angstrom Linux*, pudiendo soportar *Android* y *Ubuntu*. [22]

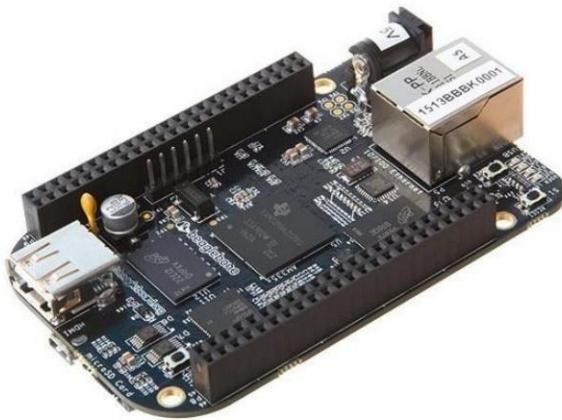


Figura 5. Beaglebone Black.

No obstante, esta placa tiene algunos inconvenientes importantes. Uno de ellos es el precio. Aunque es competitivo, es mayor que el precio de la *Raspberry Pi*. Otro inconveniente añadido es el poco recorrido que tiene esta tecnología, ya que es un producto relativamente nuevo, lo que significa que hay menos proyectos desarrollados, menos tutoriales y los errores son más difíciles de solventar.

- **Jetson TX2 de Nvidia** tiene una enorme capacidad de procesamiento. Está ideado principalmente para lidiar con la información que deben generar robots industriales, drones, cámaras inteligentes y todo tipo de dispositivos conectados para las ciudades inteligentes que están por llegar.

Este nuevo “cerebro” para máquinas se centra en el procesamiento de los datos *in situ*, en el propio aparato, pudiendo exprimir los últimos avances en inteligencia artificial, redes neuronales, reconocimiento de voz o navegación con mayor precisión y tiempos de respuesta más rápidos.

A nivel de especificaciones, el *Nvidia Jetson TX2* utiliza un procesador *Denver 2*, con una *GPU Nvidia Pascal* de 256 núcleos, 8GB de memoria RAM LPDDR4, 32GB de almacenamiento eMMC y está equipado con conectividad inalámbrica WiFi, Bluetooth y Ethernet. Todo ello en una placa de 50x87 milímetros. Este microcontrolador se orienta al sector más profesional. [23]

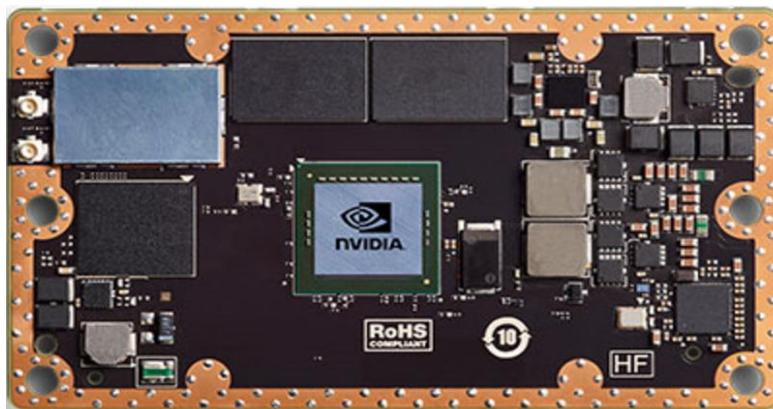


Figura 6. Jetson TX2 de Nvidia.

A continuación se muestra una tabla comparativa de los microcontroladores estudiados para realizar el proyecto. Finalmente se escoge la *Raspberry Pi* por su relación calidad-precio.

Descripción	Raspberry Pi Model B	Arduino Mega 2560	BeagleBone Black	Jetson TX2
Precio	42,90 €	35,00 €	159,00 €	400,00 €
Tamaño	8,6 x 5,4 x 1,7 cm	10,15 x 5,3 x 1,5 cm	10,2 x 5,3 x 1,6 cm	8,7 x 5 x 1,4 cm
Procesador	ARMv8	ATMega2560	ARM Cortex-A8	HMP Dual Denver + ARM A57/2
Velocidad	1,2 GHz	16 MHz	1 GHz	
RAM	1 GB	8 KB	512 MB	8 GB
Flash	SD Card	256 KB	2 GB	32 GB
EEPROM		4 KB		
Voltaje Entrada	5 V	7-12 V	5 V	5,5-19,6 V
GPIO	40	54	46	Si
PWM	2	15	4	Si
TWI/I2C	4	4	4	Si
SPI	6	4	2	Si
UART	2	4	2	Si
IDE	Linux, IDLE, Scratch	Arduino	Python, Linux, Android, Ubuntu	Linux, JetPack
Ethernet	10/100	N/A	10/100	10/100
USB	4 USB 2.0	USB 2.0	USB 2.0	USB 3.0 + USB 2.0
Video out	HDMI, CSI	N/A	HDMI	HDMI, CSI2
Audio out	HDMI, Analog	Analog	HDMI, Analog	HDMI, Analog

Tabla 2. Comparativa de microcontroladores estudiados.

Las características y especificaciones técnicas de la *Raspberry Pi* se detallarán a continuación, en el apartado 3.2.2. *Raspberry Pi 3 Model B*, dentro del apartado 3.2. Arquitectura Hardware.

## 2.2.3. DISEÑOS DE VEHÍCULOS AUTÓNOMOS.

Debido a la multitud de variantes que proporciona la conducción autónoma, se están estudiando y desarrollando diferentes diseños y aplicaciones de vehículos autónomos. Se diferencian tres diseños de vehículos, dependiendo de la distancia que sean capaces de cubrir y de su uso. Estos son: vehículos de corta distancia, media distancia y multiuso.

### 2.2.3.1. VEHÍCULOS DE CORTA DISTANCIA.

De uso principalmente en ciudades, en zonas cerradas de grandes dimensiones o para transportar desde zonas incomunicadas a las zonas más cercanas con transporte público, ya que la media de distancia de sus trayectos estará por debajo de 20-25 Km. Se pretende que tengan una gran maniobrabilidad, para poder circular por ciudades, y que tengan un bajo coste, al ser un coche de pequeño tamaño creado para compartir por 1-2 personas y muy limitado de espacio para cargar equipaje. Tendrán como puntos fuertes su bajo consumo de combustible, bajas emisiones contaminantes y muy fiables ante las roturas o averías, reduciendo así su mantenimiento. [9]

Como gran ejemplo de este tipo de diseño tenemos al vehículo autónomo de *Waymo* (empresa asociada a *Google* y por lo tanto antiguo vehículo *Google*), estudio que comenzó en 2009. Este vehículo está programado actualmente para alcanzar los 40 Km/h y realizar trayectos de 160 Km como máximo. En Mayo de 2017 alcanzó 4,5 millones de Km de experiencia de conducción autónoma, aunque todavía tiene muchos detalles por solucionar. La experiencia que tiene este sistema es equivalente a 400 años de conducción de una persona, considerando las horas que ha pasado en la carretera. [24]

El coche autónomo de *Google* conduce por sí solo, reconoce los carriles, las señales de tráfico y los semáforos, sabe que llega a un cruce, ve a los otros vehículos, ciclistas y peatones, controla la distancia de seguridad con el vehículo que va delante y toma las decisiones pertinentes para no tener ningún percance. Es capaz de detectar cualquier peligro desde su posición hasta una distancia equivalente a dos campos de futbol, en todas las direcciones. [25]

No sólo tiene la capacidad de detectar peligros, además puede predecir movimientos o actitudes que realizarán o tomarán en un futuro los demás usuarios de la carretera, ya que han enseñado al sistema a conducir gracias a la cantidad de Km que lleva de prueba.

No se puede decir que un elemento del conjunto de tecnologías que incorpora sea más importante que otro ya que es la combinación y complementación de todos ellos los que permiten que el sistema funcione con garantías y seguridad. La mayoría de ellos, además, ya existen hoy en día y se implementan de manera independiente en diferentes coches.

Los elementos más importante y destacables del vehículo probablemente sean el **LIDAR** (*Laser Imaging Detector and Ranging*) un dispositivo, situado en el techo del coche, que detecta objetos y mide la distancia a ellos mediante rayos de luz, concretamente haces láser, y la cámara, situada en mitad de la parte superior del parabrisas (véase figura 7). Gracias al LIDAR se puede construir una imagen tridimensional alrededor del coche, con todos los objetos que nos rodean,

posicionados. Mientras que la cámara permite el reconocimiento de señales de tráfico, semáforos, líneas de carretera, vehículos y todo lo que conforma el tráfico, lo que permite al vehículo poder circular sin ningun percance.



Figura 7. Vehículo Waymo.

Pero estos dos elementos no servirían de mucho por sí solos, por ello se complementan con mapas detallados, GPS y una unidad de medición inercial, para saber con precisión hacia dónde se mueve el vehículo. También incorpora tecnología actualmente en uso en algunos vehículos, como el sistema de control de velocidad de crucero, combinado con acelerador y frenos electrónicos, que permite al computador acelerar en mayor o menor medida dependiendo de su criterio.

### 2.2.3.2. VEHÍCULOS DE MEDIA DISTANCIA.

Destinados para trayectos de media-larga distancia, podrán cubrir distancias por encima de los 25 Km, y por ello serán vehículos de tamaño grande y de alta comodidad, similares a los autobuses. Debido al aumento de espacio, se consigue una mayor capacidad, tanto de pasajeros (más de 4 personas), como de equipaje y carga. Al igual que el diseño anterior, tendrán un bajo consumo de combustible, bajas emisiones y serán muy fiables. En este apartado se pueden encontrar varios ejemplos que están en procesos de pruebas debido a la mayor dificultad de implementar esta tecnología.

El primer ejemplo es el minibus autónomo llamado *Olli*, desarrollado por la empresa *IBM* y construido por *Local Motors* en su mayor parte con piezas impresas en 3D, tiene capacidad para 12 personas y es totalmente eléctrico. Basado en el sistema de inteligencia artificial y cognitiva

*Watson Internet of Things (IoT) de IBM*, es capaz de aprender de su entorno, del tráfico y de la meteorología para adaptar su conducción a estos parámetros y optimizar el recorrido.



Figura 8. Olli, desarrollado por IBM.

Por parte de *IBM* aseguran que es el sistema más inteligente en la actualidad, tanto que incluso puede alterar el recorrido habitual si detecta alguna emergencia o responder a las preguntas de los viajeros durante el trayecto. En la actualidad no supera los 15 Km/h en sus trayectos realizados en Estados Unidos, debido a que se encuentra en pruebas, pero se espera que alcance velocidades superiores. Cuenta con más de 30 sensores, entre los que destacan las cámaras ópticas, GPS y su sistema LIDAR, además el sistema siempre estará monitorizado por parte de un equipo. *Olli* no tiene acelerador ni freno convencionales, el sistema maneja la velocidad a su antojo y a diferencia del vehículo de la compañía *Waymo*, carece de volante. [26]

Otro ejemplo, mucho más cercano, es el autobús eléctrico *EZ10*, anteriormente *WePod*, que es capaz de circular sin conductor por algunas ciudades de Holanda, Finlandia, Suiza y Alemania. Esta iniciativa creada por la empresa francesa *EasyMile* realizó su primer viaje en Holanda, cubriendo un trayecto de 200 metros a 8 Km/h con 6 pasajeros a bordo. Una vez conseguido ser el primer autobus autónomo capaz de llevar pasajeros de un punto a otro, fue mejorando prestaciones y actualmente es capaz de desplazarse a una velocidad media, no muy elevada, de 25 Km/h pudiendo llegar a alcanzar los 40 Km/h. Es capaz de transportar a 12 pasajeros en rutas fijas de 20 Km, circulando entre el tráfico real. [27]



Figura 9. EZ10. Primer autobus autónomo.

Para poder moverse en consonancia con el tráfico, el *EZ10* utiliza tecnología *Nvidia* y es necesario que los pasajeros lleven un teléfono móvil para comunicarse con el vehículo. Este particular autobús está equipado con un equipo técnico de última generación. Un conjunto de cámaras de 360º, radares, láseres y sistemas GPS rastrean la posición del vehículo respecto al tráfico que lo rodea. Las cámaras se utilizan para mapear puntos de referencia en el entorno, funcionando como un mecanismo de navegación alternativo que mejora la precisión del GPS. De esta forma, el autobús puede sortear los obstáculos inesperados o reconocer lo que le rodea gracias al sistema de *deep learning*. Pese a toda esta tecnología, este autobús aún necesita mejorar, ya que mientras esté en periodo de pruebas, si hay problemas meteorológicos, es de noche o hay atasco, el vehículo no podrá circular y tendrá que interrumpir su servicio. Por último, destacar que es un proyecto de sostenibilidad apoyado por la Unión Europea con un coste de 3,5 millones de euros. [28], [29]

Como último ejemplo, el miniautobús *Harry*, que se encuentra en fase de test desde mayo de 2016. El desarrollo de esta tecnología ha sido realizado por el Laboratorio de Investigación de Transporte de Reino Unido (TRL) mediante el proyecto *GATEway*. *Harry* ya está funcionando de forma gratuita en el barrio londinense de Greenwich, donde transporta a 4 pasajeros dentro de una ruta preestablecida de 4 Km. El transporte realiza sus viajes a una velocidad de 16 Km/h y utiliza 5 cámaras y 3 sensores LIDAR que le ayudan a analizar los diferentes obstáculos y estímulos del exterior. Puede ver hasta a 100 metros de distancia y es capaz de parar si detecta un obstáculo en su camino. Según los diseñadores, por el momento, está diseñado para ser seguro en entornos peatonales. Por ello en esta fase de pruebas necesita un carril específico para su circulación.



Figura 10. Proyecto de GATEway.

Durante el periodo de pruebas siempre habrá un experto a bordo por si surgiese cualquier imprevisto. Se ha realizado una inversión de 11.5 millones de euros y se espera que en 2019 ya haya una pequeña flota de *Harrys* circulando por la ciudad. [30]

A continuación se muestra una tabla resumen de los vehículos de media distancia explicados.

Características	Olli	EZ10	GATEway
<b>Empresa</b>	IBM	EasyMile	TRL
<b>Velocidad</b>	15 - 40 Km/h	25 - 40 Km/h	16 Km/h
<b>Km</b>	Pruebas	20 Km	4 Km
<b>Cámaras 360</b>	Si	Si	Si
<b>GPS</b>	Si	Si	Si
<b>LIDAR</b>	Si	Si	Si
<b>Capacidad</b>	12 plazas	12 plazas	4 plazas
<b>Inteligencia</b>	Alta	Media	Baja
<b>Sistema Inteligencia</b>	Watson (IBM)	Nvidia	-
<b>Situación</b>	Pruebas	Circulando con tráfico	Circulando carril único
<b>Inversión</b>	Desconocida	3,5 millones de Euros	11,5 millones de Euros

Tabla 3. Resumen vehículos de media distancia.

### 2.2.3.3. VEHÍCULOS MULTIUSO.

Es el diseño más orientado a los vehículos actuales, de uso personal, que tendrán un diseño personalizado y podrán ser utilizados para viajes con varios pasajeros. Serán capaces de cubrir cualquier distancia en sus trayectos, transportando de 1-5 pasajeros y con una carga media. Con estos vehículo se gana en confort, privacidad, tendrán un servicio específico *online* para cada vehículo y mantendrán las ventajas de los dos diseños anteriores en cuanto a emisiones, consumo y fiabilidad.

Para este tipo de diseño hay varios ejemplos. El primer ejemplo es altamente conocido y predomina sobre los demás, por ahora. Este es el vehículo autónomo de la empresa americana *Tesla, Inc.* El vehículo *Tesla* lleva un motor totalmente eléctrico fabricado por la propia empresa, 8 cámaras que ofrecen una visión de 360 grados alrededor del vehículo y vigilan un área de hasta 250 metros. Existen tres cámaras frontales con distintas funciones dependiendo de la velocidad a la que se circule y el entorno, cuatro laterales divididas en dos laterales mirando hacia delante y dos mirando hacia atrás para controlar los ángulos muertos, y por último tiene una cámara de visión trasera.

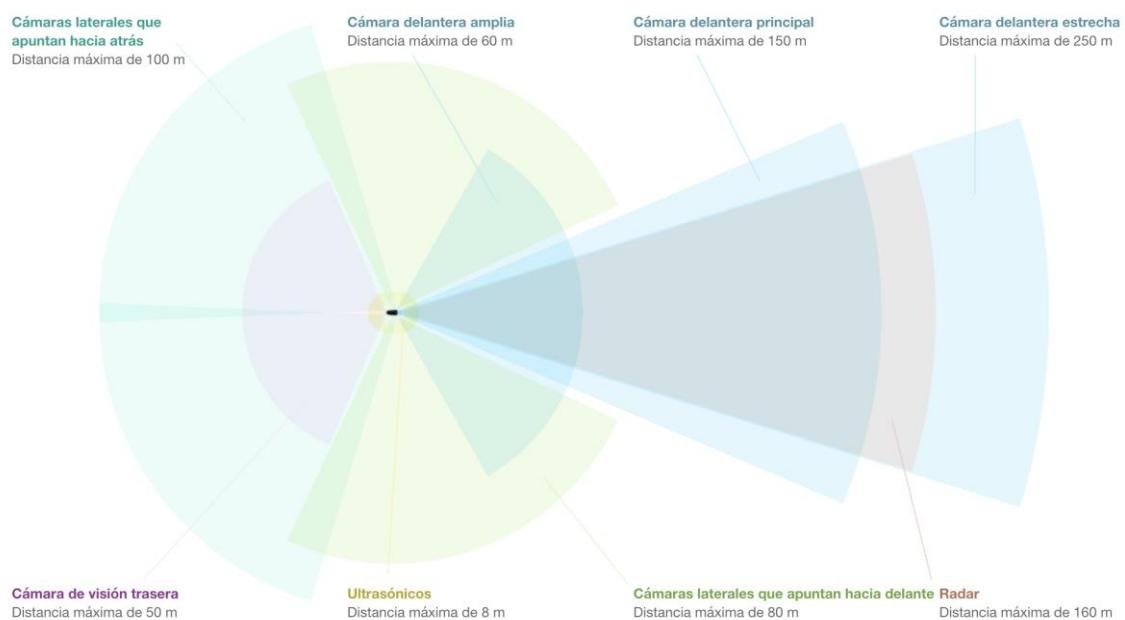


Figura 11. Dispositivos del sistema.

Además, 12 sensores ultrásonicos son capaces de detectar objetos de todo tipo y tamaño, delante, detrás o a los lados. En tercer lugar, un radar delantero ofrece datos adicionales, incluso con fuertes lluvias, niebla o polvo. Finalmente toda esta información está gestionada por un ordenador de a bordo, fabricado por *Nvidia*, que cuenta con herramientas muy potentes de procesamiento visual, construidas dentro de una profunda red neuronal. Este procesador, llamado *Hardware 2*, es 40 veces más potente que el anteriormente instalado *Hardware 1*, en los vehículos de la empresa americana pero inicialmente seguirá teniendo mejor funcionamiento *Hardware 1*, ya que *Hardware 2* se encuentra en sistema de aprendizaje y necesitará más tiempo para ir mejorando (*deep learning*). Por el momento, cuenta con volante y pedales (véase la figura 12), aunque se espera que en un futuro se supriman.



Figura 12.Visión desde el interior y cámaras en funcionamiento.

El vehículo de *Tesla* tiene una capacidad de 1-5 pasajeros y puede realizar trayectos largos que sólo estarán limitados por la autonomía del motor eléctrico, que variará dependiendo de la velocidad. Con una velocidad media de 70 Km/h el *Tesla* puede alcanzar los 849 Km, mientras que a su velocidad máxima, 120 Km/h, su autonomía es de 471 Km. Actualmente *Tesla* está dotando a sus vehículos de todos estos componentes que hemos mencionado porque, aunque aún no son totalmente autónomos (se encuentran en la etapa 3, de conducción autónoma limitada), pretenden, en 2019, alcanzar la última etapa y ser totalmente autónomos, teniendo que instalar únicamente la nueva actualización de su software de conducción autónoma *Autopilot*. Las funciones de conducción autónoma de este vehículo son: conducción autónoma en autopista, aparcamiento automático, ayudas en la conducción (ángulos muertos, aviso de semáforos, señales, etc.) y *Smart Summon*. [31]

Sin embargo, existen muchos más proyectos de vehículo multiusos autónomos, como el de *Ford*, centrado en el LIDAR y las cámaras, al igual que *Toyota*, mientras que *Volvo* ya tiene un control de conducción autónoma en autopistas y pretende tener coches totalmente autónomos en 2021, al igual que la alianza formada por *BMW e Intel* y la formada por *Renault-Nissan* y *Microsoft*. *Mercedes* también tiene un proyecto muy avanzado, ya que dispone del nuevo *Clase E*, que es capaz de conducir por sí solo en carreteras tranquilas y adelantar si el conductor se lo pide, aunque en caso de que el sistema encontrase dificultades cedería los mandos al conductor. También tiene el sistema de cambio de carril activo y asistente de frenada. [32]

Hay que añadir que *Audi* también tiene un sistema muy completo y en 2020 pretende sacar un vehículo totalmente autónomo con procesadores *Nvidia*. Por último, existen casos negativos como el de *Uber*, que ha abandonado su proyecto debido a grandes fallos en su sistema que acabaron en accidentes y atropellos de peatones y ciclistas.

A continuación se muestra un resumen de los diseños mencionados:

	Vehículo de distancias cortas	Vehículos de distancias medias	Vehículos multiusos
<b>Uso principal</b>	Viajes cortos en ciudades y suburbios	Viajes de media-larga distancia en ciudades y suburbios	Viajes personales y de ocio
<b>Característica principal</b>	Bajo coste y buena maniobrabilidad	Alta comodidad para viajes más largos	Buena experiencia y confort en viajes personales
<b>Distancia media</b>	15-20 Km	Mayormente por encima de 15-20 Km	Cualquier distancia
<b>Tamaño</b>	Pequeño	Medio-largo	Medio
<b>Capacidad</b>	1-2 pasajeros	4+ pasajeros	1-5 pasajeros
<b>Espacio Carga</b>	Limitado	Grande	Medio
<b>Propiedad</b>	Uso compartido	Uso compartido	Uso personal
<b>Áreas de uso</b>	Ciudades y suburbios	Ciudades y suburbios	Cualquier área
<b>Atributos Importantes</b>	<ul style="list-style-type: none"> <li>- Emisiones bajas</li> <li>- Bajo consumo</li> <li>- Bajo mantenimiento</li> <li>- Fiabilidad</li> </ul>	<ul style="list-style-type: none"> <li>- Comodidad</li> <li>- Emisiones bajas</li> <li>- Bajo consumo</li> <li>- Bajo mantenimiento</li> <li>- Fiabilidad</li> </ul>	<ul style="list-style-type: none"> <li>- Individualización</li> <li>- Servicios Online</li> <li>- Comodidad</li> <li>- Emisiones bajas</li> <li>- Bajo consumo</li> <li>- Bajo mantenimiento</li> <li>- Fiabilidad</li> </ul>

Tabla 4. Resumen de los diseños de vehículos autónomos.

## **2.2.4. FUTURO DE LA TECNOLOGÍA.**

La tecnología autónoma está avanzada, pero se espera que en los próximos años todos los vehículos sean espacios de confort y ocio donde no haya que preocuparse de la conducción. Todos los fabricantes siguen un mismo camino y, en el futuro, los vehículos serán totalmente eléctricos, sin volante, ni pedales de aceleración ni freno, con un sistema autónomo que aprende de las experiencias gracias a las redes neuronales y, aunque en la actualidad se piensa que el sistema más seguro es el basado en el LIDAR (práctico para bajas velocidades), en el futuro predominarán los sistemas basados en la visión artificial gracias a la implementación de técnicas de *deep learning* con redes neuronales evolucionales para la construcción de modelos que permitan el reconocimiento de objetos y una segmentación semántica de las imágenes.

Además, *Volvo* ha conseguido crear un camión autónomo que es capaz de realizar trayectos en una mina y está desarrollando un sistema conocido como tren de carretera (proyecto *SARTRE*), que consiste en formar una caravana de vehículos que se comunican inalámbricamente, pasándose de manera constante datos de posición, velocidad, distancia mínima de frenado, etc.

Según la opinión de Mark Halverson, CEO de la compañía de transporte automático *Precision Autonomy*, la implementación de tecnologías *vehicle-to-vehicle* (V2V) facilitaría la reducción de accidentes, ya que los vehículos podrían comunicarse entre ellos e informar unos a otros dando información sobre dónde se dirigen (no tiene porque ser el destino, puede ser información intermedia como, por ejemplo, la trayectoria estimada los próximos 30 segundos), permitiendo así recalcular las trayectorias sabiendo las de los demás. Para ello, será necesario primero superar los problemas burocráticos que existen a nivel legal y donde un nuevo debate ético tendrá lugar. [33]

## 2.3. APLICACIONES Y TECNOLOGÍAS SIMILARES.

Los avances tecnológicos han propiciado el desarrollo de un gran número de aplicaciones y tecnologías similares, diferenciadas en los elementos y las funciones que las componen.

