



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA


ETSI Aeroespacial y Diseño Industrial

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial
y Diseño Industrial

Diseño, implementación y control de un prototipo de
SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Contenido

Documento I: MEMORIA.....	4
Anexo I: PROGRAMA DEL CONTROL REMOTO EN ARDUINO	118
Anexo II: PROGRAMA DEL SEGUIMIENTO DE TRAYECTORIAS EN ARDUINO	123
Documento II: PLANOS.....	131
Documento III: PLIEGO DE CONDICIONES	140
Documento IV: PRESUPUESTO	146

Resumen

Este trabajo consiste en el diseño, implementación y control de un prototipo de robot móvil con forma esférica, al que se ha dado el nombre de *SpheroBot*. También forma parte del proyecto la simulación del sistema mecánico como paso previo a su implementación real. Por lo que al software respecta, los algoritmos desarrollados buscan, por un lado, el control remoto del robot y por otro, el seguimiento de trayectorias previamente definidas. En este proyecto, tras realizar un estudio del estado del arte, se selecciona un mecanismo para inducir el movimiento de la esfera y se desarrolla un diseño propio en SolidWorks de las piezas mecánicas que conforman el sistema. A continuación, se implementa una simulación del robot en el entorno que ofrece la herramienta *Simscape Multibody* de MATLAB, en el que se importan los modelos CAD (*Computer Aided Design*) de las piezas diseñadas. Una vez verificado el correcto funcionamiento del sistema mecánico en la simulación se procede al ensamblaje de las piezas y a la conexión de los componentes electrónicos. Finalmente, se programa en Arduino un algoritmo para el control remoto de la esfera mediante una aplicación Android y otro para el seguimiento de trayectorias. Las pruebas realizadas verifican el funcionamiento del prototipo al poder ser controlado de forma remota y ser capaz de seguir diferentes trayectorias de forma satisfactoria.

Palabras clave: *SpheroBot*, esfera, robot móvil, Arduino, control remoto, seguimiento de trayectorias.

Resum

Este treball consistix en el disseny, implementació i control d'un prototip de robot mòbil amb forma esfèrica, al qual s'ha donat el nom de *SpheroBot*. També forma part del projecte la simulació del sistema mecànic com pas previ a la seua implementació real. Pel que respecta al software, els algoritmes desenvolupats busquen d'una banda el control remot del robot i d'una altra, el seguiment de trajectòries prèviament definides. En este projecte, després de realitzar un estudi de l'estat de l'art, es selecciona un mecanisme per a induir el moviment de l'esfera i es desenvolupa un disseny propi en SolidWorks de les peces mecàniques que conformen el sistema. A continuació, s'implementa una simulació del robot en l'entorn que oferix la ferramenta *Simscape Multibody* de MATLAB, en el qual s'importen els models CAD (*Computer Aided Design*) de les peces dissenyades. Una vegada verificat el correcte funcionament del sistema mecànic en la simulació es procedix a l'acoblament de les peces i a la connexió dels components electrònics. Finalment, es programa en Arduino un algoritme per al control remot de l'esfera mitjançant una aplicació Android i un altre per al seguiment de trajectòries. Les proves realitzades verifiquen el funcionament del prototip, ja que pot ser controlat de manera remota i és capaç de seguir diferents trajectòries de manera satisfactòria.

Paraules clau: *SpheroBot*, esfera, robot mòbil, Arduino, control remot, seguiment de trajectòries.

Abstract

This project consists of the design, implementation and control of a prototype mobile robot with a spherical shape, which has been given the name SpheroBot. It is also part of the project the simulation of the mechanical system as a previous step to its real implementation. As far as software is concerned, the developed algorithms allow, on the one hand, the remote control of the robot and, on the other, the tracking of previously defined trajectories. In this project, after carrying out a study of the state of art, a mechanism is selected to induce the movement of the sphere and all the mechanical parts that make up the system are designed in SolidWorks. Next, a simulation of the robot is implemented in the environment offered by the MATLAB Simscape Multibody tool, in which the CAD (Computer Aided Design) models of the designed parts are imported. Once the correct functioning of the mechanical system has been verified in the simulation, the parts are assembled, and the electronic components are connected. Finally, an algorithm for remote control of the sphere using an Android application and another for trajectory tracking are programmed in Arduino. The tests carried out verify the operation of the prototype as it can be controlled remotely and is able to follow different trajectories properly.

Key words: SpheroBot, sphere, mobile robot, Arduino, remote control, trajectory tracking.



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

Documento I:

MEMORIA

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Índice

1.	Introducción.....	14
2.	Objetivos.....	15
3.	Marco teórico	16
3.1.	Robótica móvil	16
3.2.	Esferas robóticas.....	20
3.3.	Tipos de esferas según el mecanismo de locomoción	21
3.3.1.	Rueda motriz única y resorte	21
3.3.2.	Vehículo interno	21
3.3.3.	Dos hemisferios independientes	22
3.3.4.	Péndulo.....	23
3.3.5.	Reubicación del centro de gravedad mediante masas móviles	23
3.4.	Justificación del mecanismo elegido	25
3.5.	Cinemática	26
3.6.	Control automático	31
3.6.1.	Configuración de un sistema de control.....	31
3.6.2.	Función de transferencia	31
3.6.3.	Respuesta del sistema.....	33
3.6.4.	Control PID.....	34
3.7.	Seguimiento de trayectorias	37
4.	Diseño y selección de piezas mecánicas	38
4.1.	Esfera	38
4.2.	Chasis del vehículo	38
4.3.	Parte inferior del vehículo	41
4.4.	Motor y encoder	42
4.5.	Soporte para motor	42
4.6.	Rueda esférica	42
4.7.	Soporte para rueda esférica	43
4.8.	Sujeción para soporte de rueda esférica.....	43
4.9.	Resorte	44
4.10.	Rueda	44
4.11.	Ensamblaje	44
5.	Simulación	47
5.1.	<i>Simscape Multibody</i>	47
5.2.	Librería <i>Contact Forces</i>	50

5.3.	Modelo en Simscape Multibody	51
5.3.1.	Modelo del robot	51
5.3.2.	Reguladores de los motores	58
5.3.3.	Implementación de la simulación a partir de velocidades de referencia..	59
5.3.4.	Implementación de la simulación del desplazamiento a un punto	60
5.3.5.	Verificación de la simulación del desplazamiento a un punto	63
5.3.6.	Implementación de la simulación del seguimiento de trayectorias	66
5.4.	Resultados de la simulación	69
6.	Implementación real.....	80
6.1.	Componentes electrónicos	80
6.1.1.	Motores con encoder	80
6.1.2.	Driver L298N.....	80
6.1.3.	Batería de 12 V.....	81
6.1.4.	Microcontrolador	81
6.1.5.	Módulo Bluetooth	81
6.1.6.	Módulo MPU-9250	82
6.2.	Montaje	83
6.2.1.	Conexiones	83
6.2.2.	Ensamblaje mecánico	85
6.3.	Programación del control remoto	87
6.4.	Aplicación Android para control remoto	94
6.5.	Programación del seguimiento de trayectorias	96
6.6.	Aplicación Android para seguimiento de trayectorias	102
7.	Resultados	104
7.1.	Control PI de Velocidad	104
7.2.	Control remoto.....	105
7.3.	Seguimiento de trayectorias	107
8.	Relación con los Objetivos de Desarrollo Sostenible	111
9.	Conclusiones y trabajo futuro.....	113
10.	Bibliografía.....	116

Índice de Figuras

Figura 1. Brazo robótico industrial.....	16
Figura 2. Robot con seis ruedas.	17
Figura 3. Robot con cuatro ruedas omnidireccionales [2].....	17
Figura 4. Robot cuadrúpedo.	17
Figura 5. Robot bípedo humanoide.	18
Figura 6. Robot basado en rueda de oruga [3].....	18
Figura 7. Robot con ruedas de oruga.....	18
Figura 8. Robot tipo serpiente [4].	18
Figura 9. Robot aéreo.	19
Figura 10. Robot submarino.....	19
Figura 11. Mecanismo con rueda motriz única y resorte [7].	21
Figura 12. Mecanismo basado en vehículo interno [5].....	21
Figura 13. Prototipo propuesto por Bicchi et al. [8].....	22
Figura 14. Prototipo propuesto por Alves et al. [5].....	22
Figura 15. Dos hemisferios del prototipo planteado por Bhattacharya [9].	22
Figura 16. Esfera con mecanismo basado en dos hemisferios independientes [6].	23
Figura 17. Mecanismo basado en péndulo [6].	23
Figura 18. Prototipo propuesto por Mukherjee et al. basado en la distribución de masas [10].	24
Figura 19. Prototipo propuesto por Mojabi basado en la distribución de masas [11].	24
Figura 20. Prototipo propuesto por Lux basado en la distribución de masas [12].....	24
Figura 21. Punto en la esfera referenciado al sistema de referencia móvil y al sistema de referencia fijo [13].	26
Figura 22. Configuración del robot [13].	27
Figura 23. Vista lateral de las ruedas [13].	28
Figura 24. Esquema básico de un sistema de control [14].	31
Figura 25. Diagrama de bloques del motor de corriente continua.....	32
Figura 26. Diagrama de bloques de un sistema de control en bucle cerrado [14].....	32
Figura 27. Diagrama de bloques de un sistema de control con regulador PID [14].	35
Figura 28. Esfera divisible en dos semiesferas seleccionada.....	38
Figura 29. Vista isométrica del chasis del vehículo diseñado en SolidWorks.....	39
Figura 30. Hueco para albergar los motores de las ruedas en el chasis diseñado en SolidWorks.....	39
Figura 31. Vista inferior del chasis diseñado en SolidWorks.	40
Figura 32. Hendidura para las ruedas esféricas en el chasis diseñado en SolidWorks.	40
Figura 33. Vista superior del chasis diseñado en SolidWorks.	41
Figura 34. Vista isométrica de la parte inferior del vehículo diseñada en SolidWorks.....	41
Figura 35. Vista inferior de la parte inferior del vehículo diseñada en SolidWorks.	41
Figura 36. Modelo CAD del motor EMG30 con encoder.	42
Figura 37. Soporte para motor.....	42
Figura 38. Ruedas esféricas.....	42
Figura 39. Pieza para sujetar una rueda esférica diseñada en SolidWorks.....	43
Figura 40. Parte trasera de la pieza para sujetar una rueda esférica diseñada en SolidWorks.....	43
Figura 41. Pieza para sujetar el soporte de la rueda esférica diseñada en SolidWorks.	43
Figura 42. Resorte para garantizar el contacto entre la esfera y las ruedas esféricas.	44