### 2.3.1. ROBOT SIGUE LINEA INFRARROJOS.

Este tipo de robot móvil es un robot muy simple, no necesita de un microcontrolador principal, ya que el control se puede realizar con puertas lógicas y un circuito integrado. Estará compuesto por un chasis, un sistema de locomoción tipo triciclo, incluso bastaría con un sistema de dos ruedas motrices y dos sensores infrarrojos. Tras obtener la información proporcionada por los dos sensores, la diferencia del ancho que capten uno u otro, se traslada al circuito de puertas lógicas, devolviendo la dirección hacia donde debe dirigirse el robot.

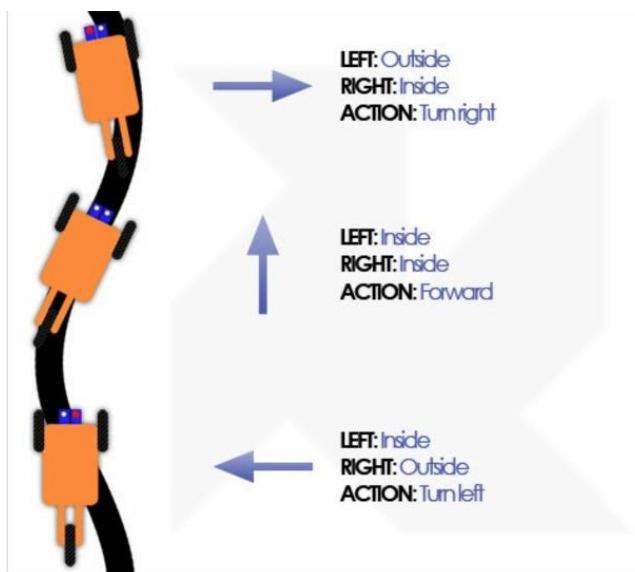


Figura 13. Robot sigue línea infrarrojos.

### 2.3.2. ROBOT SIGUE LÍNEA POR CÁMARA.

En este caso, es un robot que desempeña la misma función que el robot anterior pero con un sistema mucho más complejo. Este robot estará compuesto por una cámara, en lugar de los dos sensores infrarrojos y esta vez si que contará con un microcontrolador, que procesará las imágenes recibidas. La placa central será la encargada de tratar las diferentes imágenes con la librería *OpenCV* o *SimpleCV*. Existen varios procesos para la detección de línea, ya que se puede realizar distinguiendo entre el color de la línea y los demás, haciendo una distinción de contornos, o utilizando funciones específicas para distinguir segmentos de líneas rectas.

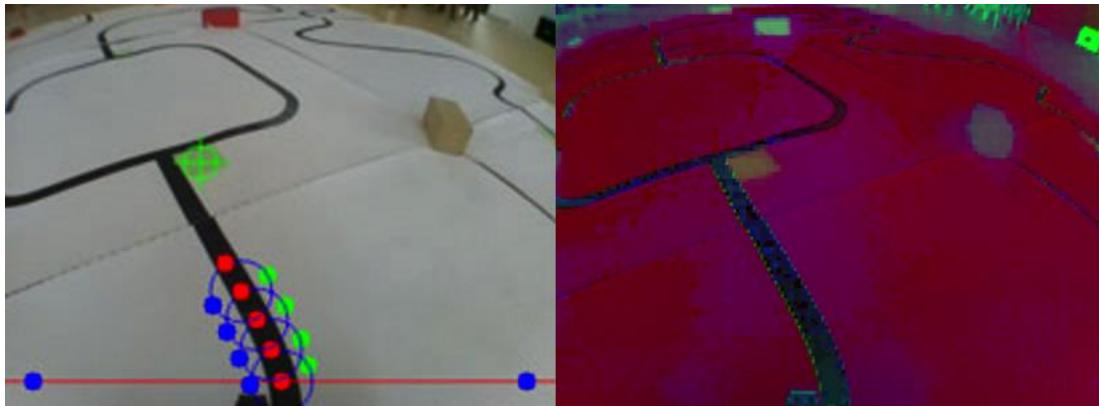


Figura 14. Detectando la línea con OpenCV

### 2.3.3. ROBOT CONTROLADO CON SISTEMA ANDROID.

Utilizando los avances de las tecnologías se puede conseguir crear un robot que siga las instrucciones ordenadas desde un móvil con sistema operativo *Android*. El robot, en este caso, estará compuesto por un microcontrolador con acceso bluetooth o wifi, una cámara, y se podrían añadir sensores para mejorar las prestaciones. Una vez exista la conexión del móvil con la placa, ya sea por bluetooth o por red wifi, se le pueden enviar órdenes o tareas. Existen distintas formas de enviarle estas tareas: algunos desarrolladores lo han conseguido mediante notas de voz, otros mediante una aplicación, y en otras ocasiones mediante el envío de una fotografía, para que la analice y encuentre esa figura en la estancia. En el caso de que el robot necesitase realizar ciertos movimientos, podría realizarlos gracias a la visión artificial y el software que ejecutará el microcontrolador.

### 2.3.4. COCHE AUTÓNOMO RC.

Un grupo de estudiantes ha sido capaz de crear un vehículo radiocontrol totalmente autónomo. Está compuesto de cámaras que van procesando las distintas imágenes que va recibiendo, sensores ultrasónicos e infrarrojos, un giroscopio y dos microcontroladores, una *Raspberry Pi* y una placa *Arduino*. Utiliza algoritmos de visión artificial, junto con los datos proporcionados por sus sensores de a bordo, para conseguir que el vehículo siga los carriles de una calle, realizar maniobras de aparcamiento y mantenerse a salvo de obstáculos. Las placas analizan los datos obtenidos por los diferentes sensores en el vehículo autónomo y los envia por vía bluetooth al teléfono móvil de a bordo. [34]

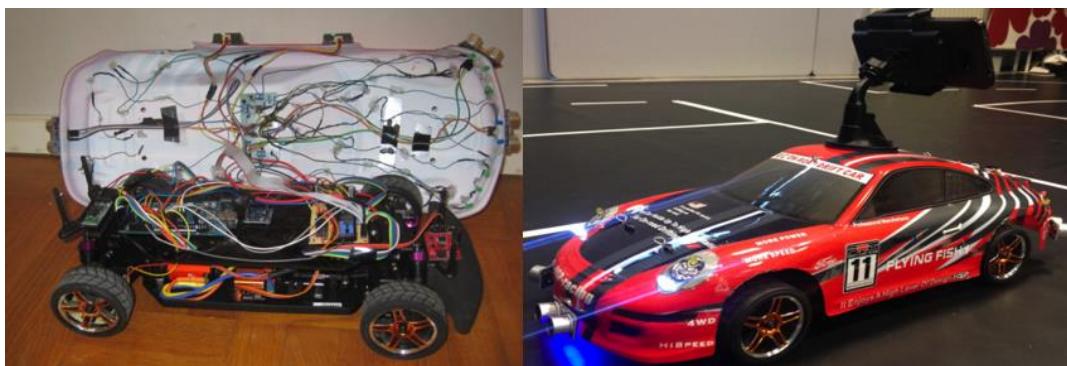


Figura 15. Coche autónomo RC.

### 2.3.5. SISTEMA DE DETECCIÓN DE SEÑALES DE TRÁFICO (TSR).

El sistema, creado por *BMW* en 2008, es un avisador automático de los límites de velocidad, señales de prohibición, y también puede ser un avisador para cuando se realiza un cambio de carril involuntariamente. Estas opciones varían dependiendo del fabricante.

La solución para detectar y reconocer las diferentes señales pasa por el uso de una cámara, de alta resolución, y presenta al conductor los datos en tiempo real, pudiendo consultar todas las limitaciones en activo en una pantalla digital del cuadro de mando, si en algún momento lo necesita. Este sistema combina la información captada con los datos de otros dispositivos como el sistema de navegación, el reloj o el sensor de lluvia, antes de mostrar cualquier restricción.

Sin embargo, disponer de esta tecnología no implica que el conductor pueda permitirse un despiste, sino que actúa a modo de recordatorio, contribuyendo así a mejorar la seguridad propia y del resto de conductores. [35]



Figura 16. Aviso en el salpicadero de las señales próximas.

### 2.3.6. LDW Y LKS.

Conocido por sus siglas *LDW* (*Lane Departure Warning*), el detector o avisador de cambio de carril es un sistema de seguridad que detecta la variación de trayectoria del vehículo sobre el carril e interpreta cuando ésta es involuntaria, para avisar al conductor y evitar así colisiones con otros vehículos o que se produzca una salida de la vía. En los sistemas más avanzados, el sistema es capaz de corregir la trayectoria cuando el conductor no responde a las alertas que le envía el sistema. [36]

Para detectar la desviación de la trayectoria, el *LDW* emplea cámaras que siguen el trazado de las marcas viales. La cámara suele ir colocada en el parabrisas y va realizando una lectura de las marcas viales longitudinales, ya sean líneas discontinuas o continuas, e informando al sistema central del *LDW*, véase figura 17).

Dependiendo de la velocidad, del grado de giro del volante, de la activación o no de los intermitentes, en función de los parámetros que tenga programados, el sistema determina que el conductor ha perdido la trayectoria del carril y le avisa, normalmente con una señal acústica y otra visual que se aprecia en el panel de instrumentos. De forma adicional, hay modelos en los que el sistema hace vibrar el volante o incluso el asiento para alertar al conductor del riesgo de pérdida de control del vehículo.

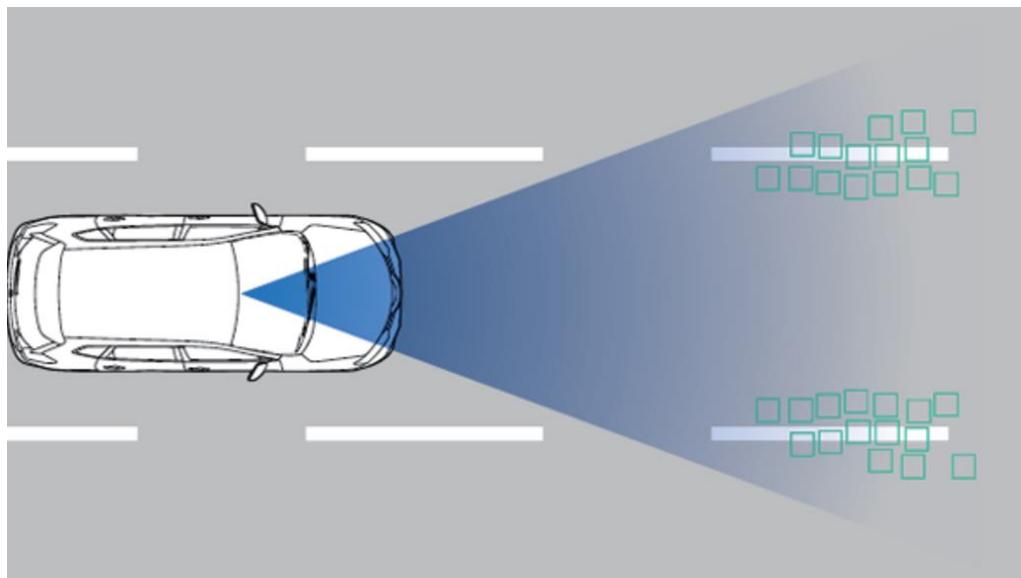


Figura 17. Funcionamiento LDW.

El sistema *LKS* (*Lane Keeping System*) es la evolución del sistema *LDW*. Mientras que el *LDW* se dedica a detectar el problema y avisar de él, el sistema *LKS* tiene por misión mantener el coche en el carril, actuando sobre la columna de la dirección para girar las ruedas de manera que el vehículo se mantenga en el trazado y avisa a los conductores cuando no se detectan sus manos en el volante.

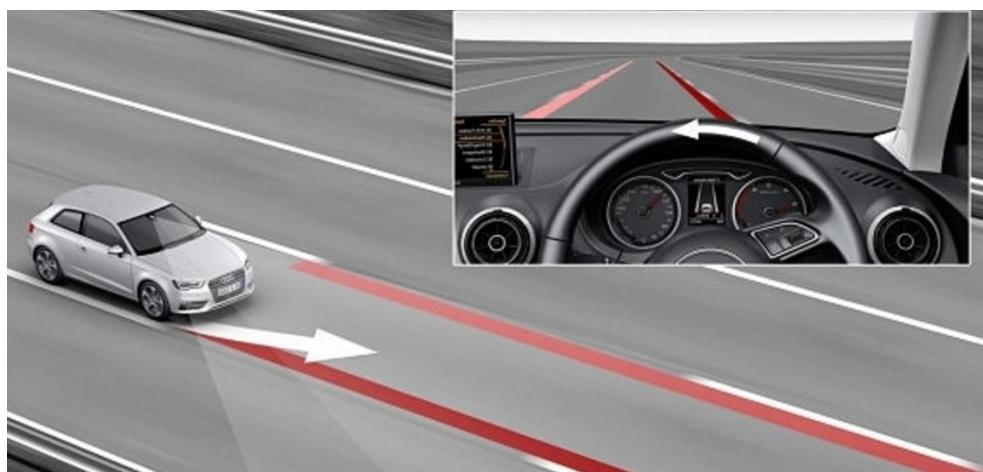


Figura 18. Funcionamiento LKS.

Sin embargo, en la práctica, estos sistemas tienen un problema común, y es que no siempre son capaces de detectar la desviación de la trayectoria, bien porque las marcas del trazado no sean lo suficientemente visibles o bien porque el sensor no sea capaz de detectarlas.

### 2.3.7. OTRAS TECNOLOGÍAS.

Hay varias tecnologías autónomas y automatizadas que utilizan microcontroladores como la *Raspberry Pi*, como pueden ser el *Coconut Pi*, que es un robot sumergible autónomo. Otra de las funciones de la *Raspberry Pi*, combinada con una cámara y las librerías de *OpenCV*, es controlar el vuelo de un dron y así convertirlo en un UAV, véase figura 19. También se puede convertir la placa en un centro de seguridad, en un escáner 3D autónomo y podemos realizar el control autónomo de riegos.



Figura 19. Detección de un objetivo con un dron.

# CAPÍTULO 3. SISTEMA PROUESTO.

En el presente capítulo se detalla el sistema propuesto, así como la arquitectura de hardware y software utilizado.

## 3.1. DESCRIPCIÓN GENERAL.

El sistema propuesto se basa en un robot móvil de locomoción diferencial, es decir, no tiene ruedas directrices, equipado con una cámara, *RaspiCam*, la cual irá obteniendo imágenes del entorno que serán analizadas por controlador principal (*RaspberryPi 3 Model B*). A su vez la *Raspberry Pi* se comunicará con el controlador de motores por los pines *GPIO*, el cual enviará las señales necesarias a los motores.

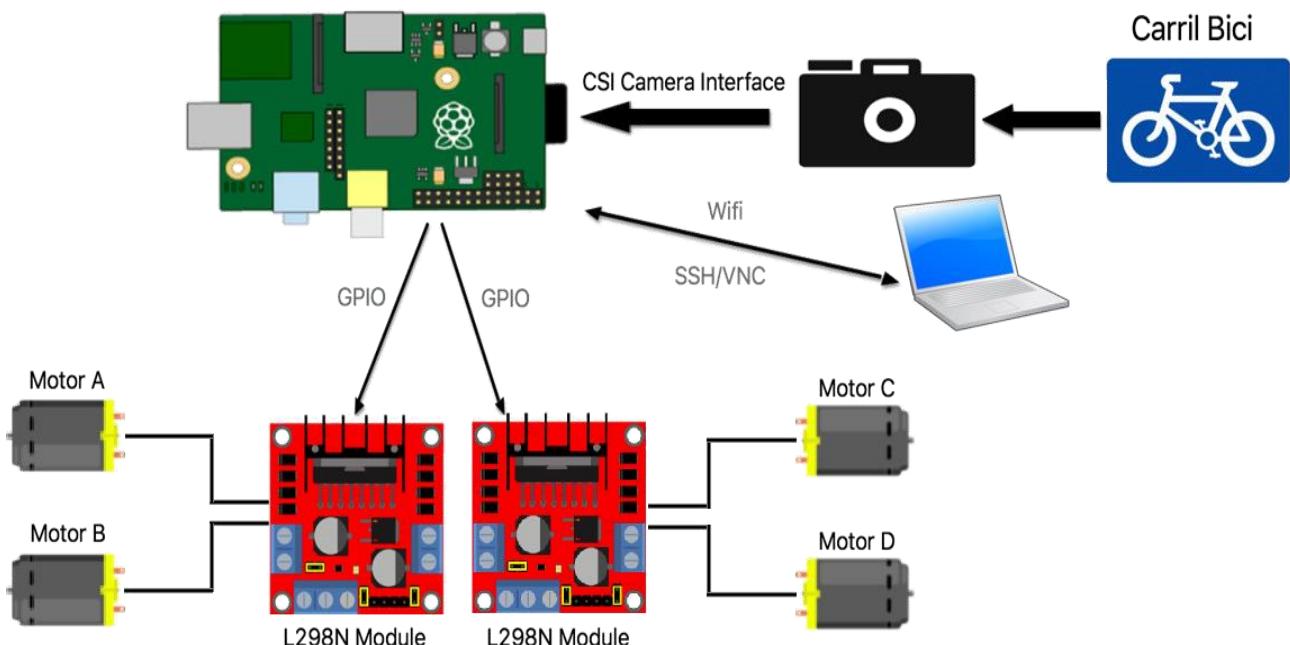


Figura 20. Esquema del sistema propuesto.

Como se puede observar en la figura 20, las imágenes son obtenidas por la cámara, que serán enviadas y analizadas directamente por la *Raspberry Pi*. Las imágenes serán analizadas gracias a un software creado en el lenguaje de programación *Python* utilizando las librerías de *OpenCV*. Una vez que hayan sido analizadas, el software habrá obtenido un resultado que será la dirección a seguir en ese instante. Dependiendo de la dirección que se deba seguir, el controlador principal enviará distintas señales a los motores, los cuales están conectados a la *Raspberry Pi* a través de los controladores de motores. Esto es posible gracias a la utilización de librerías *GPIO*.

A continuación se puede ver el posicionamiento de los distintos elementos en el chasis del robot (figura 21) y la vista superior del robot (figura 22).

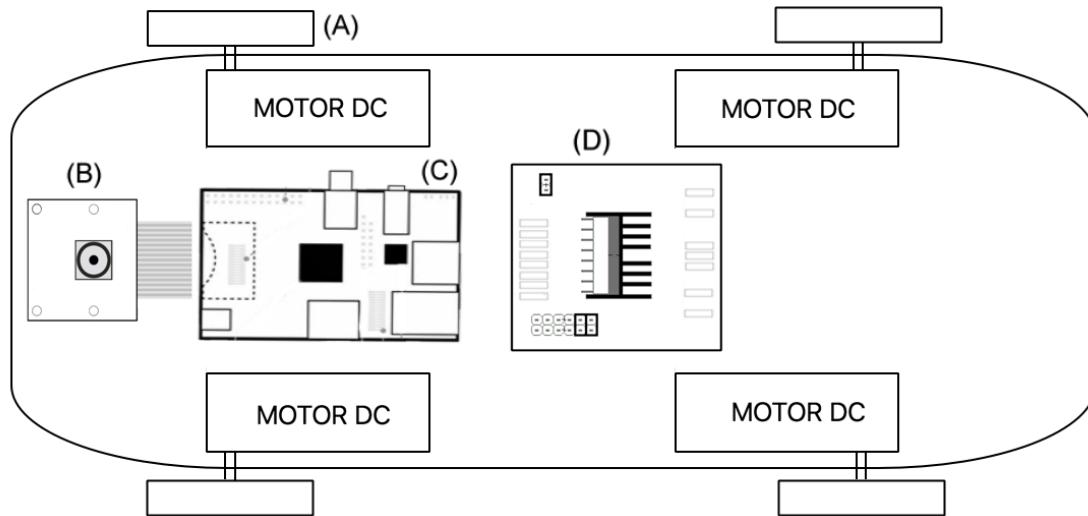


Figura 21. Diagrama del robot: (A) Ruedas; (B) RaspiCam; (C) Raspberry Pi; (D) Módulo L298N, controlador de motores.

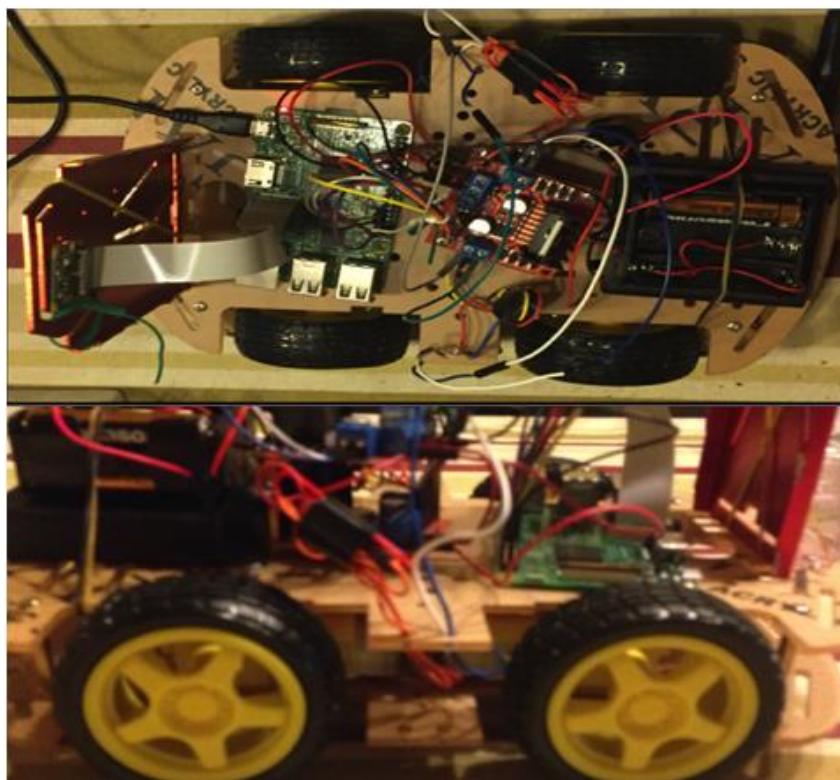


Figura 22. Vista superior y lateral del robot.

Por último, en la siguiente página se puede ver el conexionado eléctrico del sistema propuesto.

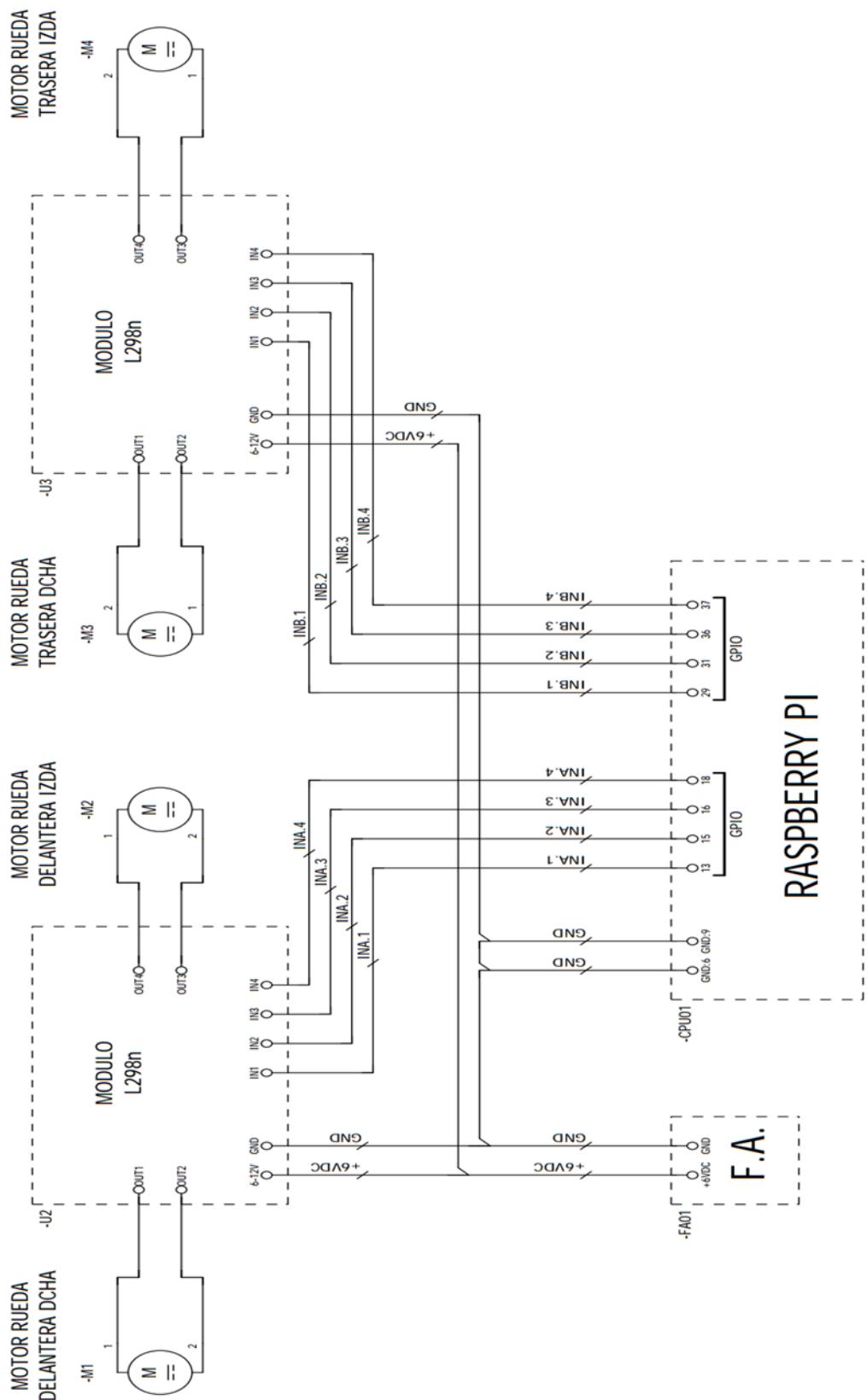


Figura 23. Conexionado eléctrico.

## 3.2. ARQUITECTURA HARDWARE.

En este apartado se procede a explicar los elementos utilizados en el proyecto, además de su arquitectura.

### 3.2.1. RASPBERRY PI CAMERA BOARD v2.

Es una cámara de 8 Megapíxeles capaz de capturar video e imágenes a 1080 píxeles. Está conectada a la *Raspberry Pi*, directamente por su cable ribbon, al conector CSI (*Camera Serial Interfaz*), véase figura 23. El bus CSI es capaz de transmitir grandes cadenas de datos y sólo envía los datos al procesador. La cámara es muy pequeña y pesa apenas 3 gramos, por lo que es perfecta para aplicaciones de robots móviles. Aunque la cámara pueda grabar a 1080 píxeles no se ha utilizado esa resolución porque convertiría el análisis en un proceso muy lento, por lo tanto se ha utilizado menos resolución. La cámara no puede grabar sonido, debido a que no tiene incorporado un micrófono.

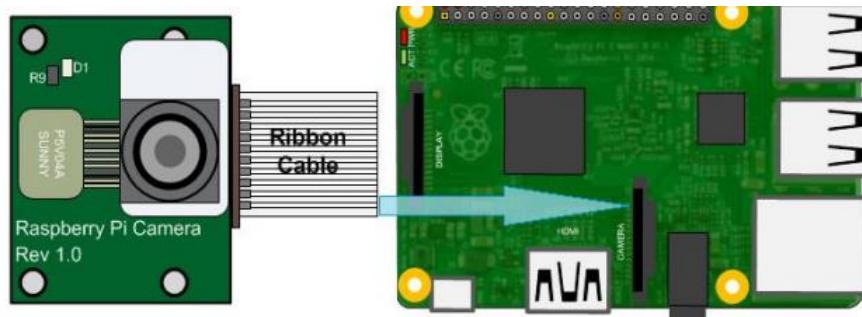


Figura 24. Conexión de RaspiCam en Raspberry Pi.

Se decidió elegir esta cámara y no cámaras USB convencionales y de menor coste por la compatibilidad con la *Raspberry Pi*, ya que es del mismo fabricante y tiene una fácil instalación, necesitando sólo descargar unas librerías, y no necesita drivers.



Figura 25. RaspiCam.

### Características más importantes [37]:

- El sensor de imagen utilizado es un *Omnivision 6547 CMOS*.
- Resolución de 5 Megapíxeles.
- Resolución de imagen fija de 2592x1944 píxeles.
- Tasa máxima de transferencia de imágenes a 1080 píxeles de 30 *frames* por segundo.
- Tasa máxima de transferencia de imágenes a 720 píxeles de 60 *frames* por segundo.
- Funciones de control de imagen
  - o Control de exposición automático
  - o Balance de contraste automático
  - o Filtro de banda automático
  - o Detección de luminancia 50/60 Hz automática
  - o Calibración de nivel de oscuridad automática
- Rango de temperatura de -20°C a 60°C
- Tamaño de la lente de 1,4 "
- Dimensiones: 23.86 x 25 x 9 milímetros
- Peso: 3 gramos
- 

### 3.2.2. RASPBERRY PI 3 MODEL B



Figura 26. Raspberry Pi 3 Model B+.

*Raspberry Pi 3 Model B* es una placa microcontrolador originalmente desarrollado con el objetivo de estimular la enseñanza informática en los colegios y diseñada para tener un bajo coste. El hardware tiene un tamaño algo más pequeño al de la placa *Arduino*, siendo de 8,6 centímetros de largo, 5,4 centímetros de ancho y 1,7 centímetros de grosor. Tiene como gran ventaja la variedad de sistemas operativos que puede soportar, al igual que los lenguajes de programación (*C*, *C++*, *Python*). Esta placa es la elegida para realizar el proyecto por la combinación de buenas características y un precio bastante asequible. Dispone de un procesador *ARMv8*, un procesador de cuatro nucleos a 1,2 GHz de 64 bits, un 1 GB de RAM, 40 I/O digitales, 2 de las cuales tienen salida PWM, 2 UARTS (puertos de comunicación Serie Tx Rx), 4 conexiones USB, puerto Ethernet, conexión HDMI, conexión audio jack, CSI (*Camera Interface*)

y DSI (*Display Interface*). Además, lleva incorporado un módulo WIFI 802.11n Wireless LAN y un módulo Bluetooth 4.1 Low Energy y tiene entrada para tarjeta micro SD.

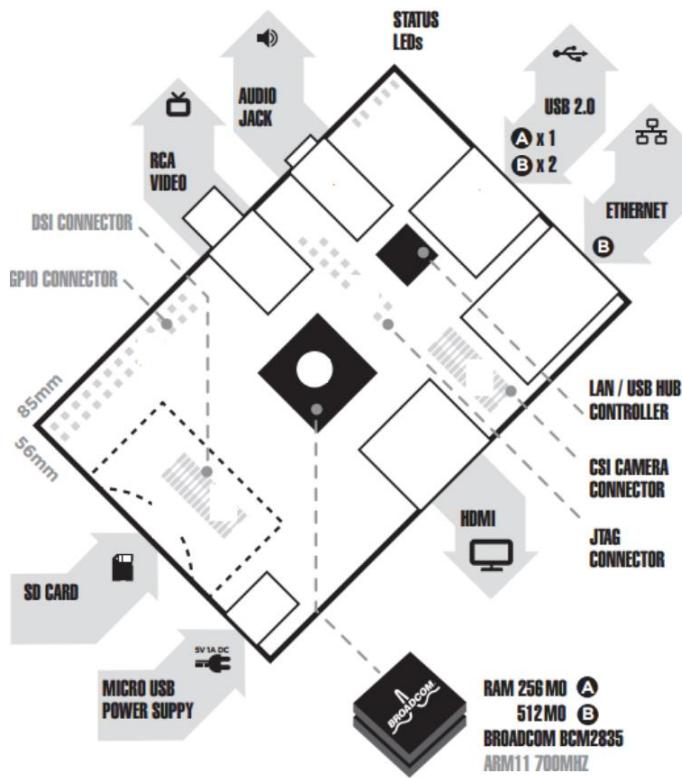


Figura 27. Esquema Raspberry Pi.

Para la comunicación usando pines no definidos como UARTS es necesario el uso de la librería *WiringPi* y tener especial cuidado con la diferencia de voltajes de las conexiones.

#### Entradas y Salidas

- Todas las entradas y salidas no tienen ninguna protección de circuitería, por lo que hay que ir con cuidado a la hora de conectar los diferentes dispositivos.
- Los pines [8,10] disponen de las funciones para la comunicación Serie Tx Rx
- Los pines [3,5,27,28] son para las conexiones I2C
- PWM de los pines [32,33]
- Pines SPI, 21 y 35 (MISO), 38 y 19 (MOSI), 40 y 23 (SCLK)
- Alimentación 5V, pines [2,4]
- Alimentación 3,3V, pines [1,17]
- Tierra [6, 9, 14, 20, 25, 30, 34, 39]. Los pines 6 y 9 serán utilizados en el proyecto
- Los pines 13, 15, 16, 18, 29, 31, 36 y 37 serán los pines GPIO utilizados para las conexiones con los controladores de motores.

### 3.2.3. MÓDULO L298N.

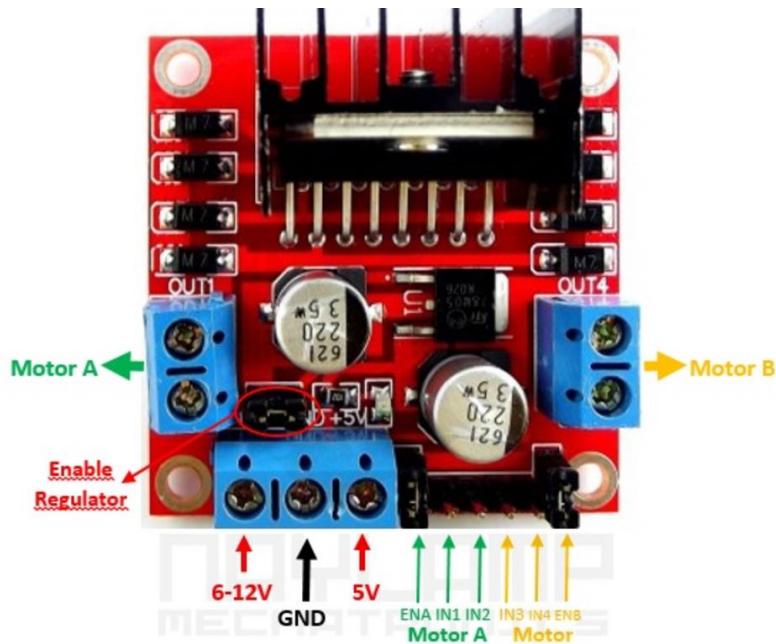


Figura 28. Módulo L298N.

El módulo L298N permite a la placa *Raspberry Pi* controlar el sentido y la velocidad de giro de dos motores de corriente continua, ya que posee dos canales de Puente H. Básicamente esta formado por un driver L298N, sus diodos de protección y un regulador de voltaje 5V (78M05)

Posee un conector de 6 pines para ingresar las señales TTL para controlar los motores, una bornera de tres pines para la alimentación, y dos borneras de 2 pines para la salida a los motores.

El módulo se ha alimentado utilizando una sola fuente conectada a la entrada de 6-12V y con el Jumper para habilitar el regulador, teniendo en cuenta que el voltaje de la fuente es el que soporta el motor. De esta forma no tenemos que conectar la entrada de 5V a ninguna fuente y automáticamente podemos utilizar este pin como una salida de 5V que irá conectado al Pin de 5V de la *Raspberry Pi*.

En cuanto al control del módulo, los pines ENA, IN1, IN2 corresponden a las entradas para controlar el motor A (OUT1, OUT2). De igual manera ENB, IN3, IN4 permiten controlar el motor B (OUT3, OUT4). En nuestro sistema ENA y ENB siempre estarán habilitados, IN1, IN2, IN3, IN4 irán conectados con los pines GPIO de nuestra placa *Raspberry Pi*. De este modo a partir del envío de 1 ó 0 conseguiremos modificar el sentido del giro del motor.

A continuación se puede ver de forma sencilla el conexionado de un solo motor:

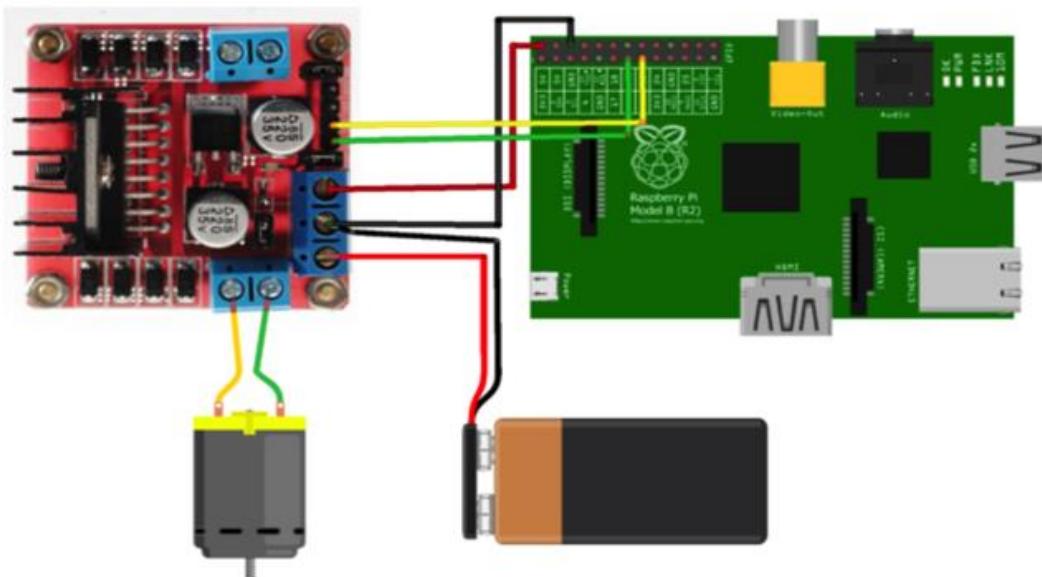


Figura 29. Esquema simple del conexionado de un motor.

	Controlador 1		Controlador 2	
	Pines	GPIO	Pines	GPIO
<b>IN1</b>	13	27	29	5
<b>IN2</b>	15	22	31	6
<b>IN3</b>	16	23	36	16
<b>IN4</b>	18	24	37	26

Tabla 5. Conexiones Raspberry Pi y Módulo L298N.

### 3.2.4. CHASIS Y MOTORES.

En un primer momento se eligió un chasis que tendría dos ruedas motrices, con sus respectivos motores de corriente continua, y una rueda directriz, formando así un tipo de locomoción llamada triciclo. El mayor problema de este chasis, por el que se descartó finalmente, fue el tamaño, ya que todos los componentes necesarios para que el robot realizara sus funciones, no cabían en el chasis de una forma en la que el peso estuviese repartido.

Además esta configuración presentaba un problema en el desplazamiento en línea recta porque la rueda delantera iba cambiando su dirección constantemente, al no ser fija, y era prácticamente imposible mantener una línea recta de forma prolongada y controlada.

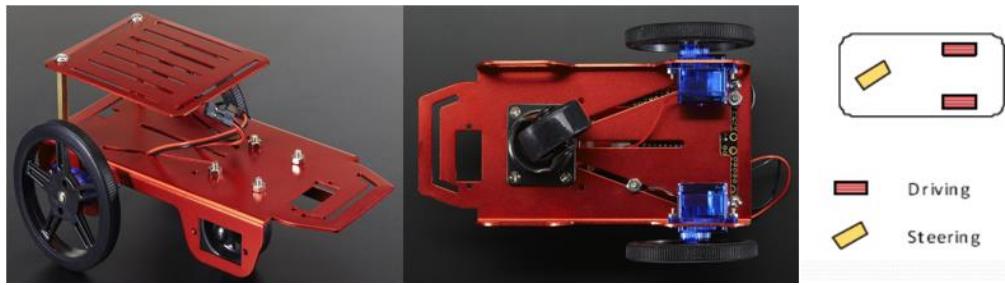


Figura 30. Primera elección de chasis.

Una vez comprobado que era imposible realizar esta configuración se optó por una configuración de un robot con cuatro ruedas motrices, cuatro motores de DC y de un chasis con un tamaño bastante mayor.

Con esta nueva configuración, no hay ruedas directrices, por lo que la conducción y los cambios de dirección se realizan a partir de la modificación de las velocidades lineales de las ruedas situadas en el plano izquierdo o derecho.

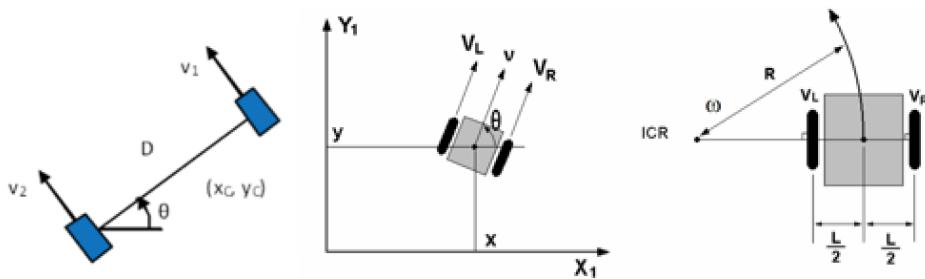


Figura 31. Sistema de locomoción diferencial.

Este sistema de locomoción es un sistema barato, bastante fácil de implementar y con un diseño muy simple, sin embargo requiere un control de precisión para trayectorias rectas. Además, al no tener ruedas directrices y debido al posible cambio de diámetro de las ruedas puede distorsionar el control de dirección del vehículo. Gracias a este cambio se consigue un mejor reparto de pesos y colocación en el chasis de los distintos componentes. Además el precio del chasis es bastante asequible comparándolo con otros del mercado.

El conjunto está formado por cuatro motores de corriente continua, que deben ser alimentados a 5V, dos estructuras de plástico (27x16 centímetros), cuatro ruedas y un sistema de alimentación para introducir baterías.



Figura 32. Opción elegida de chasis.

### 3.2.5. SISTEMA DE ALIMENTACIÓN.

En el sistema de alimentación se han utilizado dos opciones, ya que se debe alimentar tanto a la *Raspberry Pi* como al módulo L298N (driver) y los motores.

La opción elegida para dar potencia al microcontrolador es una pequeña batería recargable de litio con una capacidad de 2200 mAh, que proporciona una salida de 5V hasta 1A a través de un puerto USB.



Figura 33. Batería de litio recargable.

Principalmente se ha optado por este tipo de batería porque al tratarse de un robot móvil autónomo era necesario una batería portátil, además la salida USB proporciona una conexión directa con la conexión microUSB de la fuente de alimentación de la *Raspberry Pi*, es recargable, por lo que reduce el gasto en baterías no recargables y tiene un tamaño pequeño, lo que ha facilitado su integración en el robot.

En cuanto a la alimentación del módulo L298N y de los motores, se han elegido dos fuentes de alimentación de 6V, colocadas en paralelo, con un switch de apagado y encendido y con unos cables de conexión que irán conectados al driver, como ya se explicó anteriormente en el apartado del módulo L298N (figura 28). El switch es muy útil para poder realizar pruebas de software sin tener la necesidad de utilizar los motores, como ocurría al inicio. Por lo anteriormente expuesto, y porque trabajan a voltajes diferentes, se requieren dos fuentes de alimentación.

## 3.3. ARQUITECTURA SOFTWARE.

### 3.3.1. HERRAMIENTAS SOFTWARE.

En este apartado se describen los sistemas operativos utilizados, tanto en la *Raspberry Pi* como en el ordenador, para poder llevar a cabo este proyecto.

#### 3.3.1.1. SISTEMA OPERATIVO RASPBIAN.

El sistema operativo *Raspbian* es una distribución del sistema operativo *GNU/Linux* y por lo tanto está basado en *Debian Wheezy*, optimizado para la placa *Raspberry Pi*. Este sistema operativo tiene los programas y utilidades básicas para poder arrancar la *Raspberry Pi*. Además, *Raspbian* proporciona más de 35.000 paquetes y software precompilados de fácil instalación para una mejora del rendimiento de la placa.



Figura 34. Logo Raspbian.

Tras instalar el sistema operativo se debe configurar el sistema, activando la opción de la cámara y descargando sus paquetes correspondientes. Una vez activada la cámara se deben instalar los paquetes necesarios de *Python* y *OpenCV* para poder comenzar a programar el software que posteriormente se ejecutará. Por último, se deben instalar las librerías *GPIO* para poder comunicar y controlar los motores.

Para realizar todos estos procesos fue necesario la activación de los puertos *SSH*, *VNC* y la configuración de una red wifi con una dirección IP fija, para establecer la conexión a la placa desde el ordenador sin necesidad de conexión a un monitor.

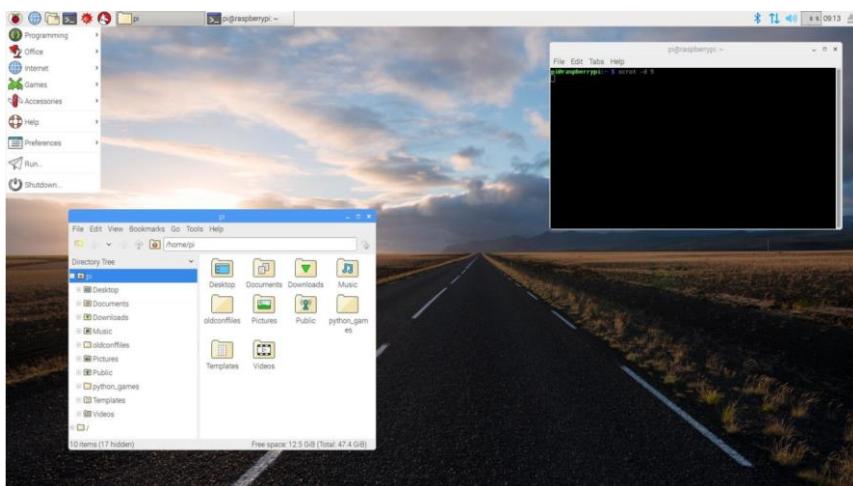


Figura 35. Escritorio de la Raspberry Pi.

### 3.3.1.2. ANACONDA Y PYTHON.

Anaconda es un programa de fuente libre que permite programar en el lenguaje de programación *Python*. Está enfocado al campo de la ciencia, ya que una vez instalado se puede acceder a multitud de paquetes científicos o librerías tecnológicas colgados en la nube. Para descargarse sus diferentes librerías hay que utilizar su gestor de paquetes *Conda*, y se deberá realizar su llamada desde la terminal, a través de comandos. Gracias a *Conda* se podrán descargar todas las librerías necesarias para ejecutar la librería *OpenCV*, vital en este proyecto.



Figura 36. Logo de Anaconda.

El lenguaje de programación *Python*, es el que se ha utilizado para realizar el software de este proyecto. Es un lenguaje poderoso, fácil de aprender, capaz de soportar multitud de formas de programar y compatible con la librería *OpenCV*.



Figura 37. Logo de Python.

### 3.3.1.3. OPENCV.

Es una biblioteca libre de visión artificial, fácil de utilizar y altamente eficiente. Tiene licencia BSD, que permite ser usada libremente para propósitos comerciales y de investigación, con las condiciones en ella expresada. [38]



Figura 38. Logo de OpenCV.

Contiene más de 500 funciones, algunas de las cuales se han utilizado en el proyecto. Esta librería nos permitirá leer una imagen, pasarl a distintos escalas de colores (RGB, HSV, blanco y negro), eliminar contornos no deseados o encontrar líneas rectas, y analizar sus ángulos y sus puntos de fuga. Todos estos procesos serán explicados posteriormente.

### 3.3.2. DESARROLLO DE LA APLICACIÓN.

El objetivo final, del software, es conseguir distinguir las líneas del carril bici para poder guiar a nuestro robot móvil. Para conseguirlo, se ha tratado cada imagen o *frame* con distintos filtros, obteniendo una serie de resultados que se han interpretado en nuestro código.

La lógica del programa se basa en detectar los diferentes tipos de línea existentes en la imagen obtenida. Los tipos de línea vienen determinados por una serie de parámetros, principalmente el ángulo de inclinación y el lugar de aparición de las líneas, que se han definido dentro del propio programa. Cada vez que el programa diferencia un tipo de línea, lo registra en un contador y, tras haber realizado todo el proceso, el tipo que más líneas obtenga será la elección elegida para dirigir al robot. Por lo tanto, habrá tres opciones, dependiendo de si el sistema detecta recta, si detecta curva hacia la derecha y, por último, si detecta curva hacia la izquierda.

Puede ocurrir que el software no encuentre ninguna línea, por lo que se irá llenando un contador a medida que esto ocurra. En el caso de que ocurriese 100 veces de forma consecutiva, el programa daría un mensaje de error y el robot se detendría, entendiendo que no existe carril bici.