Figura 43. Rueda seleccionada.....	44
Figura 44. Ensamblaje del robot completo.	45
Figura 45. Motor montado en su soporte.	45
Figura 46. Motores con ruedas en sus soportes atornillados al chasis.....	45
Figura 47. Soporte de rueda esférica montado en el chasis.....	46
Figura 48. Parte trasera de soporte de rueda esférica montado en el chasis.	46
Figura 49. Ensamblaje del vehículo sin la esfera.	46
Figura 50. Bloque Solver Configuration de Simscape Multibody.	47
Figura 51. Bloque World Frame de Simscape Multibody.	47
Figura 52. Bloque Mechanism Configuration en Simscape Multibody.	47
Figura 53. Bloque Rigid Transform en Simscape Multibody.	48
Figura 54. Bloques Spherical Solid, Cylindrical Solid y Brick Solid en Simscape Multibody.	48
Figura 55. Bloque File Solid en Simscape Multibody.....	48
Figura 56. Articulación de revolución en Simscape Multibody.	48
Figura 57. Articulación prismática en Simscape Multibody.	48
Figura 58. Articulación esférica en Simscape Multibody.....	48
Figura 59. Articulación planar en Simscape Multibody.	49
Figura 60. Bloque Transform Sensor en Simscape Multibody.....	49
Figura 61. Conector PS-Simulink Converter en Simscape Multibody.....	49
Figura 62. Conector Simulink-PS Converter en Simscape Multibody.....	49
Figura 63. Bloque Spatial Contact Force en Simscape Multibody.	49
Figura 64. Bloques de fuerzas de contacto para cuerpos 2D en la librería Contact Forces Library.	50
Figura 65. Bloques de fuerzas de contacto para cuerpos 3D en la librería Contact Forces Library.	50
Figura 66. Bloque Sphere in Sphere Force de la librería Contact Forces Library.	50
Figura 67. Esquema en Simulink del modelo del robot.....	51
Figura 68. Detalle de los bloques Solver Configuration, World Frame y Mechanism Configuration en el esquema del modelo del robot en Simulink.	52
Figura 69. Detalle de los bloques correspondientes al suelo y a la esfera en el esquema del modelo del robot en Simulink.	52
Figura 70. CAD de la esfera importado en Simscape Multibody.	53
Figura 71. Detalle de la unión entre la esfera y las ruedas en el modelo del robot en Simulink.	53
Figura 72. CAD de la rueda importado en Simscape Multibody.....	54
Figura 73. Articulaciones de revolución de las ruedas y reguladores de los motores en el modelo del robot en Simulink.....	54
Figura 74. Filtro paso-bajo en la medida de la velocidad de las ruedas en el modelo del robot en Simulink.....	55
Figura 75. Medida de la velocidad de una rueda sin filtrar en el modelo del robot en Simulink.	55
Figura 76. Medida de la velocidad de una rueda filtrada en el modelo del robot en Simulink.	55
Figura 77. Detalle de la conexión entre las articulaciones de las ruedas con los motores y de estos con el vehículo en el modelo del robot en Simulink.....	56
Figura 78. CAD del motor importado en Simscape Multibody.	56
Figura 79. CAD del soporte del motor importado en Simscape Multibody.	57

Figura 80. Detalle de la conexión del vehículo a la esfera y de las medidas de la posición y orientación del robot en el modelo del robot en Simulink.	57
Figura 81. CAD del vehículo importado en Simscape Multibody.	58
Figura 82. Contenido del subsistema de medida en el modelo del robot en Simulink.	58
Figura 83. Regulador PI de los motores en el modelo del robot en Simulink.	59
Figura 84. Seguimiento de la referencia de velocidad con el regulador diseñado en la simulación.	59
Figura 85. Simulación del robot a partir de las referencias de velocidad.	60
Figura 86. Esquema en Simulink para el desplazamiento del robot a un punto objetivo. ...	61
Figura 87. Contenido del bloque "Distancia y ángulo" del esquema de Simulink para el desplazamiento del robot a un punto objetivo.	61
Figura 88. Contenido del bloque "Velocidades de referencia" del esquema de Simulink para el desplazamiento del robot a un punto objetivo.	62
Figura 89. Contenido del subsistema "Límites ángulo".	62
Figura 90. Cilindro añadido al subsistema "Robot" para marcar el punto objetivo en Simulink.	63
Figura 91. Captura de la visualización de la simulación del desplazamiento a un punto en Simscape Multibody.	63
Figura 92. Esquema en Simulink para el seguimiento de trayectorias.	66
Figura 93. Contenido del subsistema "Posiciones de referencia" del esquema en Simulink para el seguimiento de trayectorias.	67
Figura 94. Contenido del subsistema "Contador" dentro del subsistema "Posiciones de referencia" del esquema en Simulink para el seguimiento de trayectorias.	68
Figura 95. Configuración del entorno de simulación de Simscape Multibody.	68
Figura 96. Motor Pololu 37D con encoder.	80
Figura 97. Puente en H L298N.	80
Figura 98. Batería recargable de 12 V.	81
Figura 99. Placa Arduino MEGA.	81
Figura 100. Módulo Bluetooth HC-06.	82
Figura 101. Módulo MPU-9250.	82
Figura 102. Pines del módulo L298N.	83
Figura 103. Conector del motor Pololu 37D.	84
Figura 104. Diagrama de conexiones.	84
Figura 105. Detalle del ensamblaje de la rueda y el motor al vehículo.	85
Figura 106. Detalle de la sujeción de la rueda esférica.	85
Figura 107. Detalle de la sujeción de la rueda esférica vista desde arriba.	86
Figura 108. Prototipo de SpheroBot montado.	86
Figura 109. Flujograma del programa para el control remoto del robot (Parte 1).	89
Figura 110. Flujograma del programa para el control remoto del robot (Parte 2).	90
Figura 111. Flujograma del programa para el control remoto del robot (Parte 3).	91
Figura 112. Flujograma de la función Límites_ref del programa para el control remoto del robot.	92
Figura 113. Flujograma de las interrupciones del programa para el control remoto del robot.	93
Figura 114. Interfaz de usuario de la aplicación de control remoto creada en Bluetooth Electronics.	94
Figura 115. Programación en Bluetooth Electronics de un pulsador para enviar un carácter al ser pulsado.	94

Figura 116. Programación en Bluetooth Electronics de un display para mostrar el valor de la referencia de la rueda izquierda.	95
Figura 117. Flujograma del programa para el seguimiento de trayectorias (Parte 1).	98
Figura 118. Flujograma del programa para el seguimiento de trayectorias (Parte 2).	99
Figura 119. Flujograma del programa para el seguimiento de trayectorias (Parte 3).	100
Figura 120. Flujograma del programa para el seguimiento de trayectorias (Parte 4).	101
Figura 121. Interfaz de la aplicación para el seguimiento de trayectorias creada en Bluetooth Electronics.	102
Figura 122. Configuración de gráfico en Bluetooth Electronics (Parte 1).	103
Figura 123. Configuración de gráfico en Bluetooth Electronics (Parte 2).	103
Figura 124. Configuración de gráfico en Bluetooth Electronics (Parte 3).	103
Figura 126. Seguimiento de la trayectoria circular en la aplicación Android.	108
Figura 127. Seguimiento de la trayectoria con la forma del símbolo del infinito en la aplicación Android.	109
Figura 128. Seguimiento de la trayectoria cuadrada en la aplicación Android.	109
Figura 129. Objetivos de Desarrollo Sostenible [18].	111