A continuación se muestra el esquema de este funcionamiento lógico:

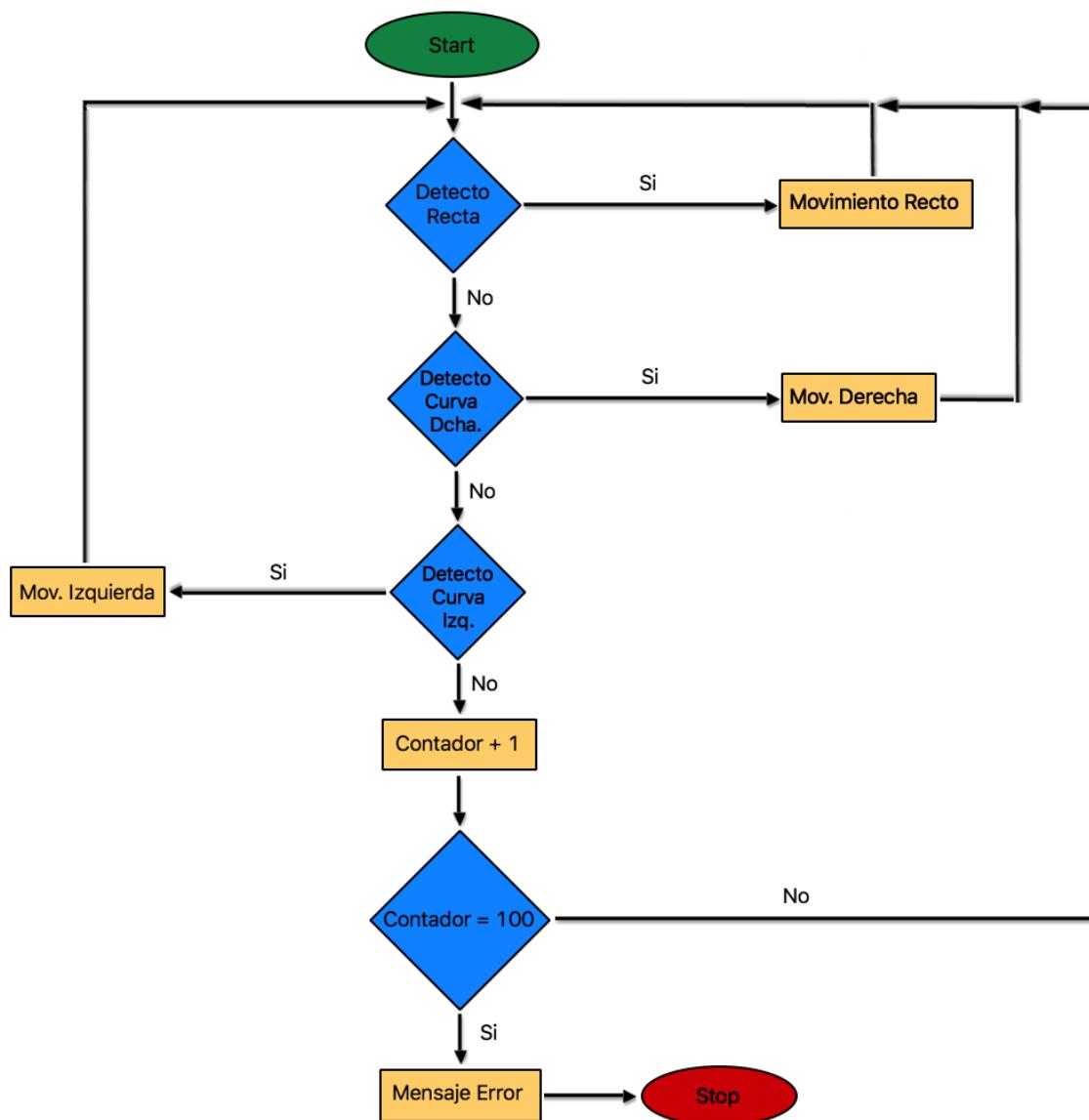


Figura 39. Funcionamiento lógico de la aplicación.

A continuación se muestra una imagen del carril bici donde se diferencian claramente las líneas de color blanco. Esta imagen y similares han sido utilizadas para la explicación del software.



Figura 40. Carril bici.

La estructura que se ha seguido para analizar la imagen es la siguiente:



Figura 41. Estructura código.

### 3.3.2.1. CONVERSIÓN DE RGB A HSV.

Función: `cvtColor(frame, code)`

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir
- **code:** Código del espacio de color al que se quiere convertir, en este caso es `cv2.COLOR_BGR2HSV`

Para comenzar se realiza una conversión de la imagen del espacio de color BGR (*Blue, Green, Red*) a HSV (*Hue, Saturation y Value*; traducido al español serían: Matíz, Saturación y Valor). En el espacio HSV, *Hue* indica el color, *Saturation* indica la oscuridad, la escala de gris, el valor más alto de saturación dará el color que se haya indicado en el matiz y un valor pequeño de saturación siempre será gris. Por último, el valor indica el brillo, siendo el extremo inferior negro y el extremo superior blanco.

En la siguiente imagen se puede apreciar como varían los colores dependiendo de las distintas variables:

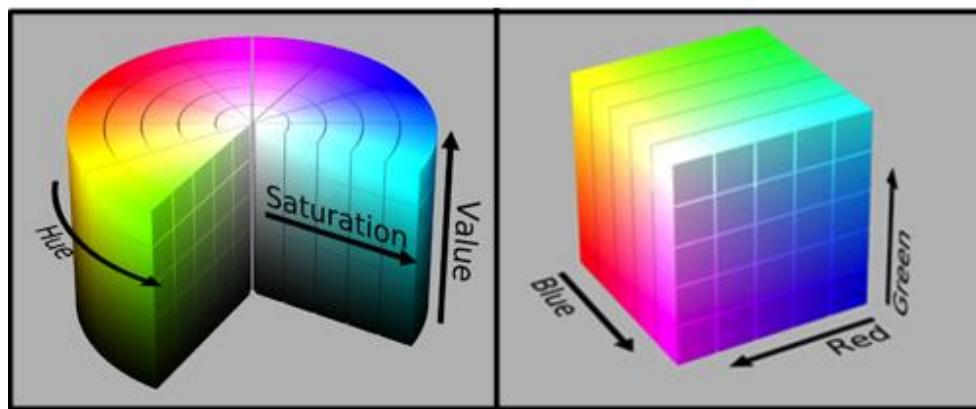


Figura 42. Representación cilíndrica HSV frente representación cúbica BGR.

Se realiza esta conversión porque es más fácil buscar un color en el espacio HSV que en el BGR, ya que en BGR no se tiene en cuenta la luz, el brillo y las sombras, y mientras una sombra en RGB cambia el color, en HSV solo habría que ajustar el brillo.



Figura 43. Imagen del carril bici en el espacio de color HSV.

### 3.3.2.2. SELECCIÓN DE REGIÓN DE INTERÉS.

La selección de región de interés se crea para eliminar la información innecesaria que proporcionan las imágenes obtenidas. La primera sección eliminada será la parte superior de la imagen, con más exactitud: desde el horizonte, donde finaliza el carril bici, hasta el límite superior de la imagen. Además, se puede eliminar cierta información de los laterales, ya que las zonas más extremas de la cámara no aportan ninguna información, solo falsos positivos.

De esta forma se elimina dicha información innecesaria, que sólo proporciona ruido y un gran número de falsos positivos, lo que se traduce en una gran disminución de los errores del sistema.

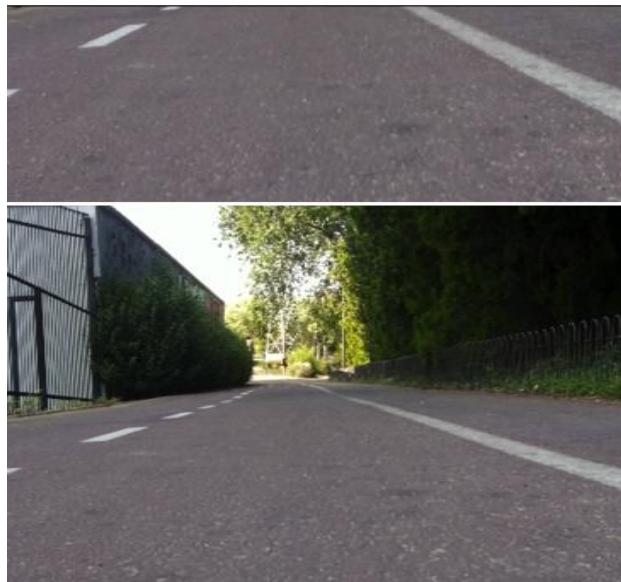


Figura 44. Arriba región de interés; abajo imagen sin region de interés.

### 3.3.2.3. FILTRAR RUIDO.

Funciones: `erode (frame,element,iterations = 1)`, `dilate (frame,element,iterations = 1)`

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir.
- **element:** estructura del elemento que se va a usar en el filtro, se usa un elemento rectangular de 5x5, conseguido con:  
`getStructuringElement(cv2.MORPH_RECT,(5,5))`.
- **iterations:** número de veces que se aplica el filtro, en el proyecto es 1.

Para filtrar el ruido en las imágenes se han utilizado dos funciones que aplican las operaciones morfológicas de erosión y dilatación, consiguiendo eliminar el ruido, aislar los elementos individuales y las uniones de elementos dispares en una imagen y hallando protuberancias o agujeros de intensidad en una imagen. Gracias a la eliminación de cierto ruido y a la selección de la región de interés, se facilita que los procesos posteriores estén menos expuestos a posibles errores.

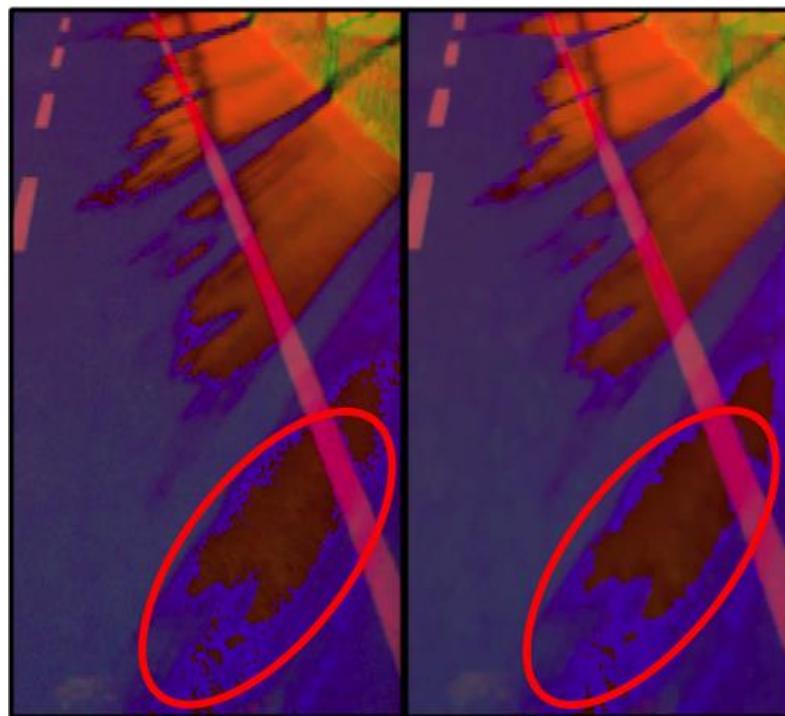


Figura 45. Imagen anterior y posterior al filtrado.

### 3.3.2.4. SUAVIZAR CONTORNOS.

Función: GaussianBlur(frame, (width, height), sigmaY)

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir.
- **width:** ancho del tamaño del nucleo de Gauss, en el programa es 5.
- **height:** altura del tamaño del nucleo de Gauss, en el programa es 5.
- **sigmaY:** desviación estándar en Y, si este campo es “0” significa que la desviación estándar de Y es la misma que la de X, éste es el caso que se utiliza en el proyecto.

En este punto del código se realiza un filtro de Gauss para suavizar los contornos y hacer la imagen más clara, es decir, eliminar los detalles. Puede verse como un desenfoque en una cámara fotográfica. Es uno de los filtros más útiles, aunque no el más rápido, además es el método de difuminación más comúnmente utilizado.

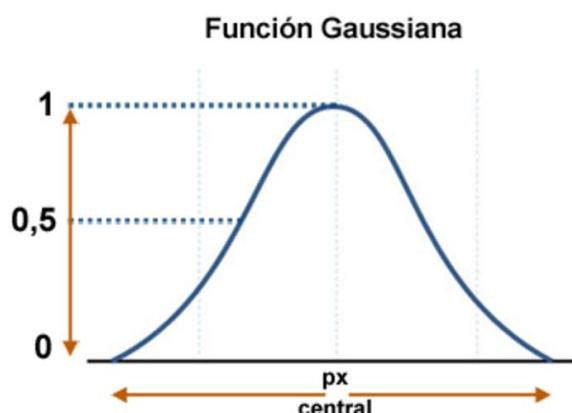


Figura 46. Campana de Gauss.

En el filtro Gaussiano se realiza una ponderación siguiendo, lógicamente, la campana de Gauss. Con esto se consigue dar más importancia a los píxeles que están más cerca del centro de los que están más alejados. Se utiliza la campana de Gauss porque es una aproximación de cómo ve el ojo humano, intentando ser lo más natural posible. Para poder aplicar el filtro debemos hacerlo en dos dimensiones, por lo que se hará a través de una máscara o kernel de convolución.

La convolución consiste en ir recorriendo píxel a píxel una imagen con una máscara o kernel de NxN. Este tamaño determina el número de píxeles con el que vamos a trabajar. Es muy importante que el tamaño sea impar para siempre tener un píxel central que será el píxel en cuestión que estamos tratando. [39]

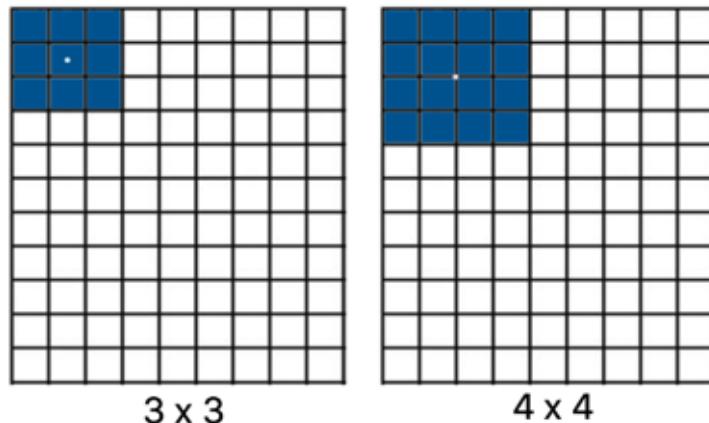


Figura 47. Máscara o Kernel.

### 3.3.2.5. FILTRAR COLOR BLANCO.

Función: `cv2.inRange(frame, lim_inferior, lim_superior)`

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir.
- **lim\_inferior:** array del límite inferior del rango.
- **lim\_superior:** array del límite superior del rango.

Una vez suavizada la imagen, se procede a seleccionar únicamente aquello que sea de color blanco. Para realizar esta acción se debe crear dos arrays donde se guarda el rango de color blanco en el espacio HSV que se desea seleccionar, siendo un array el rango límite inferior y otro el límite superior. Una vez creados estos arrays o matrices, se utiliza la función, que es la encargada de comprobar si los elementos de nuestra imagen se encuentran dentro del rango indicado, de la siguiente forma:

$$\text{dst}(I) = \text{lowerb}(I)_0 \leq \text{src}(I)_0 \leq \text{upperb}(I)_0$$

Siendo “src” la matriz de entrada de la imagen que queremos procesar, “lowerb” es el límite inferior, al contrario de “upperb”, que es el límite superior y, por último, “dst” es la matriz de salida, es decir, nuestra imagen filtrada.

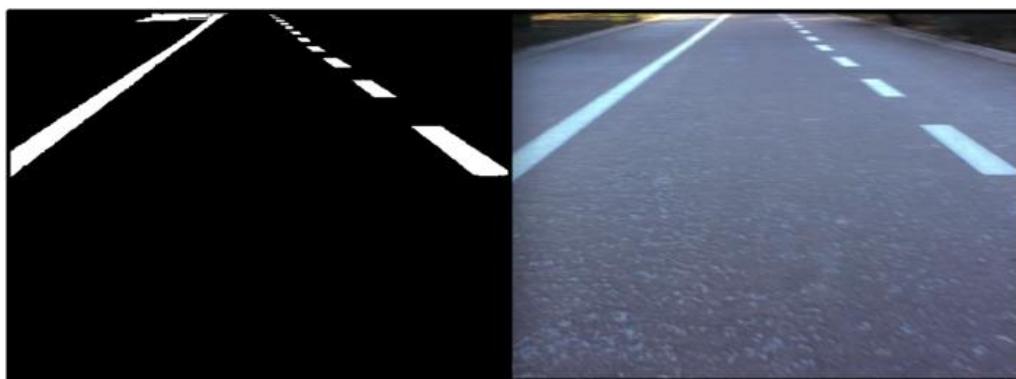


Figura 48. Imagen después y antes del proceso.

### 3.3.2.6. FILTRO CANNY.

Funcion: **Canny(frame, threshold1, threshold2, apertureSize = n)**

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir.
- **threshold1:** primera medida para el procedimiento de histéresis. En el código es igual a 100.
- **threshold2:** segunda medida para el procedimiento de histéresis. En el código es igual a 300.
- **apertureSize:** tamaño de apertura para el operador sobel, n es igual a 3.

La detección de bordes, en procesamiento digital de imágenes, nos permite averiguar los contornos de la imagen o, mejor dicho, los cambios bruscos de intensidad lumínica. Para la detección de bordes, en esta aplicación, se utiliza el filtro *Canny*. Este algoritmo está considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución y basado en la primera derivada.

*Canny* propuso un método para la detección de bordes, el cual se basaba en satisfacer tres criterios principales: un criterio de **detección** que expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos positivos, un criterio de **localización** que establece que la distancia entre la posición real y la localizada del borde se debe minimizar, y un criterio de **una respuesta** que integre las respuestas múltiples correspondientes a un único borde.

Como se ha mencionado anteriormente, el algoritmo *Canny* se trata de un método basado en la primera derivada, es usada por que toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por lo tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada, característica que es usada para detectar un borde. [40]

Tras aplicar el filtro *Canny* a dos imágenes, las imágenes obtenidas se muestran a continuación:

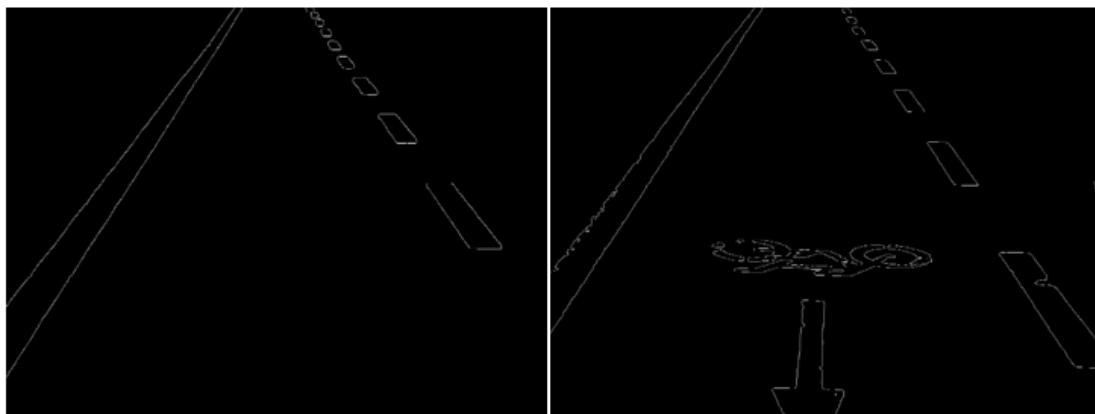


Figura 49. Imágenes con filtro Canny.

### 3.3.2.7. TRANSFORMADA DE HOUGH.

Funcion: `HoughLines(frame, rho, tetha, threshold)`

- **frame:** Imagen de entrada, donde se le pasa la imagen que se quiera convertir.
- **rho:** resolución de la distancia del acumulador de píxeles.
- **tetha:** resolución del ángulo del acumulador de píxeles.
- **threshold:** parámetro umbral del acumulador.

La transformada de Hough es un algoritmo empleado en reconocimiento de patrones de una imagen, ya que es una técnica muy robusta frente al ruido. Tiene un funcionamiento estadístico y en función de los puntos que se tengan se debe averiguar las posibles líneas en las que se encuentren los distintos puntos. Este proceso se logra aplicando una operación a cada línea en un rango determinado.

Dentro de su funcionamiento la trasformada de Hough utiliza una representación paramétrica de forma geométrica, es decir, una recta representada con los parámetros “r” y  $\theta$  (*theta*), donde “r” es la distancia entre a línea y el origen, y  $\theta$  es el ángulo del vector desde el origen al punto más cercano. Por medio de esta parametrización, la ecuación de la recta se podría escribir de la siguiente forma:  $r = x \cos\theta + y \sin\theta$ . [41]

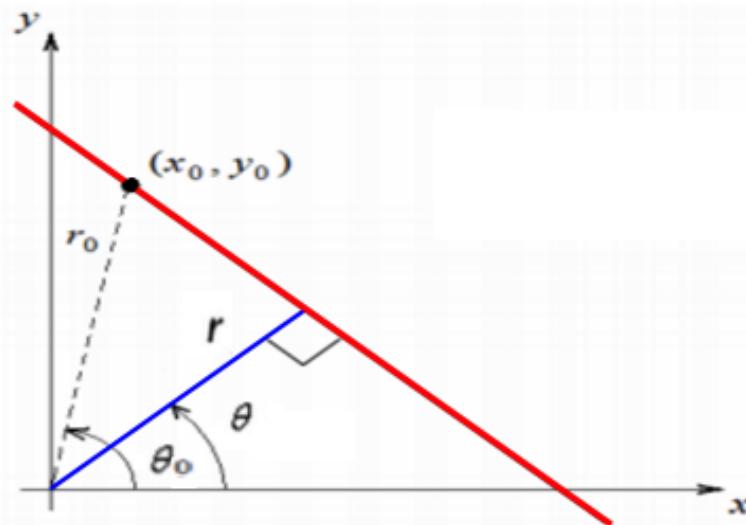
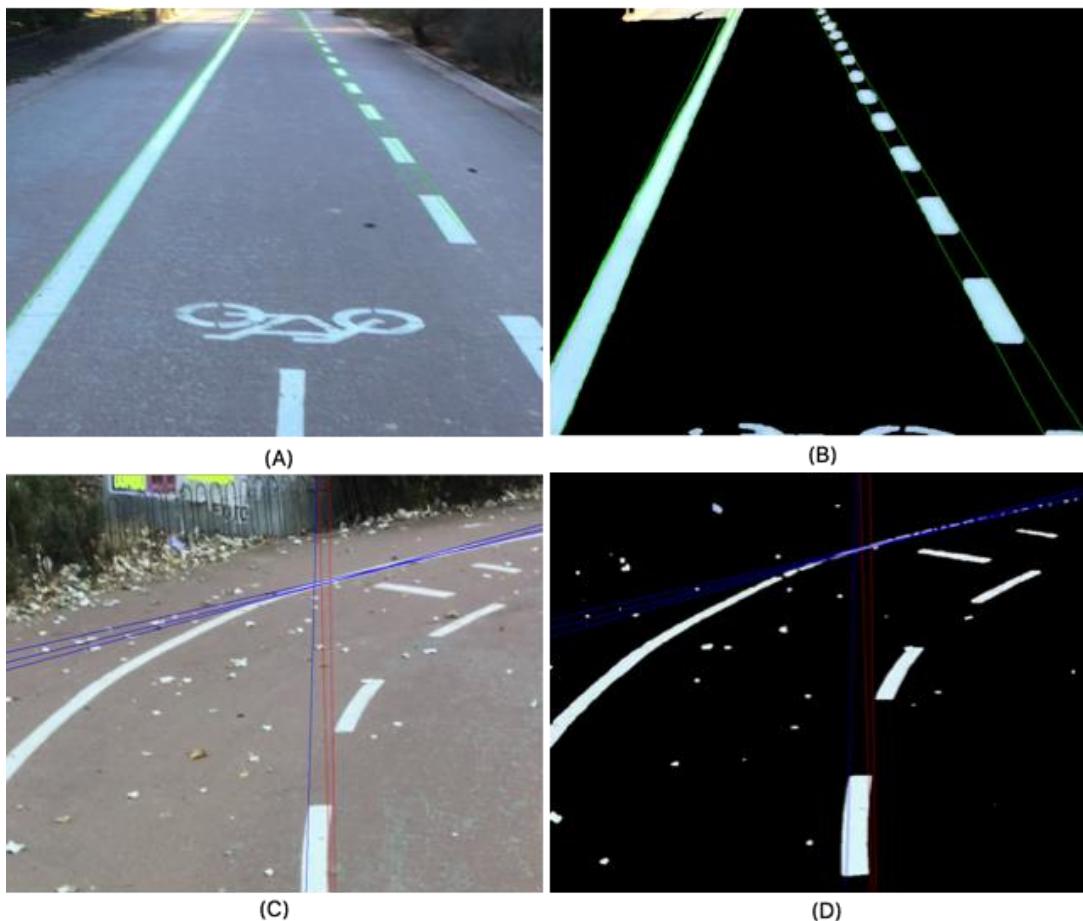


Figura 50. Representación transformada de Hough para rectas.

Una vez conocida la teoría de la transformada de Hough, se puede aplicar a nuestra imagen. Puesto que se pueden generar líneas con distintos ángulos, se debe realizar un proceso donde dependiendo del ángulo y la sección de la imagen donde aparezca esa linea el robot realice un giro, continúe de frente o se desestime por encontrarse en un lugar donde no deben existir líneas.

Por lo tanto, para que el robot continúe de frente solo interesan las líneas con un ángulo determinado, descartando el resto de líneas con distinto ángulo, con la ayuda del parámetro  $\theta$ . También se descartan las líneas con el parámetro  $\theta$  deseado pero que no se encuentren en la sección deseada. Para realizar los giros se realiza un proceso similar, pero en este caso, dependiendo del lugar donde se encuentre la línea y su ángulo, se buscan líneas cercanas a la horizontal. Dependiendo del parámetro  $\theta$  se diferencia si la línea está inclinada hacia la derecha, teniendo un ángulo cercano a la horizontal, donde se realiza un giro hacia la derecha. En cambio, en la situación contraria, donde la inclinación sea hacia la izquierda, lógicamente, se gira hacia la izquierda.

Por último, el resultado obtenido tras aplicar la transformada de Hough es el siguiente:



**Figura 51. Ejemplos tras aplicar la transformada de Hough.**

En el anterior conjunto de imágenes, (A) y (C) son imágenes reales combinadas con todo el proceso; (B) y (D) son las imágenes que se obtienen tras realizar la transofrmada de Hough. Además, (A) y (B) detectan la línea recta y (C) y (D) el giro hacia la derecha.

### 3.3.2.8. CONTROL DE DIRECCIÓN.

**Funcion: GPIO.setup(pin, type)**

- **pin:** pin de la *Raspberry Pi* que se desea utilizar.
- **type:** tipo de pin, si es de salida o de entrada, en el proyecto es de tipo **GPIO.OUT**.

**Funcion: GPIO.output(pin,type)**

- **pin:** pin de la *Raspberry Pi* que se desea utilizar.
- **type:** tipo de señal, si es de activación (**GPIO.HIGH**), si es de apagado (**GPIO.LOW**).

**Funcion: time.sleep(seconds)**

- **seconds:** segundos en los que el programa va a estar parado, mientras los motores continuan activos haciendo girar las ruedas

Una vez realizados todos los procesos anteriores, se obtiene toda la información necesaria para poder decidir sobre la dirección a tomar por el robot, como bien se ha explicado anteriormente. Para ello se utiliza la librería **GPIO** de *Raspberry Pi* y la librería “*time*” de *Python*. Una vez importadas estas dos librerías se debe indicar qué salidas de **GPIO** están conectadas a nuestro controlador de motores y manejan por lo tanto los motores. Tras haberlo indicado, ya sólo se deben gestionar dos señales: “HIGH” y “LOW”. Dependiendo de la señal **GPIO** que se active las ruedas girarán en una dirección u otra.

Para encender los motores se utiliza “HIGH” y se indica, por medio de la librería “*time*”, cuento tiempo va a estar activo. De esta manera, cada vez que el código pase por esta señal el motor funcionará. En cambio la señal “LOW” realiza lo contrario y se utiliza cuando no se pretende que el motor funcione, es decir, se decide que la rueda no gire.

La librería “*time*” se utiliza porque es necesario indicar el tiempo que va a estar activo el motor, lo que se logra realizando una parada (*sleep*) entre una señal de encendido y apagado. Por lo tanto, una vez se tienen todos estos conceptos, sólo es necesario aplicarlos dependiendo de la dirección y fuerza con la que se desea realizar el giro. Para realizar un giro leve se necesita menos tiempo (en torno 0,3 segundos), mientras que para aumentar la dureza del giro se puede emplear hasta un segundo. Por lo tanto se debe tener en cuenta que la dureza o suavidad del giro dependerá del tiempo empleado en el mismo.

De esta manera se hace necesario el uso de los parámetros anteriormente mencionados en la transformada de Hough, ya que dependiendo del ángulo de las líneas creadas con la transformada se indicará que tipo de giro se debe realizar.

# CAPÍTULO 4. EVALUACIÓN DEL SISTEMA, PRUEBAS Y LIMITACIONES.

## 4.1. DETERMINACIÓN DE LOS VALORES DE FILTRADO.

A continuación se detallan las pruebas que se han llevado a cabo para determinar los valores de filtrado y los parámetros de las diferentes funciones utilizadas que arrojan mejores resultados.

### 4.1.1. FILTRADO DE LA REGIÓN DE INTERÉS.

Para este filtrado se establecen unas dimensiones máximas y mínimas que la región de interés puede adoptar. Esta región de interés tendrá una forma rectangular, ya que las líneas terminan en el horizonte y todo lo que aparezca por encima de él no aporta ninguna información en esta aplicación. Para los límites inferiores se buscó la distancia más alejada al robot sin perder información útil para el control de la dirección del mismo.

Para ello se realizaron varias pruebas, utilizando videos grabados con la cámara de la *Raspberry Pi* (con una resolución de 1920x1080 píxeles), de las que se determinó que la altura mínima del área de interés sería de 950 píxeles, mientras que el ancho debería ser de 1900 píxeles. El proceso para hallar las dimensiones superiores de la región de interés es similar. Se realizó una medida de la altura donde se encontraba el horizonte y se estimó el ancho máximo que podría tener el carril bici a esa altura. Al igual que en el caso anterior, tras varias pruebas, se llegó a la conclusión de que la altura máxima sería de 600 píxeles y el ancho máximo de 1900 píxeles.

Sin embargo, se tuvo que tener en cuenta que la resolución de la cámara con la que se iba a realizar las verdaderas pruebas de funcionamiento era de 640x480 píxeles, por lo que se realizaron los cálculos necesarios para obtener los nuevos valores de filtrado. Tras los cálculos, los nuevos valores de anchura eran de 633 píxeles, el límite superior de altura se encontraba en 266 píxeles y el límite inferior se encontraba en 422 píxeles.

Resolución		
	1920x1080 píxeles	640x480 píxeles
Ancho	1900 píxeles	633 píxeles
Altura superior	600 píxeles	266 píxeles
Altura inferior	950 píxeles	422 píxeles

Tabla 6. Valores basados en la resolución.

Aunque se pueda pensar que se trata de un filtrado relativamente simple, la correcta asignación de los distintos valores hizo disminuir el número de falsos positivos drásticamente. Este proceso

elimina grandes áreas de la imagen correspondientes a aceras, vehículos, césped, árboles o el cielo, que podían contener patrones detectables por la función de detección de líneas.

#### 4.1.2. FILTRADO POR COLOR BLANCO.

Este filtro establece el rango para considerar que una forma es de color blanco. Para hallar este rango se ha tenido que realizar pruebas en diferentes escenarios. En estos escenarios se buscaba que hubiese variaciones de luz para poder hallar el blanco tanto en sombras como en lugares con grandes reflejos de luz.

En primer lugar las pruebas se realizaron en el interior de una casa siguiendo una única línea blanca situada en un suelo oscuro, en este escenario se fue variando entre un escenario con luz ambiental, un escenario con las luces de la propia casa y en el último se utilizó una fuente de luz incidiendo sobre la línea donde se resaltaba aún más el color blanco. De aquí se obtuvo un pequeño rango de filtrado del color blanco, debido a las pocas variantes.

Posteriormente se pasó a las pruebas en el exterior y se realizaron diferentes grabaciones con la cámara del robot en el carril bici, en varios recorridos, y en diferentes horarios, donde se pueden encontrar sombras, reflejos y destellos de luz. Con estas grabaciones se pudo calibrar de mejor manera el rango, ampliéndolo de manera sustancial. De las pruebas se obtuvieron los siguientes resultados:

		Luminosidad				
		Muy Baja	Baja	Media	Alta	Muy Alta
Límite inferior	[5,5,130]	[5,5,160]	[5,5,185]	[5,5,190]	[5,5,215]	
Límite superior	[179,85,200]	[179,85,225]	[179,85,235]	[179,85,255]	[179,85,255]	

Tabla 7. Rangos del filtro de color blanco basado en la luminosidad.

Gracias a los resultados se puede comprobar cómo varía el valor de brillo dependiendo de la luz que existe en el entorno. En base a los resultados y realizando un pequeño estudio del horario y zonas donde se han realizado las pruebas se ha considerado que el rango que mejor se ajusta al programa y capta la mayoría de casos para trabajar en horario diurno es el siguiente:

- Límite inferior: [5,5,170]
- Límite superior [179,85,255]

Por último, se debe destacar que, con el rango elegido, el programa tiene un gran funcionamiento con luminosidad media; un funcionamiento normal, con pequeños errores, con la luminosidad baja y alta; y comienza a tener un mayor número de errores con luminosidad muy baja y muy alta.

#### 4.1.3. FILTRADO DE LAS LÍNEAS.

Este filtro, sin duda alguna, es el más importante del sistema, debido a que en él se basa el funcionamiento del robot. Para conseguir un buen funcionamiento del sistema se debe eliminar falsos positivos o líneas que aparecen en lugares donde no son relevantes.

Por ello, se va a explicar el procedimiento de elección de los parámetros de filtrado de las líneas que nos indican que se debe seguir de frente. Para este apartado se han diferenciado cuatro casos, los cuales son los siguientes:

- **Detección única de la línea izquierda:** El robot sólo es capaz de ver la línea izquierda porque se encuentra pegado a ella, y sólo es capaz de ver la línea más próxima a él.
- **Detección de línea izquierda acompañada de línea derecha:** El robot está situado en mitad del carril y es capaz de ver las dos líneas.
- **Detección de línea derecha acompañada de línea izquierda:** El robot está situado en mitad del carril y es capaz de ver las dos líneas.
- **Detección única de la línea derecha:** El robot sólo es capaz de ver la línea derecha porque se encuentra pegado a ella, y sólo es capaz de ver la línea más próxima a él.



Figura 52. Distintos casos de detección.

En el primer y segundo caso se ha elegido, en base a las pruebas realizadas, que el ángulo de detección debe ser mayor de  $50^\circ$  y menor de  $75^\circ$ , además el parámetro rho debe ser mayor que 500 píxeles. En cambio, para los dos casos restantes, el ángulo debe ser mayor de  $115^\circ$  y menor de  $140^\circ$  y el parámetro rho debe modificarse siendo menor que 200 píxeles.

En cuanto a la detección de las líneas en el giro hacia la izquierda se ha decidido utilizar la línea exterior como referencia para realizar el giro, ya que realizando las pruebas se ha observado que en la mayoría de los casos pierde de vista la línea interior. Por lo tanto, para realizar el giro se utiliza el rango comprendido entre  $105^\circ$  y  $110^\circ$ , pero en este caso, indicar el rango no es suficiente y se necesita indicar la altura del punto inicial de la línea y del punto final de la misma. Se debe realizar este procedimiento porque las líneas de giro hacia izquierda siempre se situarán en la parte superior de la imagen captada por el robot, véase figura 52. De este modo el punto inicial debe estar situado por encima del píxel 600 y el punto final por encima del píxel 680. En este caso no se ha utilizado rho para evitar un mayor número de falsos positivos en otros movimientos.



Figura 53. Visión del robot del giro hacia la izquierda.

Por último, en el giro hacia la derecha se ha seguido el modo de trabajo del giro hacia la izquierda, pero con una leve modificación, ya que, en este caso, sí se puede utilizar la rho. En este caso, los ángulos deberán ser mayores de  $78^\circ$  y menores de  $83^\circ$ , siendo la rho menor que 750 píxeles.

## 4.2. DETERMINACIÓN DE LOS PARAMETROS DE LAS FUNCIONES.

A continuación se analizan los resultados obtenidos al asignar diferentes valores a los parámetros de las funciones y se determina cuáles son los más adecuados.

### 4.2.1. FUNCIONES DE FILTRADO DE RUIDO.

Las primeras funciones que se encuentran en el código son las funciones *erode* y *dilate*, son funciones que hacen un primer barrido del ruido dilatando y disminuyendo la imagen.

Para ambas funciones se elige una plantilla rectangular para el filtro *kernel*, mucho más indicado que las otras dos opciones (cruces o elipse), y se utiliza una matriz, en ambos casos, de 5x5. Se toma esta decisión porque tras realizar varias pruebas no se nota una gran variación entre una matriz de 5x5, 3x3 o incluso una de 7x7. Y se decide la matriz de 5x5 porque es la más útil en las siguientes funciones utilizadas dentro del programa. En cuanto a estas funciones, se puede decir que la variación leve de los parámetros no implica ningún cambio relevante.

A continuación se pueden apreciar los leves cambios entre los filtros con tamaños de 3x3 y los de 5x5. La única diferencia es la ausencia de pequeños puntos en las zonas señaladas.

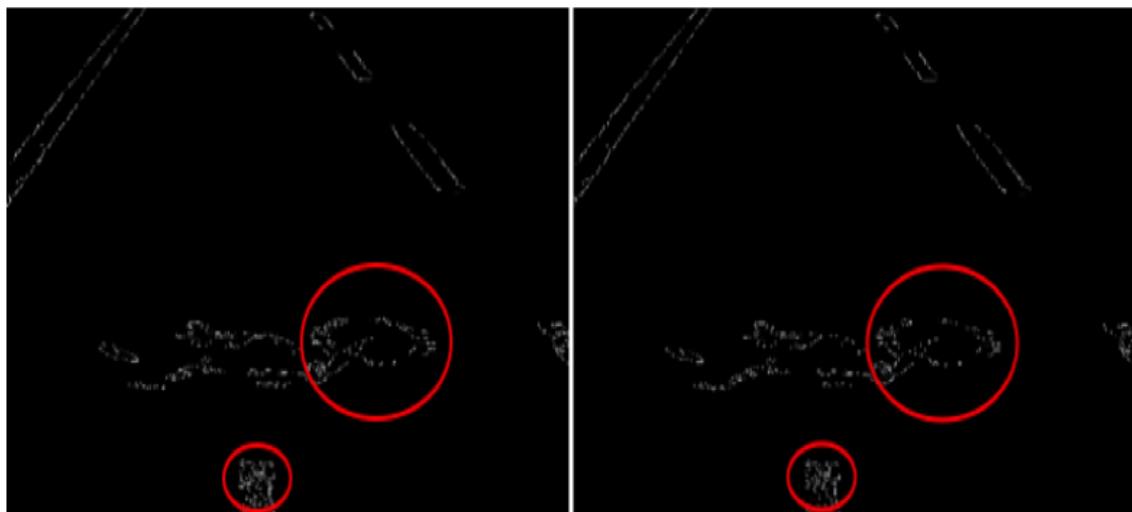


Figura 54. Filtro 3x3 (izq.) y 5x5 (dcha.).

A diferencia de las imágenes anteriores, en esta comparación de 3x3 y 7x7 se puede observar una mayor ausencia de puntos.

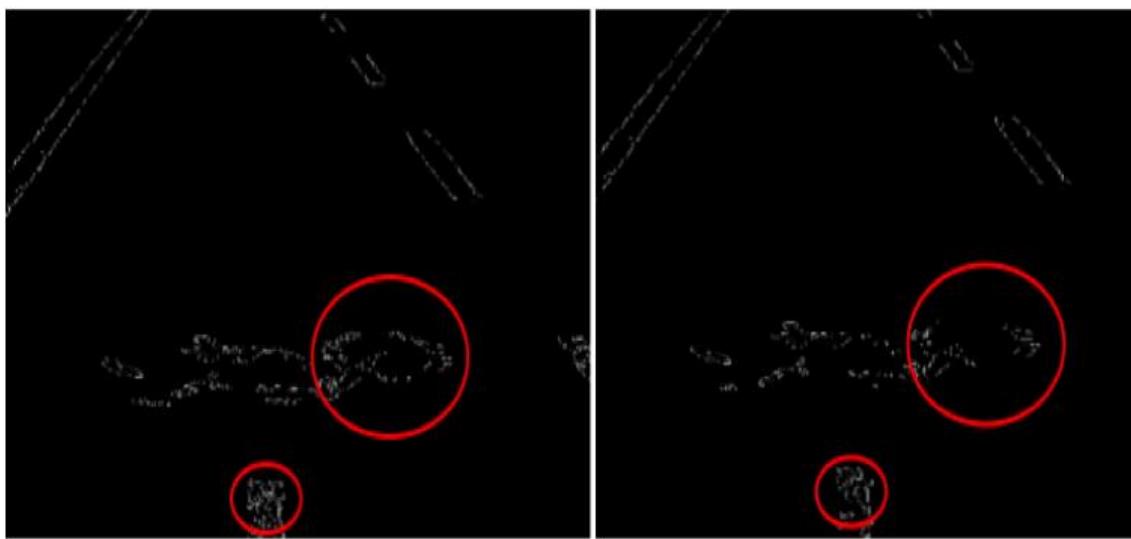


Figura 55. Filtro 3x3 (izq.) y 7x7 (dcha.).

Como ya se ha explicado anteriormente, debido a las leves diferencias entre 3x3 y 5x5, se ha elegido la segunda opción, ya que se adapta mejor a las siguientes funciones.

Tras realizar el primer barrido de ruido se encuentra la función *GaussianBlur*, en la cual se ha elegido una matriz igual que en las funciones anteriores (5x5), en este punto del código se probó también la función *MedianBlur*, encontrando unos resultados similares, sólo diferenciados por una ligera aparición de ruido, véase en la figura 55. Se ha optado por la primera función porque, pese a los resultados similares, y que en este proyecto esas leves diferencias de ruido no son significativas, en un proyecto a mayor escala podrían marcar la diferencia.



Figura 56. Diferencias entre GaussianBlur (izq.) y MedianBlur (dcha.).

#### 4.2.2. FUNCIONES DE CONTORNO.

En este apartado se encuentra la función *Canny* que tiene una de las labores más importantes, la de averiguar los contornos de las líneas que se desean analizar. Para encontrar los parámetros correctos en esta función, en primer lugar se ha acudido a la web oficial de *OpenCV*, donde aconseja que el ratio entre el valor máximo y el mínimo debe encontrarse entre 2:1 y 3:1. [42]

Por ello se ha realizado un programa paralelo donde se han probado distintas configuraciones. Este programa se basa en fotos del carril bici a las cuales se les añade una barra de valor máximo y valor mínimo. Estas barras son modificadas hasta encontrar las configuraciones que mejor resultado aportan. A continuación, se pueden ver algunas imágenes del funcionamiento del programa paralelo.

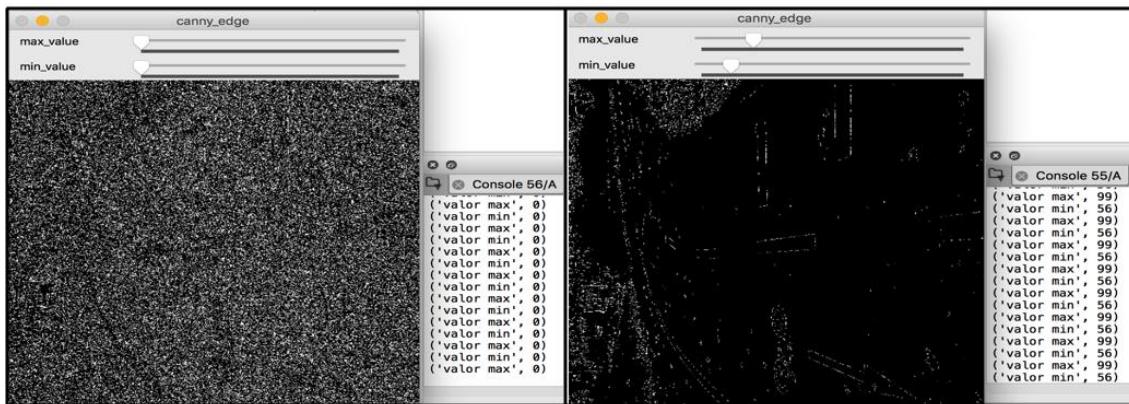


Figura 57. Programa filtro Canny. Al inicio (izq.) y modificando la foto (dcha.).

Tras las pruebas realizadas en el programa paralelo, se obtienen tres configuraciones de valores que proporcionan buenos resultados, siendo las siguientes: 50:150, 100:200 y 100:300. El siguiente paso ha sido realizar pruebas en el video del carril bici. En el código se comparan simultáneamente las tres configuraciones, viendo cuál mostraba menos ruido y contornos más diferenciados. Al finalizar este proceso, se han obtenido resultados muy similares, pero debido a leves diferencias se ha decidido que la opción más acertada es la configuración 100:300 que muestra levemente menor ruido y en algunas ocasiones líneas más definidas y continuas.

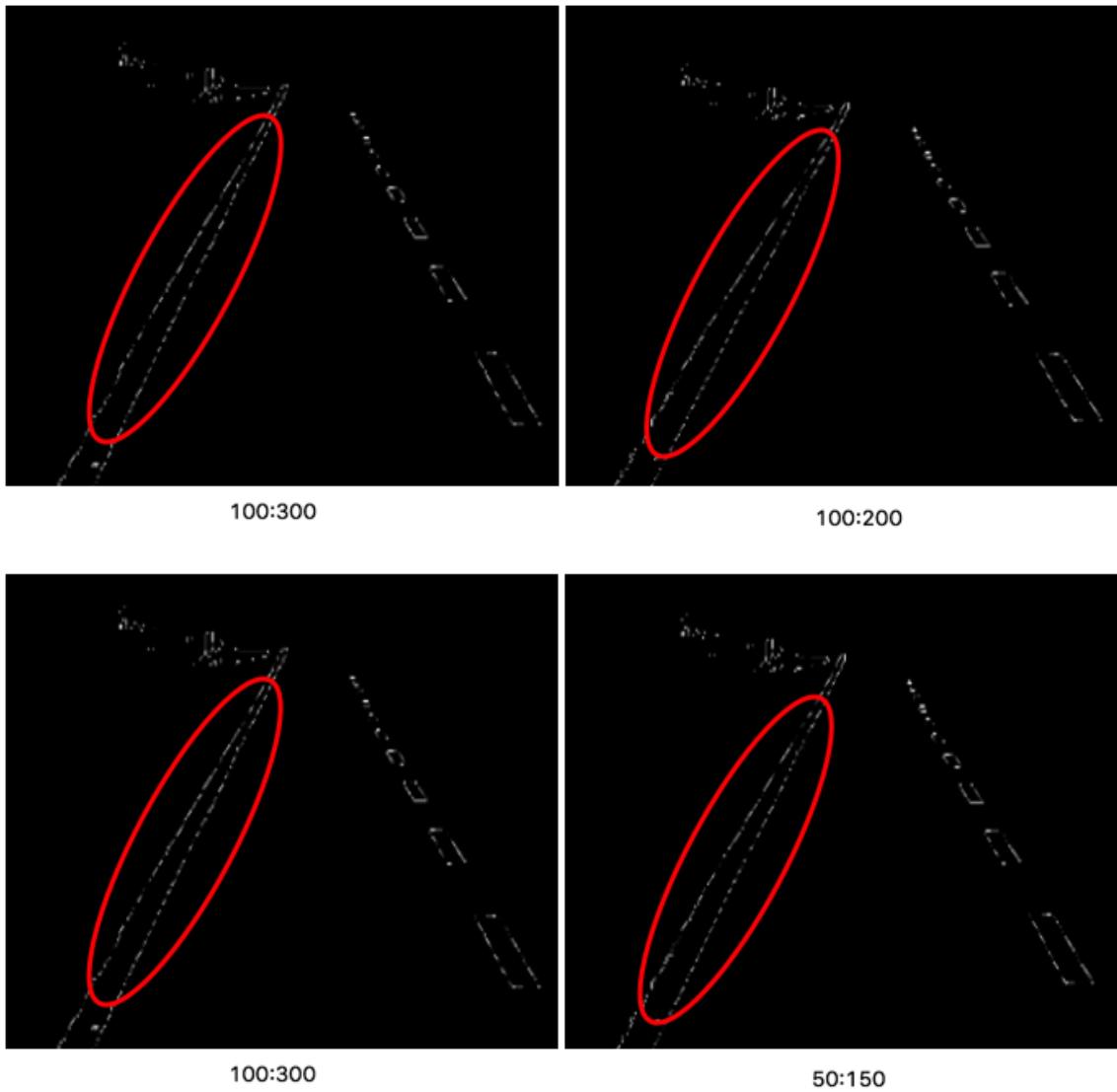


Figura 58. Comparacion de la configuración elegida frente a las no elegidas.

## 4.3. PRUEBAS Y FUNCIONAMIENTO.

En el presente apartado se mencionan algunas de las pruebas de funcionamiento que se han realizado en las diferentes fases del proyecto, así como unas comprobaciones finales:

- Funcionamiento del software de visión artificial
- Funcionamiento control de motores
- Funcionamiento global

### 4.3.1. FUNCIONAMIENTO DEL SOFTWARE DE VISIÓN POR COMPUTADOR.

Para conseguir un funcionamiento óptimo del software se han seguido una serie de pasos y se han dividido las tareas, sin intentar abarcar todo el programa de una vez (“Divide y vencerás”).

Inicialmente se realizaron fotografías y un video de una línea creada dentro de una casa (espacio cerrado) y a partir de ese video se comenzó a realizar un programa capaz de detectar la línea. En este primer programa se seleccionaba una región de interés muy pequeña, pensando que fuese lo más cercano que viese el robot. Además, no se realizaba ningún filtrado de ruido, sí se filtraba por el color blanco y, para detectar la línea, se utilizaba una función de detección de formas llamada *findContours*. La función de detección de formas geométricas se parametrizó para encontrar formas similares a un prisma, por lo que era capaz de encontrar toda la forma de la línea. Por último, al marcar la línea encontrada se colocó un indicador pensado para que fuera la guía del robot para el control de dirección, como se puede ver en la figura 58)

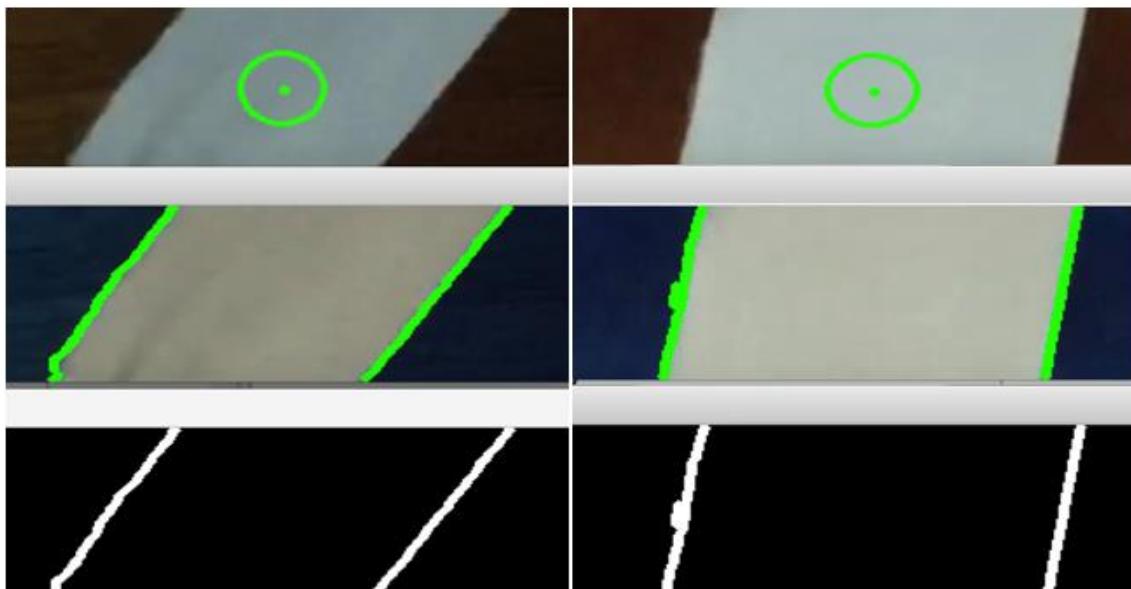


Figura 59. Resultados primer software.