Índice de Ecuaciones

Ecuación 1. Posición de un punto de la esfera en el sistema de referencia fijo [13].	26
Ecuación 2. Velocidad de un punto de la esfera en el sistema de referencia fijo [13].	26
Ecuación 3. Velocidad angular en el sistema de referencia fijo [13].	26
Ecuación 4. Restricción de no deslizamiento [13].	26
Ecuación 5. Ecuación del punto de contacto a partir de la restricción de no deslizamiento [13].	27
Ecuación 6. Velocidad del punto de contacto en el sistema de referencia fijo [13].	27
Ecuación 7. Velocidad del punto central de la esfera en el sistema de referencia fijo [13].	27
Ecuación 8. Posición de los puntos de contacto entre las ruedas y la esfera en el sistema de referencia móvil [13].	28
Ecuación 9. Posición de los puntos de contacto de las ruedas en el sistema de referencia fijo [13].	28
Ecuación 10. Velocidad de los puntos de contacto de las ruedas en el sistema de referencia fijo [13].	28
Ecuación 11. Vector unitario en la dirección de ψ [13].	28
Ecuación 12. Posición de los puntos de contacto de la esfera en el sistema de referencia móvil [13].	28
Ecuación 13. Velocidad de los puntos de contacto en la esfera en el sistema de referencia fijo [13].	28
Ecuación 14. Sistema de ecuaciones resultante al igualar la velocidad de los puntos de contacto en las ruedas y en la esfera [13].	29
Ecuación 15. Sistema de ecuaciones (Ecuación 14) reformulado [13].	29
Ecuación 16. Sistema de ecuaciones resultante [13].	29
Ecuación 17. Solución del sistema de ecuaciones lineal [13].	29
Ecuación 18. Principio de conservación del momento angular [13].	29
Ecuación 19. Velocidad angular en z en función de las velocidades de las ruedas [13].	29
Ecuación 20. Derivada del ángulo ψ [13].	29
Ecuación 21. Coeficiente "c" de la ecuación X [13].	30
Ecuación 22. Ángulo ψ en función de los ángulos girados por las ruedas [13].	30
Ecuación 23. Velocidad angular del robot en función de la velocidad de las ruedas [13].	30
Ecuación 24. Velocidad del centro de la esfera en función de la velocidad y el ángulo girado de las ruedas [13].	30
Ecuación 25. Ecuaciones diferenciales del motor de corriente continua.	32
Ecuación 26. Función de transferencia del motor de corriente continua.	32
Ecuación 28. Función de transferencia de la variable controlada frente a la referencia [14].	33
Ecuación 29. Función de transferencia de la variable controlada frente a la perturbación [14].	33
Ecuación 30. Función de transferencia de la variable controlada frente al ruido en la medida [14].	33
Ecuación 31. Salida del sistema [14].	33
Ecuación 32. Error entre la referencia y la variable de control [14].	33
Ecuación 33. Ecuación característica de la función de transferencia [14].	33
Ecuación 34. Error en régimen permanente frente a entrada escalón de amplitud a .	34
Ecuación 35. Tiempo de establecimiento.	34

Ecuación 36. Sobreoscilación.	34
Ecuación 37. Acción de control del regulador PID [14].	35
Ecuación 38. Acción de control del regulador PID en función del tiempo integral y del tiempo derivativo [14].	35
Ecuación 39. Función de transferencia de un regulador PID [14].	36
Ecuación 40. Velocidades de referencia de las ruedas con el algoritmo "follow the carot".	37

Índice de Gráficos

Gráfico 1. Trayectoria seguida en el desplazamiento al punto (1,0).....	64
Gráfico 2. Trayectoria seguida en el desplazamiento al punto (0,1).....	64
Gráfico 3. Trayectoria seguida en el desplazamiento al punto (-1,0).....	65
Gráfico 4. Trayectoria seguida en el desplazamiento al punto (-1,1).....	65
Gráfico 5. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria circular.	69
Gráfico 6. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria circular.	70
Gráfico 7. Seguimiento de la referencia de la coordenada X para una trayectoria circular.	70
Gráfico 8. Seguimiento de la referencia de la coordenada Y para una trayectoria circular.	71
Gráfico 9. Seguimiento de la referencia de la orientación para una trayectoria circular. ..	71
Gráfico 10. Seguimiento de una trayectoria circular.....	72
Gráfico 11. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria con forma del símbolo del infinito.....	73
Gráfico 12. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria con forma del símbolo del infinito.....	73
Gráfico 13. Seguimiento de la referencia de la coordenada X para una trayectoria con forma del símbolo del infinito.	74
Gráfico 14. Seguimiento de la referencia de la coordenada Y para una trayectoria con forma del símbolo del infinito.	74
Gráfico 15. Seguimiento de la referencia de la orientación para una trayectoria con forma del símbolo del infinito.....	75
Gráfico 16. Seguimiento de una trayectoria con forma del símbolo del infinito.	75
Gráfico 17. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria cuadrada.....	76
Gráfico 18. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria cuadrada.....	76
Gráfico 19. Seguimiento de la referencia de la coordenada X para una trayectoria cuadrada.....	77
Gráfico 20. Seguimiento de la referencia de la coordenada Y para una trayectoria cuadrada.....	77
Gráfico 21. Seguimiento de la referencia de la orientación para una trayectoria cuadrada.	78
Gráfico 22. Seguimiento de una trayectoria cuadrada.....	78
Gráfico 23. Respuesta de la velocidad de los motores frente a un cambio en escalón de la referencia.....	104
Gráfico 24. Trayectoria circular predefinida para el seguimiento de trayectorias.	107
Gráfico 25. Trayectoria con la forma del símbolo del infinito predefinida para el seguimiento de trayectorias.	107
Gráfico 26. Trayectoria cuadrada predefinida para el seguimiento de trayectorias.	108

1. Introducción

La robótica, como campo de la ingeniería, ha experimentado en los últimos años un importante crecimiento y se espera que continúe extendiéndose en el futuro dada la elevada demanda de robots para distintas funciones, lo que requiere dispositivos de distintas características y funcionalidades según la operación a la que se desean aplicar.

Son cada vez más las diferentes tareas que se pretenden llevar a cabo por medios robóticos, y no solamente en procesos de fabricación industrial, donde se ha aplicado gran parte de la robótica tradicionalmente, sino también en otros ámbitos como el doméstico, el de la seguridad, la exploración, la distribución de mercancías o incluso como juguetes.

Ante la gran demanda de robots en nuevos ámbitos surge la necesidad de adaptar los robots clásicos o incluso de diseñar nuevos robots para llevar a cabo de forma satisfactoria operaciones que anteriormente no se realizaban con robots o que se limitaban al entorno industrial y ahora se deben llevar a cabo en otros lugares.

En este contexto, surge la idea de las esferas robóticas, entendidas como un tipo de robot móvil con forma esférica que alberga en su interior un mecanismo que permite controlar su movimiento. Con esta forma innovadora de robot se logran unas características únicas como la imposibilidad de volcar, lo que los hace idóneos para terrenos irregulares o desconocidos, o su alta maniobrabilidad.

El uso de este tipo de robots no está aún extendido, aunque ya se han diseñado y comercializado algunos modelos que utilizan diferentes sistemas para su movimiento, como son un vehículo interno, un péndulo que desplaza su centro gravedad o diferentes masas móviles que se desplazan también con el fin de cambiar la posición del centro de masas de la esfera provocando así su movimiento. Los diferentes métodos investigados hasta ahora para las esferas robóticas presentan ciertas ventajas e inconvenientes que han de estudiarse antes de decantarse por uno de ellos.

Este proyecto va dirigido al diseño, simulación, implementación real y programación de un prototipo de esfera robótica, que se ha llamado *SpheroBot*. Se pretende fabricar un prototipo funcional de un robot móvil con forma esférica que se desplace siguiendo órdenes recibidas desde un mando de control mediante una conexión inalámbrica y que sea además capaz de seguir trayectorias. Para esto último será preciso desarrollar un modelo matemático que permita estimar la posición del robot a partir del movimiento de las ruedas del vehículo interno, o bien utilizar sensores que permitan medir su posición y orientación.

En este trabajo, tras realizar un estudio teórico sobre este tipo de sistemas, se comienza con el diseño de las piezas del robot en *SolidWorks*. A continuación, se lleva a cabo una simulación en *Simscape Multibody*, que es un entorno de simulación de Matlab, para comprobar que el comportamiento del sistema es el esperado. Finalmente, se procede al montaje y la programación del robot y se realizan distintas pruebas para verificar su funcionamiento.

2. Objetivos

El objetivo de este proyecto es diseñar, implementar y controlar un prototipo de un robot móvil con forma de esfera, al que se le ha dado el nombre de *SpheroBot*. Se pretende, por tanto, construir un robot esférico con un diseño propio que pueda ser controlado remotamente por el usuario. Este fin se consigue a través de los siguientes objetivos específicos:

- Diseñar en *SolidWorks* el conjunto de piezas que conforman el robot esférico, incluyendo todo el mecanismo interno que posibilita su movimiento.
- Realizar una simulación en el entorno que ofrece *Simscape Multibody* importando el CAD diseñado para verificar el correcto comportamiento del sistema.
- Desarrollar y simular un algoritmo que permita el control remoto del robot.
- Desarrollar y simular un algoritmo que permita el seguimiento de trayectorias por parte del robot.
- Fabricar y ensamblar todas las piezas que conforman el robot, así como todos los componentes electrónicos necesarios.
- Programar del robot para su control remoto por medio de una aplicación móvil y para el seguimiento de trayectorias.
- Realizar pruebas que certifiquen el correcto funcionamiento del sistema implementado.

3. Marco teórico

3.1. Robótica móvil

En la actualidad, el estudio y la investigación en el ámbito de la robótica móvil está experimentando un importante crecimiento. Los robots móviles son capaces de sustituir a los seres humanos en múltiples tareas. Entre sus muchas aplicaciones destacan la vigilancia, la exploración planetaria, seguridad, operaciones de rescate, reconocimiento, automatización industrial, construcción, entretenimiento, guías en museos y atracciones turísticas, servicios personales, intervención en ambientes peligrosos para el hombre, transporte, cuidados médicos, etc. [1]

Algunos robots móviles necesitan ser controlados por un operario, mientras que otros muchos son autónomos y no necesitan la intervención humana para funcionar, sino que son capaces de determinar las acciones que deben ejecutar mediante la información recogida a través de sensores [1].