El funcionamiento de este programa era satisfactorio y eficiente porque se desarrollaba en un entorno cerrado sin variaciones de luz y con la simplicidad de seguir solamente una línea. Esto no fue suficiente y se decidió descartar este código porque en ambientes exteriores confundiría muchas formas, ya que se debería aumentar la región de interés. Además, con las distintas variaciones de luz comenzaba a bajar su rendimiento y contenía mucho ruido, ya que no existían suficientes filtros.

La última e importante desventaja fue la guía, ya que con una línea funcionaba bien pero con dos líneas y giros perdía mucha eficiencia, debido también al poco rango de visión elegido, y todo el proceso para colocar esta guía relentizaba en gran medida el programa. La forma de medir la rapidez del programa se basó en la duración del video, 9 segundos, frente a la duración del proceso informático, el cual triplicaba la duración del video, siendo de 27 segundos. Esta falta de rapidez del programa, unido a la dificultad para la futura toma de decisiones en tiempo real, indicó que el programa no era viable.

Tras descartar la primera idea, se comienza a trabajar en la opción definitiva. Para esta segunda opción se realizan una serie de 8 videos de distintos tramos del carril bici y bastantes fotografías con distintas variaciones de luz. Esta serie de archivos son los utilizados para comenzar y configurar un nuevo programa.

Con esta prueba se añaden y se parametrizan de manera óptima los filtros de ruido, el rango para filtrar el color blanco se aumenta y se configura para detectar las líneas blancas en todo momento, sin importar cuánta luz incidiese sobre ellas. El mayor cambio de este programa comienza en la introducción de la función de la transformada de Hough para detectar líneas. Se debe mencionar que al introducir esta función se deben variar algunos parámetros de la función *Canny*. En primer lugar, debido al desconocimiento de la librería *OpenCV*, se comienza utilizando la función *HoughLinesP*, la cual va detectando segmentos de líneas en función de los valores que se le pasen, véase figura 59. Esto mejora el procedimiento anterior de encontrar formas geométricas pero no termina de ser totalmente eficiente porque no abarca la totalidad de la línea del carril y para situar la guía se utiliza un procedimiento similar al anterior.



Figura 60. Diferencias de resultados entre HoughLinesP (arriba) y HoughLines (abajo).

Al comprobar que el programa no es lo suficientemente rápido se investiga la librería *OpenCV* y se encuentra la función *HoughLines*, que muestra líneas que abarcan todo el carril (figura 59) y además la función nos devuelve valores rho y theta que son utilizados para la dirección y guía del robot. De esta forma, se elimina código que recorría toda la imagen buscando segmentos de líneas para colocar la guía y se utilizan las variables de la función para guiar al robot dependiendo de los casos que se identifiquen, dejando el programa mucho más depurado.

Una vez conseguido mejorar el software se comienza el proceso de traspaso del código a la *Raspberry Pi*. La primera acción consiste en probar el correcto funcionamiento de la *Pi Camera*. Este proceso se inicia con una serie de comandos y realizando fotografías y videos a lo largo de una casa. Después se ejecutan códigos sencillos de pruebas para ver que la librería *OpenCV* funciona de manera correcta. Estas pruebas se basan principalmente en filtrar el color blanco de las distintas fotos. Por último, hay que traspasar el código probado en el ordenador al microcontrolador. Este procedimiento implica algunos cambios, como la adición de las librerías necesarias para la *PiCamera*, el bucle (donde se ejecuta todo el código) se debe modificar al realizarse de forma distinta, ya que hay que indicar en qué formato debe obtener cada *frame*.

Además, se prueban distintas resoluciones para la cámara, equilibrando entre calidad obtenida de los *frames* y rapidez del programa. En este caso, al importar únicamente las líneas, no se

necesita una resolución alta y se ha elegido la más baja, 640x480 píxeles. También se han realizado ajustes en el *framerate*, optándose finalmente por un *framerate* de 30 *frames* por segundo.

#### 4.3.2. FUNCIONAMIENTO CONTROL DE MOTORES.

Una vez encaminado el software de visión, se inicia la creación del robot, montando el chasis e instalando las ruedas y los motores. Una vez se tiene el chasis, se realizan las conexiones entre motores, módulo L298N y las conexiones GPIO de la *Raspberry Pi* y comienzan las pruebas de verificación del correcto funcionamiento. Antes de montar todo en el chasis se realizan pruebas con otros motores más baratos para comprobar que los pines utilizados (13, 15, 16, 18, 29, 31, 36, 37) son los correctos y se comprueba que giran hacia delante y hacia atrás, enviando señales de “HIGH” y “LOW” a los pines indicados.

```
9 import RPi.GPIO as GPIO
10 from time import sleep
11
12 GPIO.setmode(GPIO.BCM)
13
14 MotorA = 16
15
16 GPIO.setup(MotorA,GPIO.OUT)
17
18 print "Turning motor on"
19 GPIO.output(MotorA,GPIO.HIGH)
20
21 sleep(2)
22
23 print "Stopping motor"
24 GPIO.output(MotorA,GPIO.LOW)
25
26 GPIO.cleanup()
```

Figura 61. Código de comprobación de los motores.

Para la primera prueba se crea una aplicación donde el robot es controlado a través del ordenador portátil mediante las teclas “W” (delante), “A” (izquierda), “S” (atrás), “D” (derecha). En esta prueba se comienza a configurar la fuerza de los motores en los giros para realizar giros controlados.

Gracias a esta prueba se ha descubierto que en una parte del recorrido la potencia del robot no era suficiente para subir una cuesta. Ante este error se decide poner una fuente de alimentación adicional de 6V en paralelo con la ya instalada con anterioridad. Con esta modificación se consigue más corriente y, por lo tanto, mayor potencia en los motores, solventando el problema.

Una vez que se comprueba que todo funciona correctamente se combina el código de visión con el de control de motores y así se crea el código del programa.

#### 4.3.3. FUNCIONAMIENTO GLOBAL.

Para comprobar el funcionamiento global se acude al carril bici. Se comienza creando una red de trabajo con la red del teléfono móvil y se conecta el ordenador y la *Raspberry Pi* a la red. Una vez conectados por vía SSH y VNC el ordenador es capaz de mandar los comandos necesarios al microcontrolador, es importante recordar que la *Raspberry Pi* no tiene ni teclado ni pantalla. Por medio de esta conexión, se activa el modo de grabación de la cámara y, mediante el software de control del robot por teclado, se graba el carril bici.

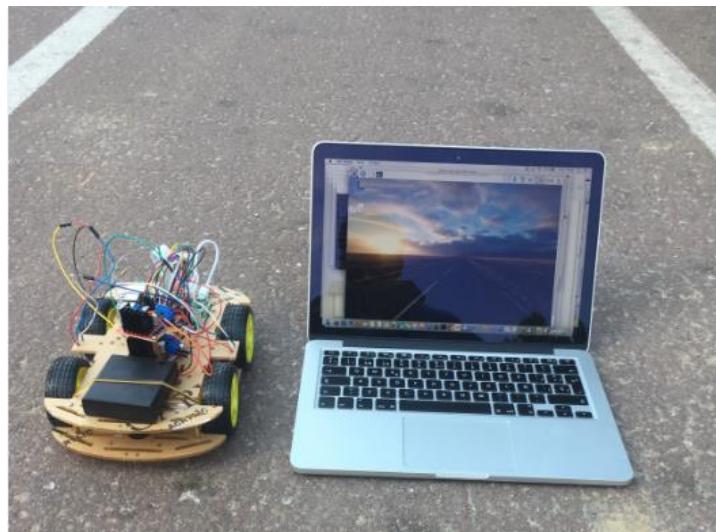


Figura 62. Robot y ordenador en las pruebas.

En las primeras grabaciones se vió que la cámara estaba situada en una posición muy baja y por lo tanto no se podían detectar las líneas del carril, por lo que se ha diseñado un nuevo soporte para subir la altura de la cámara. Con esta modificación se consigue aumentar de 8,5 centímetros a 20 centímetros de altura, lo que mejora notablemente la detección de líneas.



Figura 63. Robot en funcionamiento.

Las grabaciones se realizan en puntos críticos como son giros a la derecha, a la izquierda y en rectas, con estas grabaciones se consigue comprobar si los filtros de color y ruido son los correctos, y si los parámetros para detectar las líneas, y de control de movimiento del robot, son los adecuados. Por lo tanto se pueden distinguir tres tipos de pruebas:

**- Pruebas recta**

Con esta prueba se consigue conocer, lógicamente, los parámetros de las líneas en una recta y se amplia el filtrado del color blanco, ya que en esta sección da la sombra y por lo tanto se tiene que aumentar el rango de filtrado del color blanco para poder detectarlo.

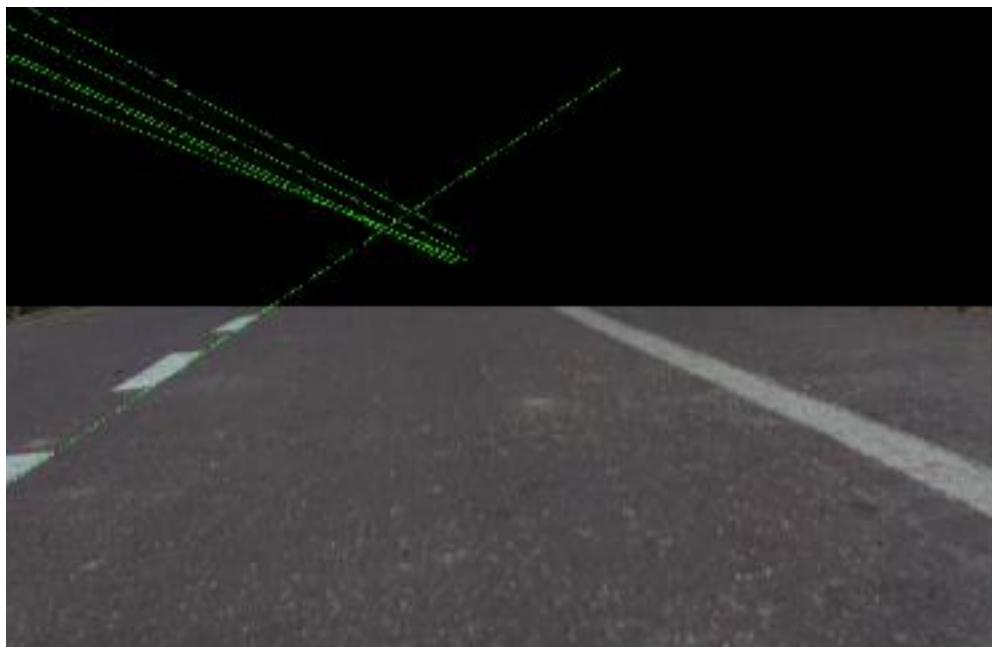


Figura 64. Visión de la recta desde la cámara del robot.



Figura 65. Visión de la recta tras el filtro Canny.

En la figura 61 se puede ver la detección de las rectas a ambos lados y en la figura 62 se puede ver el filtro *Canny* de la imagen anterior. Debido a al mal estado del carril bici se puede apreciar cierto ruido en el centro del carril, esto se consigue evitar gracias a los parámetros de la función *HoughLines*, desestimando así los falsos positivos que salgan en mitad del carril, lo que también sirve para los momentos en los que se encuentran señales pintadas en el carril.

En cambio, en la figura 63 se aprecia que el robot, en algunos momentos, sólo detecta una línea, lo que no genera ningún problema porque sigue reconociendo que debe seguir de frente. Esto se debe a las vibraciones de la cámara mientras el robot se

desplaza, y a la no existencia de líneas en algunos puntos debido al mal estado del carril bici.



Figura 66. Visión de la recta desde la cámara del robot.

Se debe mencionar que las pruebas en línea recta se realizan en un terreno con inclinación positiva y negativa, resultando correctas ambas pruebas, y funcionando de forma correcta.

#### - Pruebas curva Izquierda

Con las comprobaciones de funcionamiento en curvas se consigue encontrar los parámetros para realizar el giro en el momento adecuado, ya que, debido al reducido tamaño del robot, la curva se hace menos pronunciada, y se intenta apurar la mayor distancia posible a la línea exterior para ir realizando la curva. A continuación se pueden ver imágenes de lo que visualiza el robot mientras está circulando.

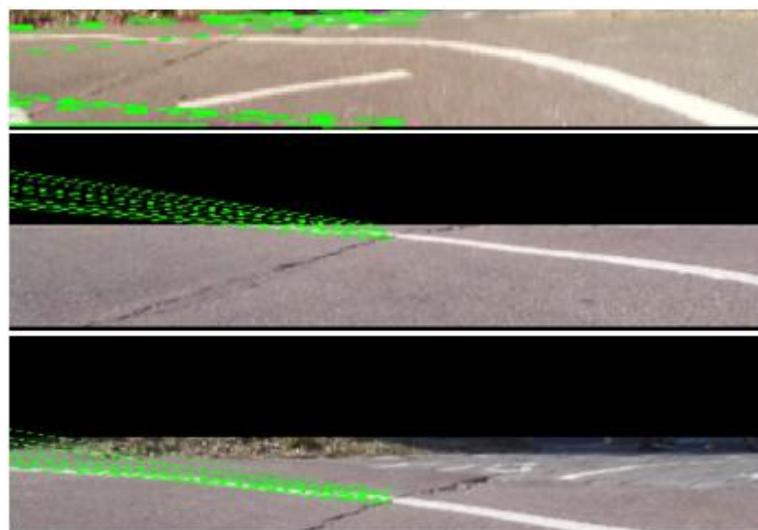


Figura 67. Distintos puntos de vista del robot.

Como en la prueba de línea recta, en estas pruebas también se consiguen encontrar los parámetros con los cuales se evitan falsos positivos por las señales, véase la figura 65

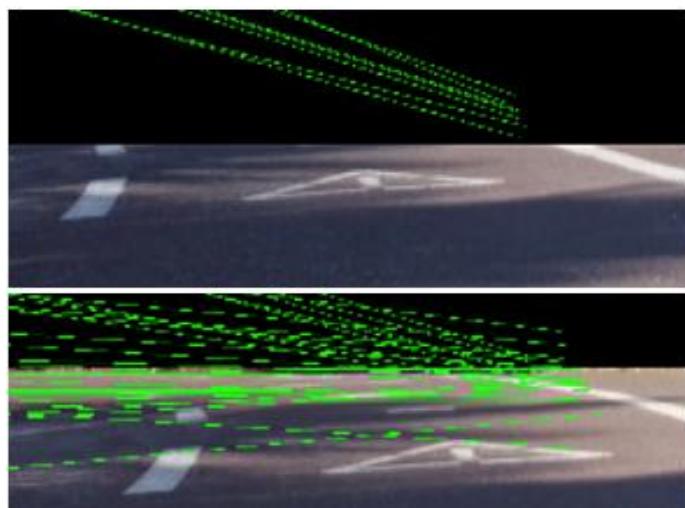


Figura 68. Utilización de Filtro (arriba); falsos positivos (abajo).

#### - Pruebas curva derecha

Para la comprobación del funcionamiento en curvas hacia la derecha se realizan las pruebas en la misma curva, que anteriormente era la curva hacia la izquierda, pero en sentido contrario.

Como en la prueba anterior, los resultados no fueron tan satisfactorios como en línea recta, es decir, existen muchas limitaciones en el robot, que posteriormente serán expuestas en el capítulo de limitaciones, lo que provoca un mayor número de errores.

A continuación se puede comprobar una imagen donde el robot detecta la línea exterior de la curva.



Figura 69. Punto de vista del robot curva derecha.

Una vez realizados los ajustes se comprueba el funcionamiento del robot en una casa, ya que si existen errores es más fácil corregirlos al tener una red doméstica configurada, teclado y monitores. Una vez se comprueba que el robot realiza los giros adecuados a partir del video, se vuelve a llevar al carril bici. En las pruebas finales, se observa que es realmente difícil eliminar todos los falsos positivos, por lo que se diseñan unos contadores para cada tipo de detección de recta. Con esta modificación se consigue obtener qué tipo de detección de líneas es el que se

encuentra en mayor número de ocasiones y, por lo tanto, se debe realizar, así se consigue otra manera de aumentar el descarte de falsos positivos.

Por último, se destaca que durante las pruebas se han notado algunos errores de detección en ciertas ocasiones y en algunos giros, por lo que ha sido necesario repetir el procedimiento anterior. Por lo tanto, se puede decir que se ha utilizado el método de prueba y error. Este procedimiento ha sido bastante complicado, ya que había que crear un ambiente de trabajo en una zona exterior con tránsito de gente y la realización de las pruebas ha sido complicada. Además, cada prueba varía dependiendo de dónde se posiciona el robot.

## 4.4. LIMITACIONES Y ERRORES.

Al ser un sistema *low cost* se encuentra varias limitaciones que se enumeran y explican en este apartado. La primera limitación se encuentra en la iluminación, debido a que se deben realizar diferentes pruebas en puntos alejados del carril bici, se encuentran grandes variaciones de iluminación entre ellos. En una de las pruebas, se encuentra una valla que al reflejar su sombra sobre el carril bici crea unas líneas rectas que el programa confunde con las líneas del carril bici, provocando un error.

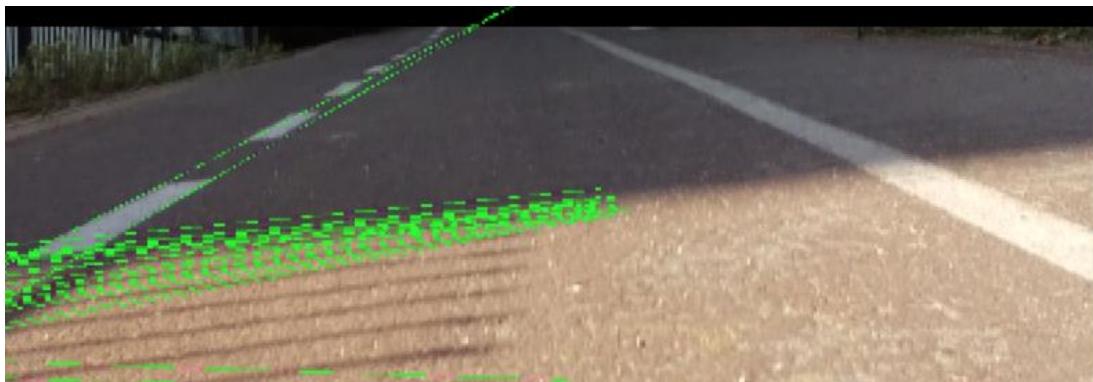


Figura 70. Falso positivo debido a la sombra de la valla.

Otra de las limitaciones es el mal estado de los carriles bici con gran suciedad y muchos baches y grietas. La suciedad y falta de mantenimiento del carril bici provoca que algunas líneas no sean visibles o que estén totalmente borradas y desaparecidas. Por otro lado, los baches y las grietas causan dos problemas: el primero es que crea movimiento en la cámara, por lo que en ciertos momentos la imagen es borrosa e incluso no apunta donde debe, provocando imágenes de mala calidad donde no se pueden detectar líneas. En segundo lugar, algunas líneas no son detectadas porque no tienen forma de línea, ya sea por una grieta o por baches lo que provoca errores en las pruebas.

Anteriormente se ha mencionado que en alguna ocasión el robot ha quedado parado en un agujero del carril bici, continuando con esta línea hay que mencionar que las ruedas fueron otra limitación, ya que con el uso han ido deteriorándose, y algunas ruedas giraban mejor que otras. Además, se encuentra la limitación de la altura de la cámara combinada con el software. Pese a cumplir el objetivo para el que fue creado, si la altura se modificase 5 centímetros hacia arriba o hacia abajo, ya no funcionaría el programa. Esto sucede porque los parámetros están elegidos desde el punto de vista de la altura actual y los ángulos cambiarían con la modificación de la altura.

Para finalizar, se ha dejado la limitación más importante. Esta limitación es la capacidad de procesamiento de la *Raspberry Pi*, que tras haber realizado las diferentes pruebas se puede afirmar que en ocasiones no es suficiente para realizar este proyecto. Se puede realizar esta afirmación porque en varios momentos se ha quedado bloqueada (también ocasionado por las altas temperaturas en el exterior). Debido a estos bloqueos, en algunos momentos no ha realizado el control de dirección que se le exige, continuando recto hasta que se ha reiniciado, y en otras ocasiones no era capaz de gestionar toda la información y lo realizaba muy

lentamente, incluso llegando a pararse. Por ello, se va a proceder a demostrar la limitación de capacidad de procesamiento frente a un ordenador normal, comparando los tiempos de ejecución del programa.

Las pruebas se realizan con un video del carril bici y utilizando un código que no incluye el movimiento del robot, lo que ralentizaría aún más los procesos. En primer lugar se van a comparar los tiempos de procesado de una imagen. Los tiempos corresponden al primer bucle del programa, debido a que es el bucle más lento. Los resultados se muestran en segundos, siendo los siguientes:

	Bucle inicial	
	Ordenador	Raspberry
Inicio	0	0
Obtener Imagen	0,263551	0,252610
Región de Interés	0,276939	0,316803
BGR a HSV	0,294633	0,465963
Filtros Ruido	0,329195	1,188649
Filtro Color Blanco	0,343384	1,244766
Filtro Canny	0,365650	1,387784
Detección Líneas	0,392707	1,624792
Inicio Nuevo	0,459488	1,658111

Tabla 8. Valores de tiempo (segundos) entre procesos.

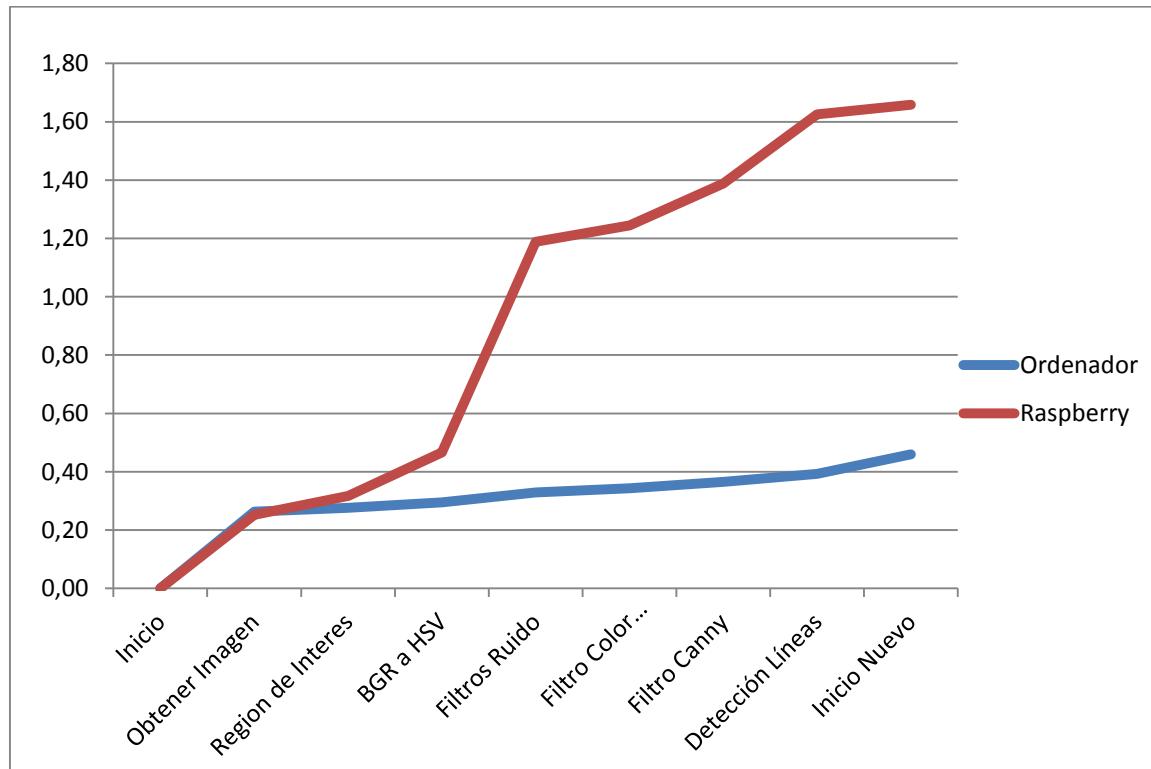


Figura 71. Valores de tiempo (segundos) entre procesos.

En segundo lugar se comparan los tiempos entre cada procesado de imagen del video. Los resultados se muestran en segundos, siendo los siguientes:

	Tiempo entre procesados	
	Ordenador	Raspberry
1	0,467623	1,508313
2	0,164980	1,349120
3	0,175044	1,345368
4	0,158467	1,380376
5	0,186797	1,366939
6	0,180469	1,370174
7	0,181669	1,362536
8	0,179187	1,379963
9	0,190284	1,372989
10	0,212090	1,347312

Tabla 9. Valores de tiempo (segundos) entre procesado de imágenes.

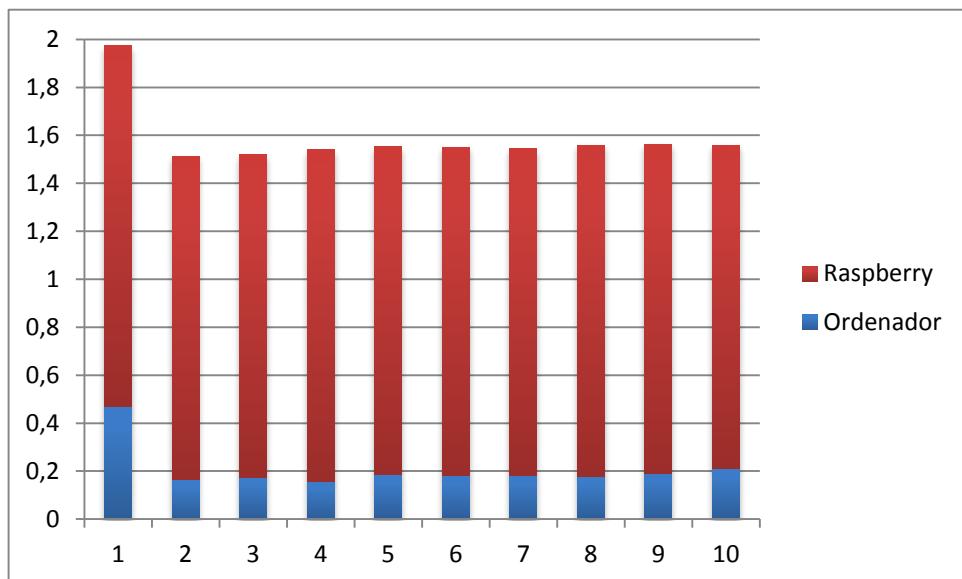


Figura 72. Valores de tiempo (segundos) entre procesado de imágenes.

# CAPÍTULO 5. PLANIFICACION.

Para la realización del proyecto se han seguido las siguientes fases, necesarias para la organización tanto del documento como de la carga de trabajo.

## Fase 1 – Búsqueda de información, elección de software y configuración del ordenador

Para comenzar el desarrollo del proyecto se ha buscado el lenguaje y las librerías más apropiadas para éste, decidiendo que la mejor opción era desarrollarlo en el software *Anaconda* y programar en el lenguaje *Python* utilizando las librerías *OpenCV*, ya que las librerías *SimpleCV*, más avanzadas que *OpenCV*, ocasionaban problemas de compatibilidad con la *Raspberry Pi*.

Una vez que se ha tenido la información requerida se ha realizado la búsqueda de los componentes necesarios. De cada componente se han buscado varias marcas y tipos para posteriormente seleccionar los componentes que tuviesen buen funcionamiento y bajo precio. Una vez elegidos todos los materiales, se han realizado las compras, que han sufrido un retraso considerable por problemas en la aduana.

## Fase 2 - Desarrollo y programación del software

Mientras la compra de material se retrasaba se ha comenzado a instalar los programas necesarios en el ordenador, *Anaconda (Python)*, las librerías *OpenCV* y programas para la conexión remota a la *Raspberry Pi*, *VNC server* y *Teamviewer*.

Una vez aprendidos los conceptos básicos de *Python* y realizados algunos ejercicios utilizando *OpenCV* para coger destrezas, se inicia el desarrollo del software. En primer lugar se ha comenzado por lo más sencillo, creando una aplicación capaz de seguir una línea del carril bici, y posteriormente se ha modificado el código para que fuese un robot capaz de mantenerse entre las dos líneas del carril bici. La fase de programación se ha prolongado hasta la finalización del proyecto, como se puede ver en el diagrama de Gantt (figura 70).

## Fase 3 - Configuración *Raspberry Pi*

A la vez que se iba desarrollando la aplicación se ha configurado la *Raspberry Pi*, descargando el sistema operativo *Raspbian* en una tarjeta microSD, creando una imagen que posteriormente se instalaría en la *Raspberry Pi*. Una vez instalado el sistema operativo se ha configurado para poder descargar todo lo necesario para el correcto funcionamiento de los distintos dispositivos del robot.

## Fase 4 - Diseño hardware del robot

Una vez configurados todos los dispositivos, actualizados y en funcionamiento, se procede a diseñar el robot. Se comienza por montar el chasis con los motores y las ruedas, después se decide dónde irían colocados los distintos elementos, para un buen reparto del peso; se instala la cámara en el microcontrolador, y se realizan las conexiones necesarias para que funcionen los motores a las órdenes del microcontrolador.

### Fase 5 - Comunicación ordenador – *Raspberry Pi* inalámbrica

Para tener un control del proyecto en el exterior ha sido necesario tener una comunicación inalámbrica entre el ordenador y el microcontrolador. También es útil para poder realizar modificaciones y ver *in situ* lo que procesa la cámara en las pruebas realizadas en el carril bici, además de la libertad y comodidad para trabajar en el proyecto sin necesidad de cables y conexiones extras.

### Fase 6 – Programación final en la *Raspberry Pi*

Por último, se han realizado unos cambios en el software debido a que la cámara de la *Raspberry Pi* tiene sus propias librerías, a diferencia de la cámara web del ordenador. Esta fase ha sufrido un retraso debido a la rotura de la *Raspberry Pi* y la espera de recibir una nueva. Este imprevisto en la planificación, aunque se ha solucionado con rapidez, ha provocado un retraso en el proyecto, ya que se ha reconfigurado la *Raspberry Pi*. Esta incidencia se refleja en el presupuesto, dónde se indica el coste de ambas *Raspberry Pi*.

### Fase 7 – Pruebas

Esta fase se ha realizado durante gran parte del proyecto, intensificándose en el tramo final. Las pruebas y la programación han ido paralelas en la planificación, ya que se necesitan entre sí, si no sería imposible realizar un buen programa informático.

En la siguiente tabla se detallan las tareas necesarias para completar las fases anteriormente descritas, y a continuación, su diagrama de Gantt.

Tarea	Inicio	Fin
Búsqueda de Información	01/11/16	10/11/16
Búsqueda y Elección de Material	10/11/16	01/12/16
Compra de Material	01/12/16	16/01/17
Instalación Programas	03/12/16	05/12/16
Aprendizaje Python y OpenCV	05/12/16	20/01/17
Programación Software	21/01/17	25/05/17
Instalación Raspberry Pi	05/12/16	07/12/16
Montaje Robot	12/02/17	19/02/17
Pruebas	20/02/17	01/06/17

Tabla 10. Tareas y fechas de la planificación.

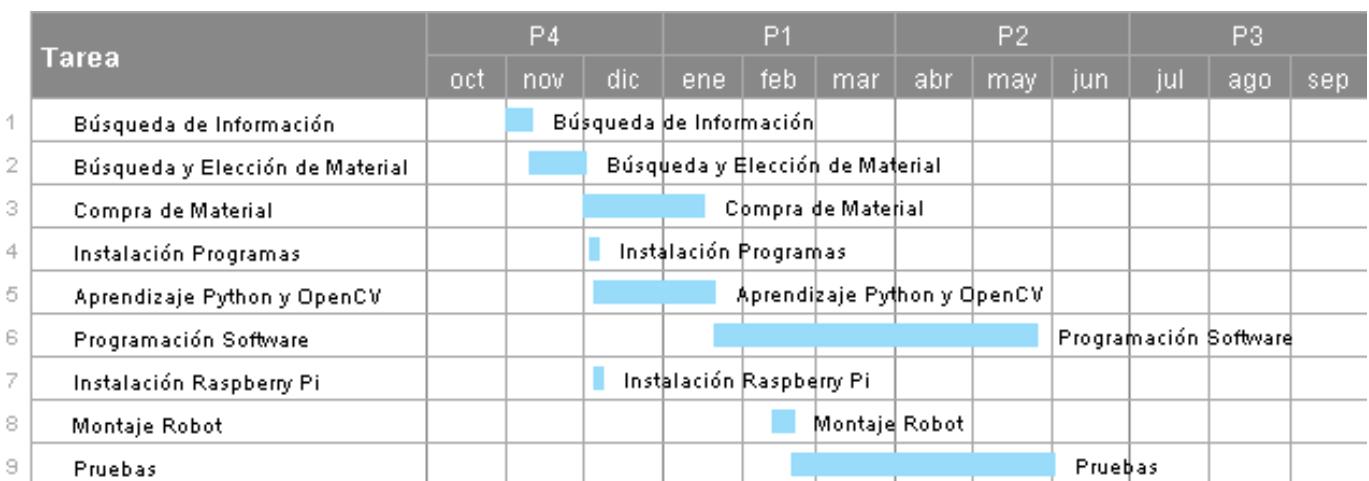


Figura 73. Diagrama de Gantt.

## CAPÍTULO 6. MARCO REGULADOR.

En la actualidad no existen leyes que regulen la robótica ni la inteligencia artificial. Sin embargo, varios países están empezando a redactar códigos regulatorios y éticos para implantar en sus marcos reguladores. En este capítulo se va a considerar que nuestro proyecto se desarrolla en Europa y más concretamente en España, por lo que los marcos reguladores de otros países no serán considerados. Además, se considerará que el proyecto pretende ser comercializado, ya que si no fuese así no debería cumplir ciertas normativas que posteriormente serán mencionadas.

El tráfico y la conducción es un área altamente regulada en los países de la Unión Europea. Implantar la tecnología de conducción autónoma cambiará muchos aspectos de la misma, entre ellos todo lo relacionado con la seguridad y la responsabilidad, es por ello que es necesaria una reconsideración del marco regulador convenido hasta ahora. [43]

Para realizar las pruebas en el carril bici se ha tenido que cumplir con la normativa reflejada en el Real Decreto Legislativo 6/2015, por el que se aprueba el texto refundido de la Ley sobre Tráfico, Circulación de Vehículos a Motor y Seguridad Vial. También se debería cumplir el marco legal que se regularizó en noviembre de 2015 en España mediante Instrucción de la Dirección General de Tráfico, anunciada por el Ministerio del Interior en una Nota de Prensa. Este marco legal regula la realización de pruebas con vehículos de conducción autónoma en las vías españolas. Dicha instrucción regula los requisitos para la solicitud y concesión de la autorización de pruebas o ensayos de investigación realizados con vehículos de conducción automatizada en vías de tráfico general. Se trata de una autorización de ámbito nacional que establece los tramos de vía urbana e interurbana por los que el vehículo está autorizado para realizar las pruebas y ensayos. La autorización tiene una vigencia máxima de 2 años, pudiendo prorrogarse sucesivamente.

En cuanto a los requisitos, el vehículo autónomo deberá contar con contrato de seguro en vigor que cubra la cuantía de los límites del aseguramiento obligatorio para vehículos a motor en España, así como la responsabilidad civil derivada de posibles daños causados. Se requiere que el vehículo autónomo haya superado en algún servicio técnico acreditado por la Entidad Nacional de Acreditación (ENAC) los procedimientos que se recogen en la Instrucción.

En cuanto a la documentación a aportar, es necesario presentar un documento de Evaluación de Riesgos basado en el HARA (*Hazard Risk Analysis*), que forma la base de toda la actividad de seguridad funcional acorde a la ISO 26262 y el FMEA (*Failure Mode Effects Analysis*). [44]

En cuanto a responsabilidades civiles y éticas todavía no existen leyes; sin embargo, es importante mencionar que el Parlamento Europeo ya está trabajando en un Informe de Recomendaciones (2015/2103, de 31 de mayo 2016) sobre normas de Derecho Civil sobre robótica, en el que se facilitan pautas para regular la Responsabilidad Civil derivada del uso de robots. Se refiere a las responsabilidades contractuales y extracontractuales que pueden derivarse de su actuación, recomendando configurar como objetiva dicha responsabilidad y

estableciendo la necesidad de disponer de un seguro obligatorio de Responsabilidad Contractual por los daños derivados de la tenencia y uso de robots.[45]

El mismo Informe de Recomendaciones señala que el potencial de empoderamiento que encierra el recurso a la robótica se ve matizado por una serie de tensiones o riesgos relacionados con la seguridad humana, la intimidad, la integridad, la dignidad, la autonomía y la propiedad de los datos. Considera que es preciso un marco ético que sirva de orientación en materia de diseño, producción y uso de los robots, a fin de complementar las recomendaciones jurídicas expuestas en el presente informe, y el acervo nacional y de la Unión en vigor y propone un marco en forma de carta que comprenda un código de conducta para los ingenieros en robótica, un código deontológico destinado a los comités de ética de la investigación para la revisión de los protocolos de robótica, y licencias tipo para los diseñadores y los usuarios. También señala que este marco ético orientador debe basarse en los principios de beneficencia, no maleficencia y autonomía, así como en los principios consagrados en la Carta de los derechos fundamentales de la Unión, como la dignidad humana y los derechos humanos, la igualdad, la justicia y la equidad, la no discriminación y no estigmatización, la autonomía y la responsabilidad individual, el consentimiento informado, la privacidad y la responsabilidad social, además de en los actuales códigos y prácticas éticas.

Por último el informe pide a los diseñadores de robots tener en cuenta y respetar la integridad física, la seguridad, la salud y los derechos de las personas. Un ingeniero en robótica debe preservar el bienestar sin dejar de respetar los derechos humanos, y divulgar con prontitud los factores susceptibles de poner en peligro a la población o al medio ambiente. [46]

En cuanto a normativa de estandarización dependería del uso final del proyecto y de su comercialización. En el caso de que se utilizase para guiar un vehículo de mayor tamaño se debería cumplir la norma ISO 26262 (Automóviles – Seguridad Funcional) para los sistemas de seguridad en los automóviles. La ISO 26262 define un marco y un modelo de aplicación, así como las actividades, los métodos y los resultados. La aplicación de esta norma tiene como objetivo garantizar la seguridad funcional de un sistema eléctrico o electrónico en un vehículo motor. Este estándar se deriva de la norma IEC 61508 para su uso específico en el sector del automóvil. [47]

Para finalizar este apartado se debe mencionar que el proyecto no va a crear ninguna patente, ni se pretende proteger el código del software. No sólo no se pretende proteger el código, sino que el objetivo es que el proyecto sea de código abierto, por lo que debería tener una licencia de código libre, como puede ser GPL, ofrecida por la *Free Software Foundation* (FSF).

## CAPÍTULO 7. ENTORNO SOCIO-ECONÓMICO.

### 7.1 PRESUPUESTO.

En este capítulo se detalla la estimación de los costes de este Trabajo de Fin de Grado. Para su fácil compresión se presentan los costes en diferentes apartados:

- Coste de personal
- Coste de software y hardware
- Costes indirectos y amortizaciones

Para la compresión del presupuesto hay que tener en cuenta los siguientes aspectos:

- Todos los precios irán en euros
- No se tomarán más de dos decimales, redondeando si fuera necesario.

#### 7.1.1. COSTE DE PERSONAL.

En este apartado se procede a detallar las personas que se han involucrado en el proyecto, así como la cantidad de tiempo y el precio por hora

Código	Descripción	Unidad	Cantidad	Precio	Precio Total	
1.1	<b>Ingeniero industrial electrónico.</b> Ingeniero industrial electrónico con conocimientos básicos en programación de visión por computador, programación en Python y circuitos.	Mes/media jornada	4	750,00	3.000,00	€
1.2	<b>Doctor Ingeniero industrial.</b> Tutor, amplios conocimientos en visión por computador y redes neuronales. Enrique Pelayo Campillos	Horas	20	60,00	1.200,00	€
			<b>Coste Total</b>		<b>4.200,00</b>	<b>€</b>

Tabla 11. Coste de personal.

## 7.1.2. COSTE DE SOFTWARE Y HARDWARE.

Aquí se detalla el cálculo del coste del material empleado. Hay que tener en cuenta que todo el software usado era software libre, por tanto no existen gastos por este concepto.

Código	Descripción	Cantidad	Precio Ud.	Precio Total	
2.1	<b>Raspberry Pi 3 Model B.</b> Microcontrolador usado como controlador principal del sistema.	2	42,90	85,80	€
2.2	<b>Módulo L298N.</b> Controlador de motores, usado en el control de dirección.	1	60,00	60,00	€
2.3	<b>1ª Opción de chasis.</b> Primera opción de chasis. Conjunto de chasis, 3 ruedas, 2 motores.	1	31,20	31,20	€
2.4	<b>2ª Opción de chasis.</b> Segunda opción de chasis, elección final. Conjunto de chasis, 4 ruedas, 4 motores y soporte para baterías.	1	34,90	34,90	€
2.5	<b>Raspberry Pi Camera Board v2.</b> Cámara de 8 Megapíxeles. Será la cámara instalada en el robot.	1	29,95	29,95	€
2.6	<b>Módulo HAT para Raspberry Pi.</b> Módulo especialmente construido para Raspberry Pi que realiza la función de un controlador de motores.	1	22,50	22,50	€
2.7	<b>Batería USB - 2200 mAh.</b> Batería USB con capacidad de 5V y un 1A de salida, alimenta el robot en entornos exteriores.	1	14,95	14,95	€
2.8	<b>Motores DC.</b> Repuestos ante posibles roturas o fallos.	2	3,50	7,00	€
2.9	<b>Soporte de 4 baterías.</b> Soporte de 4 baterías tipo AA con un switch On/Off.	1	2,95	2,95	€
2.10	<b>Cableado robot.</b> Cables utilizados para las conexiones	4	3,43	13,72	€
2.11	<b>Cableado conexiones multimedia.</b> Cables HDMI, USB y Ethernet.	1	22,59	22,59	€
2.12	<b>Tarjeta MicroSD.</b> Tarjeta MicroSD Samsung EVO 16 GB.	2	7,90	15,80	€
			Coste Total	341,36	€

Tabla 12. Coste de hardware y software.

### 7.1.3. COSTES INDIRECTOS Y AMORTIZACIONES.

Se tienen en cuenta los costes indirectos y además se cuenta con una estimación de los costes de amortización de los equipos usados.

Según la nueva Ley aprobada el 27 de Noviembre de 2014 con fecha de entrada en vigor el 1 de Enero de 2015 del Impuesto sobre Sociedades, el coeficiente lineal máximo de amortización para equipos electrónicos destinados a procesos de información es de un 25% anual (en un total de 4 años). [48].

Código	Descripción	Precio Total	Coeficiente Amor.	Amor. Mes	Nº Meses	Precio Total	
3.1	<b>Ordenador MacBook Pro.</b> Ordenador usado para realizar proyectos y prácticas.	1.274,34	25%	26,54	4	106,20	€
3.2	<b>Gastos Internet.</b> Consumo de red móvil.					20,00	€
3.3	<b>Gastos de envío.</b> Gastos de envío de los materiales y aduana.					88,79	€
3.4	<b>Gasto de luz.</b> Gasto de luz y baterías					56,00	€
						Coste Total	270,99 €

Tabla 13. Costes indirectos.

### 7.1.4. COSTE TOTAL.

En este apartado se muestra el coste total, suponiendo que este trabajo hubiera sido realizado por una empresa:

Partida	Descripción	Precio Total	
1	Coste personal	4.200,00	€
2	Coste software y hardware	341,36	€
3	Costes indirectos y amort.	270,99	€
Total		4.812,35	€
	IVA (21%)	1.010,59	€
<b>Coste Total</b>		<b>5.822,94</b>	<b>€</b>

Tabla 14. Coste total.

Por tanto, el coste total del proyecto realizado es de **5.822,94 € (cinco mil ochocientos veintidós euros con noventa y cuatro céntimos).**

## 7.2 IMPACTO SOCIO-ECONÓMICO.

Debido a que el proyecto no puede venderse, se va a realizar el impacto socio-económico sobre una posible aplicación que se base en los principios del proyecto (sencillez de sistema, y entorno sin tráfico). La aplicación elegida es un vehículo capaz de transportar medicamentos y objetos en el entorno de un hospital, ya sea en interiores o exteriores.

Para realizar esta aplicación se va a necesitar una mejora de la plataforma, instalando una de mayor potencia capaz de realizar nuevas funciones, como evitar obstáculos y mejorar la rapidez del sistema. Con el nuevo microcontrolador se instalará una interfaz gráfica en el propio vehículo para elegir donde se desea enviar los objetos.

Además de la cámara, se necesitará un LIDAR para mejorar la detección de obstáculos y la navegación y no se debe olvidar un GPS para cumplir dos funciones, localizar en todo momento a los distintos vehículos y que el propio vehículo conozca a dónde debe dirigirse. Otra modificación será un chasis con plataforma elevadora, y ruedas, con odometría, de mayor tamaño y calidad, y unos motores con más potencia, mejorando la estabilidad y la velocidad del robot móvil, consiguiendo un rango de velocidades de 0 a 5 Km/h y potencia suficiente para arrastrar una carga máxima de 50 Kg. Para finalizar las mejoras de hardware, se deberá instalar un sistema de luces, ya que debe poder trabajar en situaciones de baja luminosidad.

Por último, al haber realizado una instalación de nuevos sensores y al tener que desarrollar nuevas funciones, se creará un nuevo software mejorado, utilizando la información obtenida por los nuevos sensores y componentes.

A continuación se detallará la estimación de los costes de esta aplicación. Para la compresión del presupuesto hay que tener en cuenta los siguientes aspectos:

- Todos los precios irán en euros
- No se tomarán más de dos decimales, redondeando si fuera necesario.
- No se calcularán impuestos debido a que la venta se puede realizar en otros países.

### 7.2.1. COSTES Y BENEFICIOS DE LA EMPRESA.

Partida	Descripción	Precio Total	
1	<b>Coste personal</b>	5.000,00	€
2	<b>LIDAR</b> Hokuyo UST-05LN Scanning Laser Obstacle Detection. [49]	1.295,00	€
3	<b>Localizador GPS</b> Livewire Micro GPS Vehicle Tracker. [50]	79,99	€
4	<b>Mejoras chasis, ruedas y motores</b>	1.000,00	€
5	<b>Luces</b>	18,95	€
6	<b>Microcontrolador</b> Jetson TX2 de Nvidia. [51]	499,00	€
7	<b>Cámara</b> 13 MP Jetson TX1 Camera Board. [52]	299,00	€
8	<b>Bastidor</b>	5.000,00	€
9	<b>Material mecánico y mecanizados</b>	2.000,00	€
Total		15.191,94	€
	<b>Beneficio del 30%</b>	4.557,58	€
	<b>Precio venta unidad</b>	19.749,52	€

Tabla 15. Costes y beneficios de la empresa.

Para la empresa creadora del robot móvil, los gastos ascienden a **15.191,94 € (quince mil ciento noventa y un euros con noventa y cuatro céntimos)**. La empresa busca un beneficio del 30%, lo que le generaría unos ingresos de **4.557,58 € (cuatro mil quinientos cincuenta y siete euros con cincuenta y ocho céntimos)**. Por ello, la suma de estas dos cantidades proporciona el precio de venta por unidad que será de **19.749,52 € (diecinueve mil setecientos cuarenta y nueve euros con cincuenta y dos céntimos)**.

Una vez realizada la previsión de ventas se espera que el primer año se amorticen costes, ya que se estima una venta de 5 robots, lo que supone unos ingresos de **22.787,90 € (veintidós mil setecientos ochenta y siete euros con noventa céntimos)** con los que se financia el coste del desarrollo e implementación de un sistema de intercambio de información entre los robots y el sistema central en la empresa que recibe los robots. En el segundo año se duplican los ingresos obtenidos y en los años siguientes se consigue un beneficio que cuadriplica los costes.

## CAPÍTULO 8. LÍNEAS FUTURAS.

A continuación se comentarán una serie de mejoras y avances que se podrían implantar en el sistema propuesto. Para conseguir estas mejoras es importante saber que se necesitaría realizar una gran inversión.

### 8.1 MEJORAS DE LA CAPACIDAD DE PROCESAMIENTO.

Como se ha podido comprobar en el apartado anterior de limitaciones es bastante importante mejorar la rapidez de procesado de nuestro robot. Estas mejoras deben ser de software y de hardware, ya que para la conducción autónoma es necesario una mayor rapidez de trabajo especialmente, al tener que tomar varias decisiones simultáneas en tiempo real. Por lo tanto se necesita que trabaje de forma rápida y segura para mejorar la seguridad.

Aunque se ha mencionado que las mejoras deben proceder tanto de software como de hardware, se debe hacer realmente hincapié en el microcontrolador. Es necesario instalar un microcontrolador mucho más potente como puede ser una placa Jetson TX2 de Nvidia, con un valor muchísimo más alto, 599 euros frente a los 40 euros de la Raspberry Pi utilizada, pero con unas prestaciones bastante altas.

Una vez mejorada la placa central se podría utilizar nuevas librerías mucho más avanzadas como SimpleCV, la cual en la Raspberry Pi daba muchos problemas. Esta librería simplificaría y agilizaría multitud de procesos mejorando en gran medida la rapidez del programa.