La robótica móvil se basa en cuatro campos de estudio: locomoción, percepción, cognición y navegación. La locomoción estudia el movimiento del robot, para lo cual se basa en la cinemática y dinámica del mecanismo. La percepción recoge información del robot y del entorno. La cognición analiza dicha información y decide en función de ella qué acciones tomar. Por último, la navegación se encarga de generar las trayectorias mediante distintos algoritmos para alcanzar la posición objetivo evitando los obstáculos que puedan existir en el entorno [1].

Existen diferentes tipos de robots según su locomoción, siendo los principales los siguientes:

Robots estacionarios

Estos robots no entran dentro de la clasificación de robots móviles, puesto que no pueden desplazarse. Normalmente son cadenas cinemáticas abiertas que parten de una base fija. Los robots manipuladores industriales, como el de la Figura 1, son el principal ejemplo de este tipo de sistemas [1].



Figura 1. Brazo robótico industrial.

Robots terrestres

Tal y como indica su nombre, se trata de robots que se desplazan sobre la superficie terrestre. Dentro de esta categoría, existen diferentes tipos de sistemas según el mecanismo que empleen para desplazarse. En primer lugar, cabe destacar los robots

basados en ruedas, cuyas ventajas son su simplicidad tanto en el diseño como en su programación, control y construcción. Además, son más baratos que los basados en piernas y es más sencillo mantener el equilibrio. Su inconveniente es la mayor dificultad para sortear obstáculos o moverse en terreno irregular o con baja fricción. La Figura 2 muestra un robot con seis ruedas. Existen además robots con ruedas omnidireccionales, como el que puede verse en la Figura 3, lo que aumenta su maniobrabilidad [1].



Figura 2. Robot con seis ruedas.

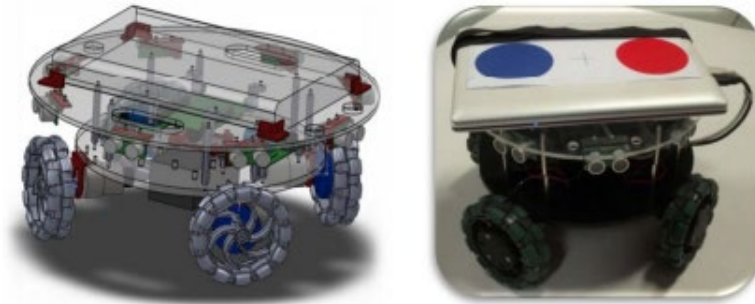


Figura 3. Robot con cuatro ruedas omnidireccionales [2].

Una alternativa a los sistemas basados en ruedas, son los robots con piernas. Son más caros que los anteriores, pero presentan ciertas ventajas como su eficiencia, versatilidad, capacidad para moverse en terreno irregular y mayor facilidad para evitar obstáculos. Sin embargo, su equilibrio y control son más complejos. Se pueden encontrar robots con distinto número de patas, predominando las estructuras con entre dos y ocho piernas [1]. La Figura 4 muestra un robot con cuatro patas. Los robots humanoides, como el de la Figura 5, entrarían también en este grupo.



Figura 4. Robot cuadrúpedo.



Figura 5. Robot bípedo humanoide.

Otro mecanismo ampliamente utilizado son las ruedas de oruga. Al existir un mayor contacto con la superficie, mejora el comportamiento en terrenos con baja fricción; aunque en otros casos resulta ineficiente. Además, al girar se produce cierto derrape de las ruedas en el suelo, lo que dificulta seguir la trayectoria con precisión en las curvas [1]. La Figura 6 y la Figura 7 muestran ejemplos de este tipo de robots.

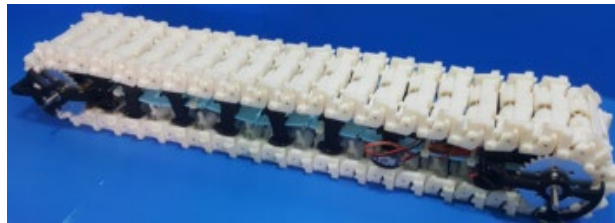


Figura 6. Robot basado en rueda de oruga [3].



Figura 7. Robot con ruedas de oruga.

También existen robots híbridos, que combinan algunos de los sistemas mencionados, así como otros tipos de robots que no encajan en ninguna de las clasificaciones expuestas, como los robots de tipo oruga que avanzan a base de contraerse y estirarse o los de tipo serpiente, como el visible en la Figura 8, que reptan sobre la superficie [1].



Figura 8. Robot tipo serpiente [4].

Robots aéreos

Comúnmente conocidos como drones se emplearon en un principio en el ámbito militar, pero actualmente se utilizan también en aplicaciones científicas, agrícolas, comerciales, de ocio, vigilancia y seguridad, distribución de productos o fotografía, entre otras muchas [1]. La Figura 9 muestra un ejemplo.



Figura 9. Robot aéreo.

Robots marinos

Se han desarrollado distintos tipos de robots para tareas de exploración submarina para áreas sumergidas inaccesibles y también en tareas de limpieza de los mares [1]. En la Figura 10 se puede apreciar un robot submarino.



Figura 10. Robot submarino.

3.2. Esferas robóticas

En el desarrollo de robots móviles para terrenos irregulares, la movilidad es uno de los aspectos fundamentales. Existen diferentes tipos de robots móviles que utilizan piernas, ruedas, ruedas de oruga o distintas combinaciones de ellas para su movimiento. Persiste, sin embargo, una necesidad de desarrollar mecanismos de locomoción más eficientes y versátiles, capaces de adaptarse a entornos irregulares [5].

Los robots móviles esféricos presentan importantes ventajas en este sentido. Para empezar, son muy maniobrables. Se pueden diseñar tanto para ser holonómicos como no holonómicos. En el primer caso, el robot puede moverse en cualquier dirección, lo cual aumenta las opciones a la hora de salvar obstáculos y evita que el robot pueda quedarse bloqueado en esquinas o frente a objetos del entorno. Por otra parte, los robots esféricos no pueden volcar, a diferencia de los robots móviles tradicionales, que pierden su capacidad de maniobra si se da esta circunstancia. Asimismo, los escalones y desniveles no son un problema para estos robots, mientras que sí lo son para los robots clásicos. Igualmente, los robots esféricos pueden dejarse caer o incluso lanzarse y se pueden mover una vez en el suelo independientemente de la posición en la que caigan. También tienen una gran capacidad para recobrase tras impactos con obstáculos, ya que no pueden quedar en posiciones irreversibles. Esto último sería de gran utilidad en aplicaciones con varios robots trabajando de forma conjunta, ya que un posible choque entre dos esferas no interferiría de forma grave en su movimiento [6].

Otra ventaja es que pueden diseñarse para estar sellados, lo que los hace ideales para trabajar en ambientes dañinos, puesto que los elementos electrónicos y mecánicos (sensores, actuadores, controladores...) pueden quedar protegidos en el interior de la carcasa esférica. Por este motivo, son capaces de trabajar en la nieve, en el barro e incluso en agua. Además, pueden ser más pequeños que otros vehículos mediante ruedas, lo que les permite tener un menor coste [6].

Entre sus principales usos en la actualidad cabe destacar aplicaciones de seguridad y defensa. Con los sensores necesarios se utilizan en detección de armas, reconocimiento, vigilancia, evaluación de riesgos, valoración de situaciones a una distancia de seguridad y detección de intrusos. Un robot esférico puede ser lanzado al interior de una habitación desde una ventana y ser controlado remotamente para recoger información. También puede ser útil para inspeccionar lugares como el interior de conductos o tuberías [6].

Otra de sus aplicaciones más populares es como juguete en la industria del entretenimiento. Pueden ser ligeros, baratos y resistentes al agua, además de animar a los niños a moverse, a interactuar con él y a introducirse en el mundo de la tecnología [6].

Por otro lado, y dadas sus ventajas para desplazarse por terreno irregular o desconocido, pueden usarse para exploración autónoma. A modo de ejemplo, en la NASA se ha propuesto la idea de emplear un robot esférico en la exploración de Marte [6].

3.3. Tipos de esferas según el mecanismo de locomoción

Las esferas robóticas deben llevar en su interior un sistema mecánico que induzca en ellas el movimiento. La solución a este problema no es única, sino que diferentes autores han propuesto distintos mecanismos que permiten provocar el movimiento de la esfera. A continuación, se explican los distintos sistemas de locomoción de esferas móviles.

3.3.1. Rueda motriz única y resorte

Este mecanismo, mostrado en la Figura 11, fue propuesto en 1996 por Halme et al. [7]. Consiste simplemente en un cuerpo central con una rueda motriz en su parte inferior en contacto con la superficie interna de la esfera, que es la responsable de inducir el movimiento. Además, en la parte superior se sitúa otra rueda, pasiva en este caso. El muelle bajo la rueda pasiva se encarga de garantizar que ambas ruedas estén permanentemente en contacto con la superficie de la esfera. La rueda inferior puede girar a derecha e izquierda, lo que permite cambiar la dirección del robot.