### 8.2 MEJORAS VISIÓN POR COMPUTADOR.

Siempre y cuando se mejore el microcontrolador se podría implementar funciones que no han sido posible implementarse por las limitaciones de la Raspberry Pi.

La posible primera mejora estaría relacionada con la iluminación inadecuada y el rango del filtro de color blanco. Se debería trabajar en conseguir conocer a través de un software y la cámara cuanta luminosidad existe en la imagen. Una vez conocido este dato, en función de su valor debería variarse el rango de filtro del color blanco y así siempre conseguir detectarlo sin importar las sombras o destellos que existan. Con esta modificación se conseguiría evitar tener ruido y que en ocasiones no se vean en su totalidad las líneas del carril bici, mejorando la detección de las líneas y evitando posibles errores. Adicionalmente, debido a que se ha mencionado anteriormente en el documento, podría mejorarse la detección de la altura de la cámara y ajuste de los parámetros para la detección de líneas.

La primera función que sería un gran avance es la detección de obstáculos, esta función se intentó implementar en el proyecto actual de forma fallida debido a que la velocidad de procesamiento del robot era lenta y la velocidad de movimiento era cercana a nula.

El proceso que se intentó realizar se basaba en la función de OpenCV, “Optical Flow”, la cual va marcando siluetas si encuentra objetos en la imagen. Por lo tanto se intentó guardar la imagen anterior e ir comparándola con la actual y si se encontraba una trayectoria se identificaba como

objeto y debía evitarse, rodearlo si era posible y posteriormente volver al carril o pararse si este proceso no era factible.

También al mejorar la placa central se podría realizar un reconocimiento de personas o de señales, ya sean situadas en postes o en el propio carril bici. De esta forma teniendo las señales reconocidas en una base de datos o realizando un trabajo previo de “deep learning” con nuestro robot, este podría saber que debe hacer al ver esas señales.

Por último, mencionar que siempre se ha trabajado en un carril bici de dos carriles por lo que se podría mejorar para trabajar en diferentes entornos no solo en el probado en el proyecto.

## 8.3 MEJORAS HARDWARE.

Debido al bajo presupuesto se prescindió de todo tipo de sensores excepto de la cámara, por lo que una gran mejora sería incluir sensores de presencia (laser o infrarrojos) para detectar presencia de obstáculos u otros vehículos alrededor del robot, donde la cámara no llega a ver.

Además se podrían mejorar los componentes actuales como un mejor chasis, más grande y más robusto, con mejores motores y ruedas de mejor calidad, no de plástico como las empleadas en el proyecto, ya que en ocasiones unas tienen menos rozamiento que otras y los giros no son precisos. Además, un robot de mayor tamaño mejoraría las vibraciones de la cámara debido a las irregularidades del terreno. También se podrían añadir luces y componentes que emitan algún tipo de sonidos por si tiene que emitir alguna alerta.

## 8.4 MEJORAS SOFTWARE.

Debido a que el proyecto se basaba en navegación a través de visión por computador la opción que será descrita a continuación no fue añadida pero es una variante que puede ser de gran utilidad.

La función sería poder alternar entre sistema de navegación autónoma y si en algún momento se deseara se podría cambiar a sistema de manejo manual a través de un portátil. Este no sería muy difícil, ya que inicialmente para realizar las pruebas se utilizó un programa en el que el robot era manejado desde un portátil.

Otra opción similar a la descrita podría ser crear una aplicación móvil para sistemas como IOS (Apple), Android o Windows Phone, donde se pudiese conectar al robot vía internet o bluetooth y elegir cómo controlar el robot y en cualquier caso, nos fuese mostrando lo que el robot está viendo en tiempo real.

En el apartado de limitaciones se mencionó un problema con la modificación de la altura de la cámara, por tanto sería una gran mejora desarrollar una función para conocer a qué altura se encuentra la cámara a partir del ángulo de las líneas que se encuentran en la imagen o de la línea del horizonte. Una vez se conociese la altura se debería realizar otra función para modificar los parámetros de la función “HoughLines” para detectar las líneas.

## **8.5 MEJORAS NAVEGACIÓN.**

Actualmente en el robot no existe modo para obtener información sobre el espacio recorrido, lugar donde se encuentra el robot y trayectoria que piensa seguir.

Dependiendo del futuro uso de este proyecto podría ser útil añadir un GPS y que este diese opción, por ejemplo a pasajeros de un campus universitario si prefieren ir por un lugar o por otro. Como añadido, al instalar el GPS se le estaría dando una función más al robot, la cual sería que los futuros clientes que deseen viajar en el vehículo sepan donde se encuentra el más cercano o que el propio robot fuese capaz por sí mismo de llegar hasta la localización de los clientes, creando por si solo una ruta.

## **8.6 OPEN SOURCE.**

Se pretende que el código del proyecto sea código abierto, algo muy común en la actualidad. Consiste en que el programa no solo pueda ser modificado por la persona que lo ha realizado si no que partiendo de una base y con unos controles la gente pueda modificar el código y crear nuevos programas o nuevas utilidades para el robot.

También se pueden crear programas paralelos, módulos o aplicaciones que no estén basadas en el código del proyecto pero complementen al proyecto. Todas estas mejoras pasarían un control y se podrían subir a un servidor. Teniendo los distintos módulos, mejoras y aplicaciones en un servidor se tiene la posibilidad de que cualquier cliente si le gusta una idea pueda descargársela.

De esta forma conseguimos que el proyecto, si es interesante para las personas, no quede parado por falta de tiempo o dedicación del creador. Además se amplía el mercado, ya que es mucho mejor comprar un servicio que puedes adecuar a tu gusto y que ofrece un gran abanico de posibilidades y usos. Por último, es una manera de mejorar el proyecto de forma gratuita y sin realizar una gran inversión solo una página web y el coste del servidor y su mantenimiento, lo que se puede considerar pequeño comparándolo con lo que supondría pagar el coste de personal trabajando en todo lo mencionado anteriormente.

## CAPÍTULO 9. CONCLUSIONES.

La visión artificial es una tecnología muy joven pero con una gran expectativa de futuro, en la que se está invirtiendo mucho para conseguir que esté al alcance de todos gracias a librerías como *OpenCV*.

En condiciones ideales de iluminación, buen contraste, ausencia de otros obstáculos, etc. el desarrollo de este tipo de programa sería mucho más sencillo. Pero la realidad es bien distinta, lo normal es que las condiciones ideales se den muy pocas veces, por eso, es importante seguir trabajando en este aspecto y evitar que este tipo de factores no impidan el buen funcionamiento de este y otro tipo de programas del mismo área.

Quizá este sea uno de los impedimentos más grandes en este campo, controlar todas las situaciones que puedan darse y actuar en consecuencia. Algo de vital importancia ya que en juego está la seguridad de las personas.

El número de sistemas que aplican la visión por computador están en auge debido a la cantidad de posibilidades que aporta aunque a la vez presenta una gran dificultad, entre otras cosas, por lo ya comentado de las numerosas variables y la infinidad de situaciones que se pueden encontrar. Un ejemplo es que cada vez salen al mercado más vehículos que incluyen estas tecnologías lo que hace pensar que tendrán una gran aceptación e impulsará un mayor desarrollo.

El proyecto tiene como objetivo llegar a diferentes entornos de trabajo, algunos de ellos complicados como el carril bici donde se ha trabajado. Por lo tanto, se piensa que esta tecnología podría llegar a muchas otras áreas como la agricultura, guiando tractores para realizar trabajos de agricultura.

En este proyecto, se ha diseñado un sistema de conducción autónoma *low cost*, capaz de crearse por un coste mínimo y con un consumo por parte de los dispositivos muy reducido.

Pese a que toda la tecnología *Raspberry Pi*, tanto placas como entornos de desarrollo, son una herramienta muy útil para aprender y desarrollar aplicaciones no muy complejas, a la hora de realizar proyectos avanzados, como es el controlador principal de un sistema guiado por visión artificial, tiene bastantes limitaciones. Sin embargo, aunque en un principio existen muchas limitaciones trabajando con estos dispositivos, con una mayor inversión, sobretodo centrada en el microcontrolador, se podrían conseguir resultados más óptimos y elevar de forma elevada la calidad del sistema.

Este proyecto sirve de guía para desarrollar un sistema guiado por visión por computador de bajo presupuesto, en el que cualquiera con unos conocimientos básicos de programación de microcontroladores y electrónica en general pueda hacer un sistema a medida con el objetivo que desee.

## CAPÍTULO 10. BIBLIOGRAFÍA.

1. Ventura Rojo, Carles. "El futuro de la conducción autónoma". Julio 2016. <http://informatica.blogs.uoc.edu/2016/07/14/el-futuro-de-la-conduccion-autonoma/> [Último acceso: junio 2017].
- 2."Robotics 2020 Multi-Anual Roadmap. For robotics in Europe. Horizon 2020 Call ICT-2016(ICT-25 & ICT-26)". SPARC (*The Partnership for Robotics in Europe*). Diciembre 2015.
- 3."Vehículo aéreo no tripulado". En línea.  
[https://es.wikipedia.org/wiki/Veh%C3%ADculo\\_a%C3%A9reo\\_no\\_tripulado](https://es.wikipedia.org/wiki/Veh%C3%ADculo_a%C3%A9reo_no_tripulado) [Último acceso: mayo 2017].
4. Web corporativa de ELIMCO. "Líneas de negocio de UAV". En línea.  
[http://www.elimco.com/c\\_Aviones-No-Tripulados\\_10.html](http://www.elimco.com/c_Aviones-No-Tripulados_10.html) [Último acceso: mayo 2017].
5. "Autonomous underwater vehicle". En línea.  
[https://en.wikipedia.org/wiki/Autonomous\\_underwater\\_vehicle](https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle) [Último acceso: mayo 2017].
6. Web corporativa de EMBENTION. "UGV, Aplicaciones y funciones para uso profesional". Enero 2016. <https://www.embention.com/es/news/ugv-aplicaciones-funciones-profesionales/> [Último acceso: mayo 2017].
7. "Unmanned ground vehicle". En línea.  
[https://en.wikipedia.org/wiki/Unmanned\\_ground\\_vehicle#cite\\_note-Gage-6](https://en.wikipedia.org/wiki/Unmanned_ground_vehicle#cite_note-Gage-6) [Último acceso: mayo 2017].
8. Litman, Todd. "Autonomous Vehicle Implementation Predictions. Implications for Transport Planning". Mayo 2017. (Victoria Transport Policy Institute).
9. Wolfgang Bernhart, Marc Winterhoff, Christopher Hoyes, Venka Tachi Vukula, Jens Garrelfs, Sage Jung, Sven Galander. "Autonomous driving. Disruptive innovation that promises to change the automotive industry as we know it". Roland Berger Strategy Consultants GmbH. 2014.
10. del Valle Hernández, Luis. "#110 Coches autónomos, el estado del arte con Alex Barredo". 2017. <https://programarfacil.com/podcast/coche-autonomo-estado-del-arte/> [Último acceso: abril 2017].
11. "Inteligencia Artificial". En línea. [https://es.wikipedia.org/wiki/Inteligencia\\_artificial](https://es.wikipedia.org/wiki/Inteligencia_artificial) [Último acceso: abril 2017].
12. Béjar Alonso, Javier. "Introducción a la Inteigencia Artificial". Asignatura de Ciencias de la Computación. Universidad Politécnica de Cataluña (UPC). Curso 2011/2012.
13. Serrano, Carlos y Gutiérrez, Begoña. "Ramas de la Inteligencia Artificial". Universidad de Zaragoza. Asignatura de Introducción a las Finanzas, Lección: Sistemas Informativos Contables, capítulo: Inteligencia Artificial.
14. Pacheco, Alberto. "Ramas de la Inteligencia Artificial". Marzo 1999. Instituto Tecnológico de Chihuahua.

15. Tolmos Rodríguez-Piñero, Piedad. *"Introducción a los algoritmos genéticos y sus aplicaciones"*. Universidad de Valencia. X Edición de las jornadas ASEPUA. Septiembre 2002.
16. Delgado Hernández, Miguel Ángel. Trabajo de Fin de Grado *"Visión artificial aplicada a la robótica"*. Julio 2016. Capítulo 2. Conceptos.
17. Maldonado Valles, Oscar. *"Visión por computadora"*. Instituto Tecnológico de Chihuahua. Mayo 2002. <http://www.depi.itch.edu.mx/apache/htm/ai11/vision.html> [Último acceso: mayo 2017].
18. Gonzalez Jimenez, Javier. *"Visión Por Computador"*. (Paraninfo. 2000 - ISBN: 84-283-2630-4)
19. de la Fuente, Eusebio. *"Diseño de los componentes del sistema de inspección"*. Capítulo 10. 2013. (ISBN: 978-84-8448-730-2)
20. Molleda Meré, Julio. Tesis doctoral *"Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados"*. Universidad de Oviedo (Departamento de Informática). Julio 2008.
21. Web oficial del fabricante Arduino. En línea. <https://www.arduino.cc> [Último acceso: Junio 2017]
22. Web oficial del fabricante BeagleBone. Características técnicas del microcontrolador BeagleBone Black. En línea. <https://beagleboard.org/black> [Último acceso: junio 2017].
23. Web oficial del fabricante Nvidia. Características técnicas del microcontrolador Jetson TX2. En línea. <http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html> [Último acceso: junio 2017].
24. Web Corporativa de Waymo (Alphabet Inc.). *"On the Road. We drive every day on public roads so we can build a safer driver. We've now launched the early rider program, a public trial of our self-driving cars in Phoenix, AZ, and we also continue to test every day in three other US locations"*. Diciembre 2016. <https://waymo.com/ontheroad/> [Último acceso: mayo 2017].
25. Ibáñez. *"Cómo funciona el coche autónomo de Google"*. Motorpasión futuro. Mayo 2012. <https://www.motorpasionfuturo.com/coches-del-futuro/como-funciona-el-coche-autonomo-de-google> [Último acceso: mayo 2017].
26. Castromil, Juan. *"Olli, el super minibús que se conduce solo ya es realidad... y es de IBM"*. 20 minutos. Junio 2016. <http://clipset.20minutos.es/olli-el-super-minibus-que-se-conduce-solo-ya-es-realidad-y-es-de-ibm/> [Último acceso: mayo 2017].
27. Contreras, Manu. *"WePod, los primeros autobuses eléctricos sin conductor llegan a Holanda"*. 20 minutos. Enero 2016. <http://clipset.20minutos.es/wepod-autobus-electrico-sin-conductor/> [Último acceso: mayo 2017].

28. Carballo, Juan. "WEpod, el primer autobús eléctrico y autónomo en circulación". *Computer hoy*. Enero 2016  
<http://computerhoy.com/noticias/hardware/wepod-primer-autobus-electrico-autonomo-circulacion-39795> [Último acceso: mayo 2017].
29. Puerto, Kote. "El autobús autónomo holandés está dando sus primeros paseos: WEPod". *Xataka*. Febrero 2016.  
<https://www.xataka.com/vehiculos/el-autobus-autonomo-holandes-esta-dando-sus-primeros-paseos-wepod> [Último acceso: mayo 2017].
30. Balcells, Marc. "Harry, un autobús público y autónomo por las calles de Londres". *EdisoNews*. Abril 2017.  
<https://www.edisonews.com/harry-autobus-autonomo-londres/> [Último acceso: mayo 2017].
31. Web oficial del fabricante Tesla. Inc. En línea. [www.tesla.com](http://www.tesla.com) [Último acceso: mayo 2017].
32. Cancela, Carlos. "Mercedes lo ha conseguido. Conducimos el primer coche autónomo de la historia." *El Confidencial*. 22 de marzo 2016.
33. Ventura Royo, Carles. "El futuro de la conducción autónoma". Blog "Informática ++", Universidad Oberta de Catalunya. Julio, 2016.  
<http://informatica.blogs.uoc.edu/2016/07/14/el-futuro-de-la-conduccion-autonoma/> [Último acceso: mayo 2017]
34. Platis, Dimitris. "The world's first Android autonomous vehicle". Blog Platis solutions. En línea. <https://platis.solutions/blog/2015/06/29/worlds-first-android-autonomous-vehicle/> [Último acceso: mayo 2017]
35. Viso, Esteban. "¿Qué es la detección de las señales de tráfico?". Enero 2013.  
<http://www.circulaseguro.com/que-es-la-deteccion-de-las-señales-de-trafico/> [Último acceso: mayo 2017]
36. Camós, Josep. "¿Qué es el LDW, o detector de cambio de carril?". Febrero 2013.  
<http://www.circulaseguro.com/que-es-el-ldw-o-detector-de-cambio-de-carril/> [Último acceso: mayo 2017]
37. Datasheet Raspberry Pi Camera Module.
38. "OpenCV". En línea <https://es.wikipedia.org/wiki/OpenCV>. [Último acceso: abril 2017]
39. del Valle Hernández, Luis. "Detector de bordes Canny cómo contar objetos con OpenCV y Python". Blog Visión Artificial. En línea. <https://programarfácil.com/blog/vision-artificial/detector-de-bordes-Canny-opencv/> [Último acceso: abril 2017].
40. Valverde Rebaza, Jorge. "Detección de bordes mediante el algoritmo de Canny". Universidad Nacional de Trujillo.

41. López, Joseph. "*Transformada Hough*". Blog Procesamiento Digital de Imágenes, (UPB). Noviembre 2012.  
<https://proceamientodigitalimagenes.wordpress.com/2012/11/02/transformada-hough/> [Último acceso: abril 2017]
42. Web OpenCV. "*Canny Edge Detector*". Diciembre 2015.  
[http://docs.opencv.org/3.1.0/da/d5c/tutorial\\_Canny\\_detector.html](http://docs.opencv.org/3.1.0/da/d5c/tutorial_Canny_detector.html) [Último acceso: abril 2017]
43. Pillath, Susanne. "Automated vehicles in the EU". EPRS (European Parliamentary Research Service). Apartado: "*Regulatory and legal framework for automated vehicles*". Enero 2016.
44. Rentería Tazo, Ainara (Abogada Snior de Gómez-Acebo & Pombo). "*¿España preparada para el arranque del coche autónomo?*". Análisis GA&P. Julio 2016.
45. Tribuna del BDS (Boletín Diario de Seguros). 7 de marzo 2017.
46. Delvaux, Many. "*Proyecto de Informe con recomendaciones destinadas a la Comisión sobre normas de Derecho civil sobre robótica*" Parlamento Europeo. Mayo 2016.
47. "*ISO 26262*". En Linea. <https://www.iso.org/search/x/query/iso%252026262> [Último acceso: junio 2017].
48. Ley 27/2014 de 27 de Noviembre, del Impuesto sobre Sociedades. Boletín Oficial del Estado (BOE). 1 de enero 2015.
49. Tienda de Robótica: Robotshop. "*Hokuyo UST-05LN Scanning Laser Obstacle Detection*". En línea. <http://www.robotshop.com/en/hokuyo-ust-05ln-scanning-laser-obstacle-detection.html> [Último acceso: junio 2017]
50. Tienda de robótica: Brickhouse Security. "*LiveWire Micro GPS Vehicle Tracker*". En línea. <http://www.brickhousesecurity.com/product/micro+gps+tracker.do?sortBy=bestSellers&from=fn> [Último acceso: junio 2017]
51. Tienda de robótica: arrow. "*Jetson TX2*". En línea.  
<https://www.arrow.com/en/products/900-83310-0001-000/nvidia> [Último acceso: junio 2017]
52. Tinda de robótica: e-con Systems. "*Jetson TX1 Camera Board*". En línea. <https://www.e-consystems.com/13mp-nvidia-jetson-tx1-camera-board.asp> [Último acceso: junio 2017]

## ANEXOS A LA MEMORIA

### ANEXO I: Código de programación del proyecto

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 16:22:54 2017

@author: samuel
"""

#importamos las librerias con las que vayamos a trabajar
import RPi.GPIO as GPIO
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import cv2
import time
import math

# Inicializamos la cámara con resolución 640x480
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))

# Tiempo de espera para que la cámara arranque
time.sleep(2)

GPIO.setmode(GPIO.BCM)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)

GPIO.setup(27, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(6, GPIO.OUT)

def delante():
    GPIO.output(22,GPIO.HIGH)
    GPIO.output(23,GPIO.HIGH)
    GPIO.output(26,GPIO.HIGH)
    GPIO.output(5,GPIO.HIGH)
    time.sleep(1)
```

```
GPIO.output(22,GPIO.LOW)
GPIO.output(23,GPIO.LOW)
GPIO.output(26,GPIO.LOW)
GPIO.output(5,GPIO.LOW)
time.sleep(0.1)

def izquierda():
    GPIO.output(27,GPIO.HIGH)
    GPIO.output(23,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(5,GPIO.HIGH)
    time.sleep(0.6)
    GPIO.output(27,GPIO.LOW)
    GPIO.output(23,GPIO.LOW)
    GPIO.output(16,GPIO.LOW)
    GPIO.output(5,GPIO.LOW)
    time.sleep(0.2)

def derecha():
    GPIO.output(22,GPIO.HIGH)
    GPIO.output(24,GPIO.HIGH)
    GPIO.output(26,GPIO.HIGH)
    GPIO.output(6,GPIO.HIGH)
    time.sleep(0.6)
    GPIO.output(22,GPIO.LOW)
    GPIO.output(24,GPIO.LOW)
    GPIO.output(26,GPIO.LOW)
    GPIO.output(6,GPIO.LOW)
    time.sleep(1)

nContadorError = 0

def region_of_interest(img,vertices):
    mask = np.zeros_like(img)
    if len(img.shape) > 2:
        channel_count = img.shape[2]
        ignore_mask_color = (255,) * channel_count
    else:
        ignore_mask_color = 255

    cv2.fillPoly(mask, vertices, ignore_mask_color)
    masked_image = cv2.bitwise_and(img,mask)
    return masked_image

def draw_lines(img_without,img):
    global nContadorError
    linesP = cv2.HoughLines(img_without,1,np.pi/180,120)
```

```

nContadorRecta = 0
nContadorDcha = 0
nContadorIzq = 0
try:
    for rho,theta in linesP[0]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*(rho)
        y0 = b*(rho)
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        #izquierda
        if math.degrees(theta)>105 and
math.degrees(theta)<115:
            if y1<600 and y2<680:
                nContadorIzq = nContadorIzq + 1
                nContadorError = 0
            else:
                #recto
                if ((math.degrees(theta)>50 and
math.degrees(theta)<75) and rho>500) or ((math.degrees(theta)<140 and
math.degrees(theta)>115) and rho<200):
                    nContadorRecta = nContadorRecta + 1
                else:
                    if (math.degrees(theta)>78 and
math.degrees(theta)<83) and rho<750:
                        nContadorDcha = nContadorDcha + 1

except TypeError, UnboundLocalError:
    print 'Error, no se econtraron lineas'

print (nContadorRecta, nContadorDcha, nContadorIzq)
if (nContadorRecta >= nContadorDcha):
    if (nContadorRecta >= nContadorIzq):
        delante()
    else:
        izquierda()
else:
    if (nContadorDcha >= nContadorIzq):
        derecha()
    else:
        izquierda()

if (nContadorRecta ==0) and (nContadorIzq ==0) and (nContadorDcha == 0):

```

```
nContadorError = nContadorError + 1
    print nContadorError

return img

def filters(img):
    img = cv2.erode
(img, cv2.getStructuringElement(cv2.MORPH_RECT,(3,3)),iterations = 1)
    img = cv2.dilate
(img, cv2.getStructuringElement(cv2.MORPH_RECT,(5,5)),iterations = 1)
    return img

#Define la region de interest que deseamos en este caso un trapecio
bot_left = [0,422]
bot_right = [633,422]
appex_left = [0,266]
appex_right = [633,266]
v = [np.array([bot_left,bot_right,appex_right,appex_left]),
dtype=np.int32]

# Define el rango de blanco en HSV
lower_blanco = np.array([5,5,160])
upper_blanco = np.array([179,85,235])
#upper_blanco = np.array([179,85,255])

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    global nContadorError

    # Leemos de lo que estamos capturando
    frame = frame.array

    #Selección region de interes
    frame = region_of_interest(frame,v)

    # Convierte BGR a HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Limpa la imagen de imperfecciones con los filtros erode y
dilate
    hsv = filters (hsv)
```

```

#Difumina la mascara para suavizar los contornos y aplicamos
filtro canny
    hsv = cv2.GaussianBlur(hsv, (5, 5), 0)

# Filtro color blanco
    hsv = cv2.inRange(hsv, lower_blanco, upper_blanco)

#Saca el contorno con la funcion Canny
    prueba = cv2.Canny(hsv,100,300,apertureSize = 3)

#Funcion para dibujar las lineas rectas en nuestra imagen
'frame'
    frame = draw_lines(prueba,frame)

# Reseteamos el archivo raw para la siguiente captura
    rawCapture.truncate(0)

if nContadorError == 100:
    cv2.DestroyAllWindows()
    # Limpiar las salidas GPIO para parar los motores
    GPIO.cleanup()
    break

tecla = cv2.waitKey(33) & 0xFF
if tecla!=255:
    if tecla==27:
        cv2.DestroyAllWindows()
        GPIO.cleanup()
        break
    elif tecla==13:
        time.sleep(10)
        print 'pausado'
    elif tecla==-1:
        continue
    else:
        print ('Debe pulsar ESC para salir no: ',tecla)

GPIO.cleanup()
cv2.DestroyAllWindows()

```

## ANEXO II: Catálogos de datos

Se presentan en documento aparte.