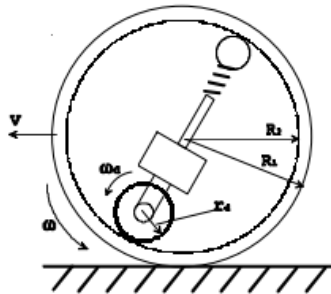


Figura 11. Mecanismo con rueda motriz única y resorte [7].

3.3.2. Vehículo interno

Otra opción para lograr el movimiento de la esfera consiste en la utilización de un vehículo interno situado dentro de la carcasa esférica, tal y como muestra la Figura 12. El primer prototipo con este mecanismo fue propuesto por Bicchi et al. [8] en 1997, y puede verse en la Figura 13. En este caso, se trata de un vehículo con dos ruedas motrices y otras dos pasivas para garantizar el contacto con la esfera. Existen otros prototipos con cuatro ruedas, como el diseñado por Alves et al. [5], que puede observarse en la Figura 14.

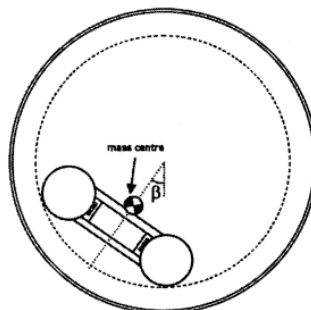


Figura 12. Mecanismo basado en vehículo interno [5].

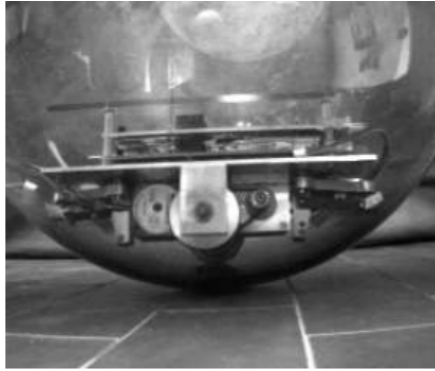


Figura 13. Prototipo propuesto por Bicchi et al. [8].



Figura 14. Prototipo propuesto por Alves et al. [5].

Este tipo de mecanismo presenta la ventaja de ser simple a nivel mecánico. Además, el sistema puede ser holónimo o no holónimo dependiendo de la configuración de las ruedas. También cabe destacar que se puede obtener con facilidad un modelo matemático del sistema, lo que facilita su control [6].

3.3.3. Dos hemisferios independientes

Este sistema fue propuesto por Bhattacharya [9] en el año 2000. La esfera está dividida en dos mitades, cada una de las cuales cuenta con un motor y un rotor. El robot se mueve gracias al principio de conservación del momento en los rotores [6] y puede girar reduciendo o incluso invirtiendo la velocidad de uno de los dos motores.

Una propiedad importante es que el centro de masas coincide con el centro geométrico, ya que ambos hemisferios tienen exactamente los mismos componentes situados en posiciones diametralmente opuestas. Por esta razón, la esfera no tiende a volcar.

En la Figura 15 y la Figura 16 se muestran ejemplos de esferas con este mecanismo.

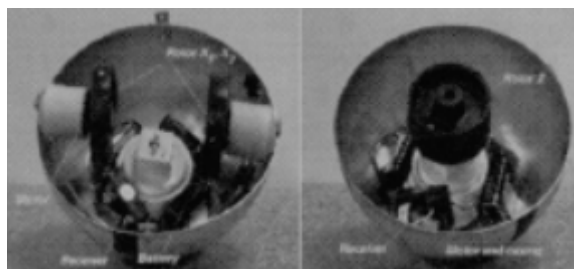


Figura 15. Dos hemisferios del prototipo planteado por Bhattacharya [9].



Figura 16. Esfera con mecanismo basado en dos hemisferios independientes [6].

3.3.4. Péndulo

En el interior de la esfera se dispone un péndulo con dos grados de libertad. El péndulo lleva en su extremo una masa que le permite, mediante su movimiento, desplazar el centro de gravedad de la esfera induciendo así su movimiento. La esfera avanza o retrocede al mover el péndulo hacia delante o hacia atrás; y gira cuando éste se mueve hacia los lados. La Figura 17 ilustra este mecanismo.

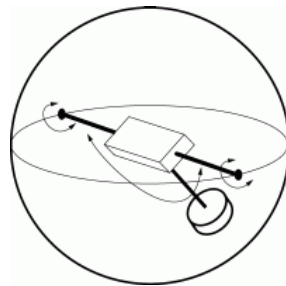


Figura 17. Mecanismo basado en péndulo [6].

3.3.5. Reubicación del centro de gravedad mediante masas móviles

Otra alternativa consiste en desplazar el centro de gravedad de la esfera redistribuyendo unas masas colocadas en su interior, lo que permite al robot moverse en cualquier dirección [6]. Un ejemplo de este mecanismo es el propuesto por Mukherjee et al. [10], mostrado en la Figura 18, en el que existe un cuerpo central en la esfera del que nacen ejes hacia la superficie de la misma, a lo largo de los cuales se desplazan las masas. Otro similar es el diseñado por Mojabi [11], que puede verse en la Figura 19. Este prototipo tiene cuatro ejes con masas móviles dispuestos en forma tetraédrica. Su ventaja es que, cuando las masas están a la misma distancia, el centro de masas coincide con el centro geométrico de la esfera. Otro modelo basado en este principio es el presentado por Lux [12], visible en la Figura 20, en el cual una masa está sujeta por cables, que pueden retraerse o extenderse para cambiar su posición.

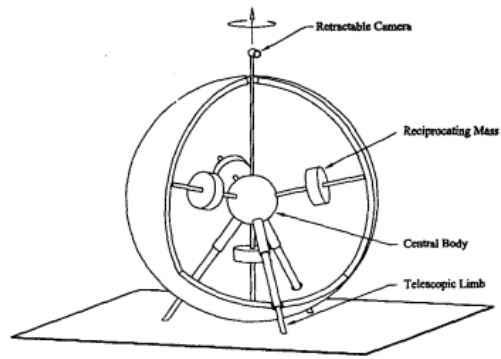


Figura 18. Prototipo propuesto por Mukherjee et al. basado en la distribución de masas [10].

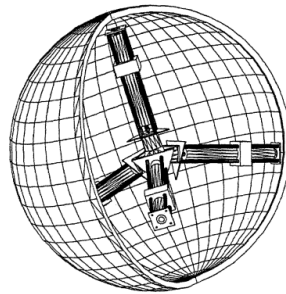


Figura 19. Prototipo propuesto por Mojabi basado en la distribución de masas [11].

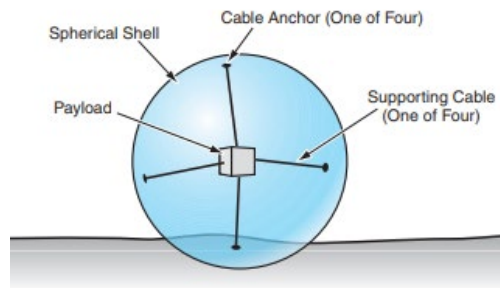


Figura 20. Prototipo propuesto por Lux basado en la distribución de masas [12].

3.4. Justificación del mecanismo elegido

Para el desarrollo de un robot esférico, en este proyecto se ha optado por utilizar un mecanismo basado en un vehículo interno de dos ruedas. Entre las principales razones que justifican esta elección cabe destacar en primer lugar la mayor sencillez constructiva del sistema, así como la menor complejidad del modelo matemático que rige su comportamiento y, por tanto, de su control. También se ha considerado su mayor capacidad para superar pendientes frente a los mecanismos basados en péndulo o en la redistribución de masas.

En lo que a la maniobrabilidad y a la capacidad para sortear obstáculos mediante cambios rápidos de dirección respecta, es preciso indicar que el mecanismo basado en un vehículo de dos ruedas representa una buena solución, ya que puede girar sobre sí mismo cambiando por tanto la dirección del robot, mientras que un vehículo de cuatro ruedas necesitaría de un cierto radio de giro; lo que también ocurriría con la opción del péndulo y de la rueda única. Con la redistribución de masas o la opción de dos hemisferios independientes se podría lograr también una alta maniobrabilidad, pero sería a costa de un mecanismo y un control mucho más complejos.

Los principales inconvenientes del mecanismo con un vehículo interno frente a las otras opciones son dos. En primer lugar, la dificultad de mantener las ruedas del vehículo en contacto continuo con la superficie interna de la esfera; y en segunda instancia, una menor eficiencia energética debido a las pérdidas por el rozamiento entre las ruedas y la superficie interior de la esfera. El primer problema puede resolverse mediante un diseño que incluya elementos que garanticen el contacto entre el vehículo y la esfera como ruedas pasivas que pueden además incluir elementos con muelles que aseguren el contacto con la esfera de forma similar al modelo con una única rueda propuesto por Halme et al. [7]. En cuanto a la segunda cuestión, es inevitable tener ciertas pérdidas por rozamiento entre las ruedas del vehículo y la esfera, que no existen en los mecanismos basados en péndulo, en dos hemisferios o en masas móviles. No obstante, en este trabajo se ha decidido sacrificar una mayor eficiencia en pro de una menor complejidad en el diseño, construcción, modelado y control del sistema, y de una alta maniobrabilidad.

3.5. Cinemática

En este apartado se exponen las ecuaciones que modelan la cinemática de una esfera robótica cuyo mecanismo de locomoción consiste en un vehículo interno de dos ruedas.

Primeramente, se ha de estudiar la cinemática de la esfera en sí. Se considera una esfera de radio r en el plano $z = 0$ cuyo centro está en el punto p_c en un sistema de referencia fijo. R es la matriz de rotación que especifica la orientación de la esfera con respecto al sistema de referencia fijo, y \bar{q} un punto cualquiera de la esfera cuyas coordenadas están expresadas con respecto a un sistema de referencia móvil situado en el centro de la esfera [13].

La posición de dicho punto en el sistema de referencia fijo es [13]:

$$q = p_c + R \cdot \bar{q}$$

Ecuación 1. Posición de un punto de la esfera en el sistema de referencia fijo [13].

La velocidad del punto q en el sistema de referencia fijo se puede expresar como:

$$\dot{q} = \begin{bmatrix} \dot{p}_c \\ 0 \end{bmatrix} + \dot{R} \cdot \bar{q} = \begin{bmatrix} \dot{p}_c \\ 0 \end{bmatrix} + \hat{\omega} \cdot R \cdot \bar{q}$$

Ecuación 2. Velocidad de un punto de la esfera en el sistema de referencia fijo [13].

donde $\hat{\omega}$ es la velocidad angular en el sistema de referencia fijo [13].

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Ecuación 3. Velocidad angular en el sistema de referencia fijo [13].

La Figura 21 ilustra el punto q tanto en el sistema de coordenadas móvil (*body frame*) como en el fijo (*spatial frame*).

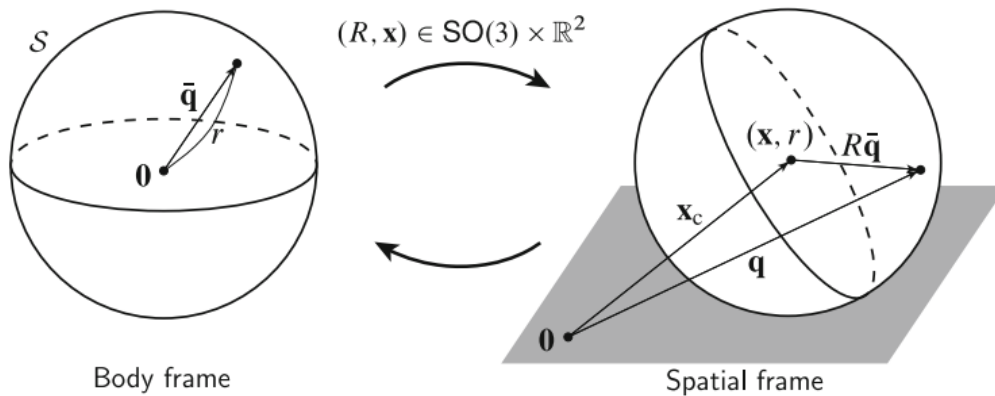


Figura 21. Punto en la esfera referenciado al sistema de referencia móvil y al sistema de referencia fijo [13].

Por otra parte, la condición de no deslizamiento en el punto de contacto entre la esfera y la superficie impone una restricción no holónoma, según la cual, el punto de contacto q_c debe satisfacer la siguiente ecuación:

$$R \cdot q_c = -r \cdot e_z$$

Ecuación 4. Restricción de no deslizamiento [13].

donde e_z es el vector unitario en la dirección del eje z .

De la ecuación anterior se deduce la que sigue:

$$q_c = -r \cdot R^T \cdot e_z$$

Ecuación 5. Ecuación del punto de contacto a partir de la restricción de no deslizamiento [13].

La velocidad del punto de contacto en el sistema de referencia fijo se puede expresar como:

$$\dot{q}_c = \begin{bmatrix} \dot{p}_c \\ 0 \end{bmatrix} - r \cdot \hat{\omega} \cdot e_z$$

Ecuación 6. Velocidad del punto de contacto en el sistema de referencia fijo [13].

Teniendo en cuenta la condición de no deslizamiento la velocidad de centro de la esfera queda:

$$\dot{p}_c = \begin{bmatrix} \dot{p}_{cx} \\ \dot{p}_{cy} \end{bmatrix} = r \cdot \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}$$

Ecuación 7. Velocidad del punto central de la esfera en el sistema de referencia fijo [13].

Se procede ahora a considerar la cinemática del robot, más concretamente la interacción entre la esfera y las ruedas del vehículo interno. Se llama ψ al ángulo de rotación entre el eje de las ruedas medido desde el sentido positivo del eje x del sistema de referencia fijo, e y_w^i (con $i = 1, 2$) a la posición en el sistema de referencia móvil de la esfera del punto de contacto entre la rueda i y la esfera, como se observa en la Figura 22, en la que $2 \cdot w$ es la distancia entre las ruedas y h es la distancia vertical entre el punto de contacto de las ruedas con la esfera y el centro de esta [13].

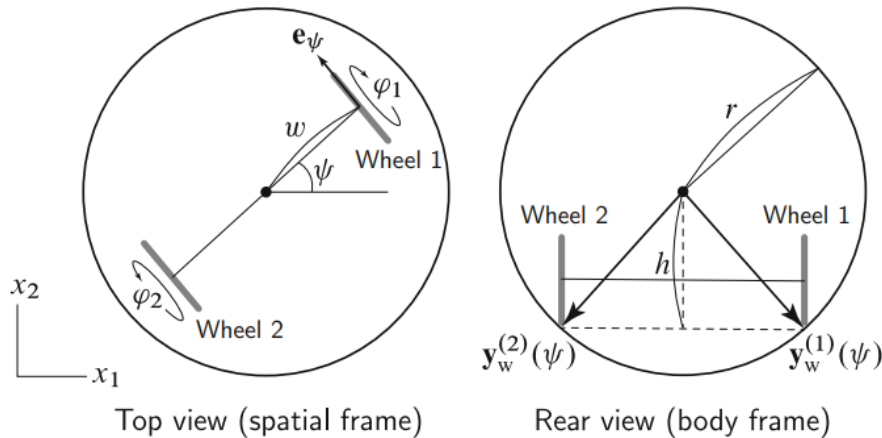


Figura 22. Configuración del robot [13].

La Figura 23 muestra la vista lateral de ambas ruedas, donde φ_i es el ángulo girado por la rueda i , ρ es el radio de las ruedas y e_ψ es el vector unitario en la dirección de ψ .

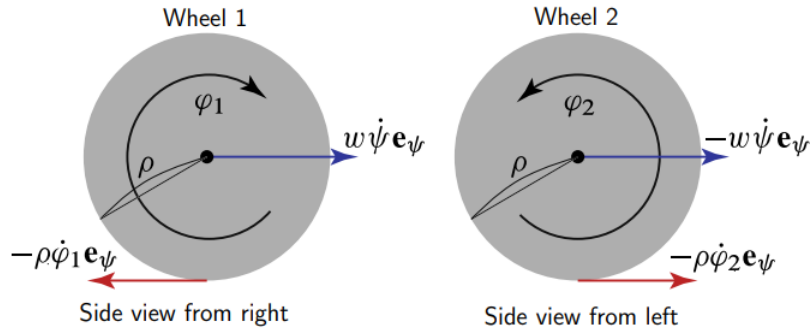


Figura 23. Vista lateral de las ruedas [13].

La posición de los puntos de contacto de las ruedas con respecto al sistema de referencia móvil situado en el centro de la esfera es:

$$y_w^1 = \begin{bmatrix} w \cdot \cos \psi \\ w \cdot \sin \psi \\ -h \end{bmatrix}; \quad y_w^2 = \begin{bmatrix} -w \cdot \cos \psi \\ -w \cdot \sin \psi \\ -h \end{bmatrix}$$

Ecuación 8. Posición de los puntos de contacto entre las ruedas y la esfera en el sistema de referencia móvil [13].

La velocidad de los puntos de contacto de cada rueda debe coincidir con la velocidad de los puntos de contacto de la esfera.

La posición de los puntos de contacto de las ruedas en el sistema de referencia fijo se calcula como sigue:

$$q_w^i = p_c + y_w^i$$

Ecuación 9. Posición de los puntos de contacto de las ruedas en el sistema de referencia fijo [13].

La velocidad de cada rueda en el sistema de coordenadas fijo es la composición de las velocidades de translación y rotación, quedando:

$$\dot{q}_w^1 = \dot{p}_c + (w \cdot \dot{\psi} - \rho \cdot \dot{\varphi}_1) \cdot e_\psi; \quad \dot{q}_w^2 = \dot{p}_c - (w \cdot \dot{\psi} + \rho \cdot \dot{\varphi}_2) \cdot e_\psi$$

Ecuación 10. Velocidad de los puntos de contacto de las ruedas en el sistema de referencia fijo [13].

donde

$$e_\psi = (-\sin \psi, \cos \psi, 0)^T$$

Ecuación 11. Vector unitario en la dirección de ψ [13].

Por otra parte, la posición de los puntos de contacto en la esfera en el sistema de referencia móvil cuyo origen está en el centro de la esfera es:

$$q_s^i = R^T \cdot y_w^i$$

Ecuación 12. Posición de los puntos de contacto de la esfera en el sistema de referencia móvil [13].

La velocidad de los puntos de contacto en la esfera, expresados en el sistema fijo son:

$$\dot{q}_s^i = \begin{bmatrix} \dot{p}_c \\ 0 \end{bmatrix} + \hat{\omega} \cdot y_w^i$$

Ecuación 13. Velocidad de los puntos de contacto en la esfera en el sistema de referencia fijo [13].

Igualando las velocidades de los puntos de contacto en las ruedas y en la esfera ($q_s^i = q_w^i$) queda:

$$\widehat{\omega} \cdot y_w^1 = (w \cdot \dot{\psi} - \rho \cdot \dot{\phi}_1) \cdot e_\psi ; \widehat{\omega} \cdot y_w^2 = -(w \cdot \dot{\psi} + \rho \cdot \dot{\phi}_2) \cdot e_\psi$$

Ecuación 14. Sistema de ecuaciones resultante al igualar la velocidad de los puntos de contacto en las ruedas y en la esfera [13].

Conociendo la siguiente propiedad:

$$\widehat{a} \cdot b = a \times b = -b \times a = -\widehat{b} \cdot a$$

puede expresarse la Ecuación 14 como:

$$-\widehat{y_w^1} \cdot \omega = (w \cdot \dot{\psi} - \rho \cdot \dot{\phi}_1) \cdot e_\psi ; -\widehat{y_w^2} \cdot \omega = -(w \cdot \dot{\psi} + \rho \cdot \dot{\phi}_2) \cdot e_\psi$$

Ecuación 15. Sistema de ecuaciones (Ecuación 14) reformulado [13].

o bien:

$$Y \cdot \omega = b \quad \text{con} \quad Y = - \begin{bmatrix} \widehat{y_w^1} \\ \widehat{y_w^2} \end{bmatrix} ; b = \begin{bmatrix} (w \cdot \dot{\psi} - \rho \cdot \dot{\phi}_1) \cdot e_\psi \\ -(w \cdot \dot{\psi} + \rho \cdot \dot{\phi}_2) \cdot e_\psi \end{bmatrix}$$

Ecuación 16. Sistema de ecuaciones resultante [13].

Resolviendo el sistema lineal de la Ecuación 15 se obtiene $\omega = (Y^T \cdot Y)^{-1} \cdot Y^T \cdot b$ o, desarrollando:

$$\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = -\frac{\rho}{2 \cdot h} \cdot (\dot{\phi}_1 + \dot{\phi}_2) \cdot \begin{bmatrix} \cos \psi \\ \sin \psi \end{bmatrix}$$

$$\omega_z = \dot{\psi} - \frac{\rho}{2 \cdot w} \cdot (\dot{\phi}_1 - \dot{\phi}_2)$$

Ecuación 17. Solución del sistema de ecuaciones lineal [13].

Por último, se añade una restricción más. El momento angular total del robot con respecto al eje vertical de la esfera se conserva. Suponiendo que inicialmente no gira, el momento angular total es nulo. Si I_s es el momento de inercia de la esfera respecto a cualquier eje que pase por su centro y J es el momento de inercia de la unidad electromecánica, se tiene:

$$I_s \cdot \omega_z + J \cdot \dot{\psi} = 0$$

Ecuación 18. Principio de conservación del momento angular [13].

Resolviendo se obtiene:

$$\omega_z = -\frac{J}{I_s} \cdot c \cdot (\dot{\phi}_1 - \dot{\phi}_2)$$

Ecuación 19. Velocidad angular en z en función de las velocidades de las ruedas [13].

con

$$\dot{\psi} = c \cdot (\dot{\phi}_1 - \dot{\phi}_2)$$

Ecuación 20. Derivada del ángulo ψ [13].

donde

$$c = \frac{\rho \cdot I_s}{2 \cdot w \cdot (I_s + J)}$$

Ecuación 21. Coeficiente "c" de la Ecuación 20 [13].

La Ecuación 20 es una restricción holonóma que puede ser integrada, quedando:

$$\psi = c \cdot (\varphi_1 - \varphi_2)$$

Ecuación 22. Ángulo ψ en función de los ángulos girados por las ruedas [13].

Con las restricciones anteriores se llega a las siguientes ecuaciones (Ecuación 23 y Ecuación 24), que definen un sistema de control cinemático:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} -\frac{\rho}{2 \cdot h} \cdot \cos(c \cdot (\varphi_1 - \varphi_2)) \\ -\frac{\rho}{2 \cdot h} \cdot \sin(c \cdot (\varphi_1 - \varphi_2)) \\ -c \cdot \frac{J}{I_s} \end{bmatrix} \cdot \dot{\varphi}_1 + \begin{bmatrix} -\frac{\rho}{2 \cdot h} \cdot \cos(c \cdot (\varphi_1 - \varphi_2)) \\ -\frac{\rho}{2 \cdot h} \cdot \sin(c \cdot (\varphi_1 - \varphi_2)) \\ c \cdot \frac{J}{I_s} \end{bmatrix} \cdot \dot{\varphi}_2$$

Ecuación 23. Velocidad angular del robot en función de la velocidad de las ruedas [13].

$$v_c = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \end{bmatrix} = \frac{r \cdot \rho}{2 \cdot h} \cdot (\dot{\varphi}_1 + \dot{\varphi}_2) \cdot \begin{bmatrix} -\sin(c \cdot (\varphi_1 - \varphi_2)) \\ \cos(c \cdot (\varphi_1 - \varphi_2)) \end{bmatrix}$$

Ecuación 24. Velocidad del centro de la esfera en función de la velocidad y el ángulo girado de las ruedas [13].

donde x_c e y_c son las coordenadas X e Y del centro de la esfera.

Si las velocidades de las ruedas son controlables, la Ecuación 23 y la Ecuación 24 definen un control cinemático no holónomo de dos entradas $\dot{\varphi}_1$ y $\dot{\varphi}_2$, que son las velocidades de giro de las dos ruedas del vehículo interno [13].

3.6. Control automático

En este apartado se describen los fundamentos teóricos del regulador que se utilizará para el control de los motores del robot.

3.6.1. Configuración de un sistema de control

Los sistemas de control pueden clasificarse en manuales y automáticos, según si requieren o no la intervención directa del ser humano. El control automático, a su vez, se puede dividir en control en lazo abierto y control en lazo cerrado [14].

Mientras que el control automático en lazo abierto se limita a enviar una acción de control al proceso a partir de una referencia dada sin conocer el estado real del sistema, el control en lazo cerrado mide la salida a controlar y compara su valor con la referencia para, a partir del error entre ambas magnitudes, enviar la acción de control precisa para corregir dicha desviación.

Se puede definir, por tanto, el control en bucle cerrado como un control que compara la variable controlada con un valor deseado (referencia) y lleva a cabo una operación correctiva para igualar el valor de la magnitud controlada al valor de consigna [14].

La Figura 24 muestra un esquema básico de control en lazo cerrado. Se compara un valor deseado (referencia) con la magnitud medida a través de un sensor. El error entre ambas es la entrada al controlador que genera una acción de control que se envía al actuador, que actúa sobre el proceso modificando la salida (variable controlada). Además, pueden existir señales de perturbación en el proceso y de ruido eléctrico en el sensor que afecten al sistema.

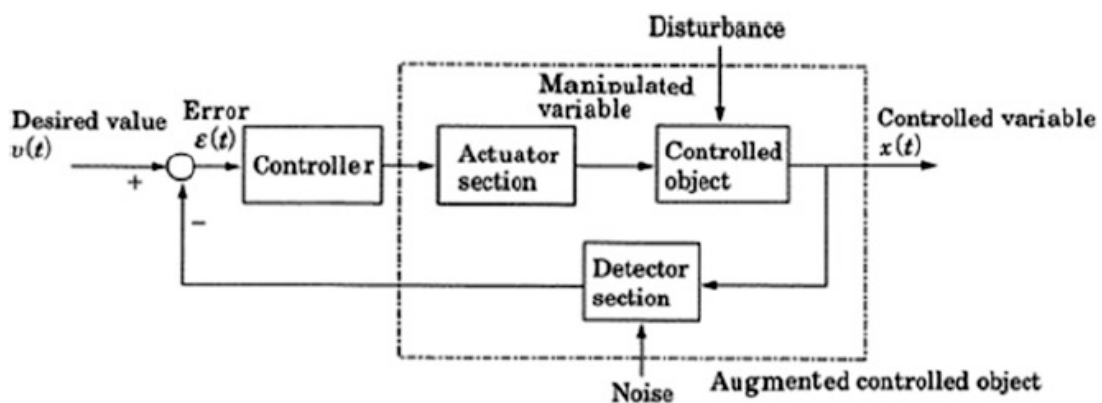


Figura 24. Esquema básico de un sistema de control [14].

3.6.2. Función de transferencia

La función de transferencia puede definirse como un modelo matemático expresado en el dominio de Laplace que, mediante un cociente, relaciona la entrada y la salida de un sistema. Esta función se obtiene a partir de las ecuaciones físicas que rigen el comportamiento del sistema en cuestión.

En el caso de un motor de corriente continua, las ecuaciones que modelan su comportamiento son las siguientes [15]:

$$L \cdot \frac{di}{dt} + i \cdot R + k_b \cdot \omega = V$$

$$J \cdot \frac{d\omega}{dt} + b \cdot \omega = T$$

$$T = k_M \cdot i$$

Ecuación 25. Ecuaciones diferenciales del motor de corriente continua.

donde L es la inductancia del circuito de armadura, i es la corriente de armadura, R es la resistencia de armadura, k_b es la constante de velocidad, ω es la velocidad angular del motor, V es la tensión de armadura, J es el momento de inercia del rotor, b es el coeficiente de fricción viscosa, T es el par de carga y k_M es la constante de par.

El diagrama de bloques es el mostrado en la Figura 25.

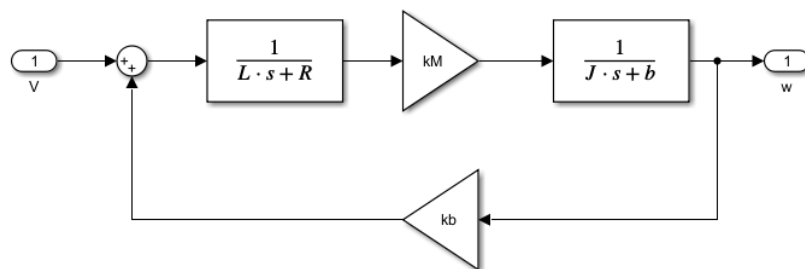


Figura 25. Diagrama de bloques del motor de corriente continua.

La función de transferencia resultante es:

$$\frac{\omega(s)}{V(s)} = \frac{k_M}{(J \cdot s + b) \cdot (L \cdot s + R) + k_b \cdot K_M}$$

Ecuación 26. Función de transferencia del motor de corriente continua.

La Figura 26 muestra el diagrama de bloques de un control en lazo cerrado, donde $C(s)$ es la función de transferencia del controlador, $P(s)$ es la función de transferencia del proceso que se desea controlar y $H(s)$ la del sensor que mide el valor de la salida. Por otra parte, $x(t)$ es la variable controlada, $v(t)$ es la señal de referencia, $d(t)$ es una perturbación que afecta al proceso y $n(t)$ es el ruido asociado a la medida del sensor [14].

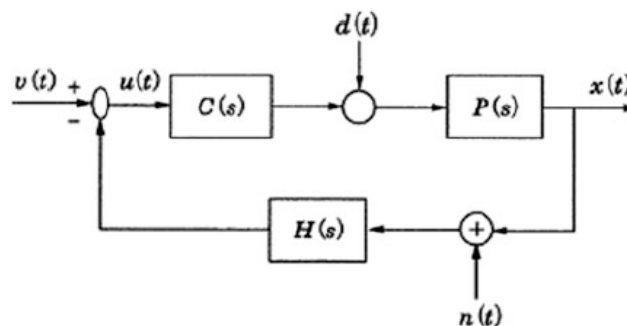


Figura 26. Diagrama de bloques de un sistema de control en bucle cerrado [14].

A partir de este diagrama, se obtienen las funciones de transferencia que relacionan la salida con la referencia, la perturbación y el ruido de la medida.

$$G_{xv(s)} = \frac{X(s)}{V(s)} = \frac{C(s) \cdot P(s)}{1 + C(s) \cdot P(s) \cdot H(s)}$$

Ecuación 27. Función de transferencia de la variable controlada frente a la referencia [14].

$$G_{xd(s)} = \frac{X(s)}{D(s)} = \frac{P(s)}{1 + C(s) \cdot P(s) \cdot H(s)}$$

Ecuación 28. Función de transferencia de la variable controlada frente a la perturbación [14].

$$G_{xn(s)} = \frac{X(s)}{N(s)} = \frac{-C(s) \cdot P(s) \cdot H(s)}{1 + C(s) \cdot P(s) \cdot H(s)}$$

Ecuación 29. Función de transferencia de la variable controlada frente al ruido en la medida [14].

La salida del sistema es la suma de todas las funciones de transferencia multiplicadas por la entrada correspondiente.

$$X(s) = G_{xv(s)} \cdot V(s) + G_{xd(s)} \cdot D(s) + G_{xn(s)} \cdot N(s)$$

Ecuación 30. Salida del sistema [14].

El error $e(t) = v(t) - x(t)$ entre la referencia y el valor real de la variable de control queda:

$$E(s) = V(s) - X(s) = \left(1 - \frac{C(s) \cdot P(s)}{1 + C(s) \cdot P(s) \cdot H(s)}\right) \cdot V(s) - \frac{P(s)}{1 + C(s) \cdot P(s) \cdot H(s)} \cdot D(s) + \frac{C(s) \cdot P(s) \cdot H(s)}{1 + C(s) \cdot P(s) \cdot H(s)} \cdot N(s)$$

Ecuación 31. Error entre la referencia y la variable de control [14].

3.6.3. Respuesta del sistema

El objetivo del control es lograr una respuesta con unas características dadas.

En primer lugar, en todo control se persigue que la señal sea estable, esto es, que, pasado un tiempo, el valor de la variable controlada alcance un valor permanente. Si el sistema es inestable, oscilará indefinidamente con una amplitud creciente.

La estabilidad del sistema en bucle cerrado se determina mediante el cociente de la función de transferencia, también llamado ecuación característica.

$$1 + C(s) \cdot P(s) \cdot H(s) = 0$$

Ecuación 32. Ecuación característica de la función de transferencia [14].

El sistema será estable si los polos de la ecuación característica tienen parte real negativa, e inestable en caso contrario.

Una vez garantizada la estabilidad del sistema, se debe buscar alcanzar los requerimientos tanto en régimen transitorio como estacionario.

En lo que al régimen permanente se refiere el parámetro más relevante es el error de posición, entendido como la diferencia existente entre la referencia y el valor de la variable de control una vez superado el transitorio. Este error se calcula, cuando a la entrada se tiene un escalón, de la siguiente forma:

$$E_p = \lim_{s \rightarrow 0} s \cdot \frac{1}{1 + C(s) \cdot P(s)} \cdot \frac{a}{s}$$

Ecuación 33. Error en régimen permanente frente a entrada escalón de amplitud a .

donde a es la amplitud del escalón aplicado en la referencia.

En la respuesta transitoria, los parámetros más importantes son el tiempo de establecimiento y la sobreoscilación. El primero de ellos es el tiempo que transcurre hasta que el valor de la variable controlada alcanza el 98 % de su valor final. El segundo de ellos es el valor máximo, expresado como un porcentaje sobre el valor final, que alcanza la salida antes de alcanzar el régimen permanente.

El tiempo de establecimiento depende de la parte real de los polos. Si existen varios, se considera que el polo dominante, esto es, el de menor valor absoluto, es el que determina el tiempo de establecimiento, que se calcula como:

$$t_e = \frac{4}{\sigma}$$

Ecuación 34. Tiempo de establecimiento.

donde σ es el valor absoluto de la parte real del polo dominante.

La sobreoscilación se da cuando los polos son complejos, y se puede calcular según la siguiente expresión:

$$\delta = e^{\frac{-\sigma \cdot \pi}{\omega_p}}$$

Ecuación 35. Sobreoscilación.

donde ω_p es el valor de la parte compleja de los polos.

Nótese que la sobreoscilación coincidirá con el resultado de la Ecuación 35 si se trata de un sistema de 2º orden sin ceros. De existir más polos o ceros, servirá como aproximación, pero los polos adicionales y los ceros influirán también en la sobreoscilación.

3.6.4. Control PID

El control PID (Proporcional-Integral-Derivativo) es un tipo de control automático en lazo cerrado muy ampliamente utilizado. Actualmente el 90 % de los controladores utilizados en la industria son PID [14].

La Figura 27 muestra el diagrama de bloques de un sistema de control con un regulador PID.

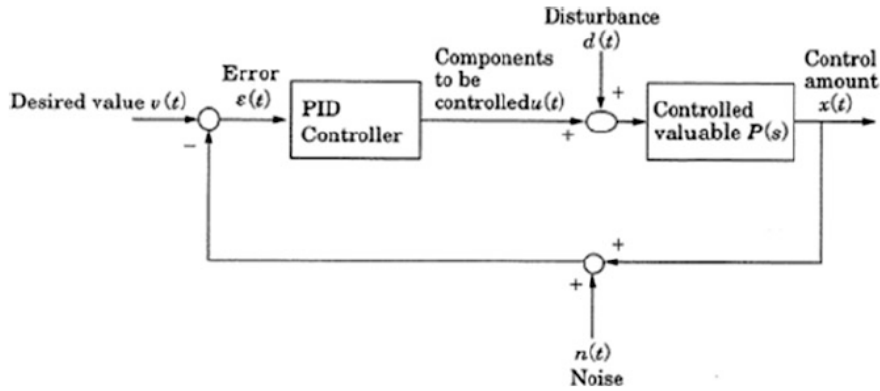


Figura 27. Diagrama de bloques de un sistema de control con regulador PID [14].

En un regulador PID existen tres acciones de control: proporcional, integral y derivativa.

La acción de control proporcional es, como su nombre indica, proporcional al error a través de una constante K_p . A mayor error en un instante dado, mayor acción de control proporcional. Aumentando el valor de K_p se logra reducir, aunque no eliminar por completo, el error en régimen permanente. Asimismo, tampoco logra eliminar el efecto de las perturbaciones sobre la variable controlada. Por otra parte, aumentar en exceso K_p puede inestabilizar el sistema.

La acción integral es proporcional a la integral del error mediante una constante K_I , es decir, tiene en cuenta la acumulación del error, sus valores en el pasado. Esta acción es capaz de suprimir el error en régimen estacionario incluso frente a perturbaciones. Por contra, una acción integral elevada también puede llevar a la inestabilidad del sistema.

Por último, la acción derivativa es proporcional a la derivada del error a través de una constante K_D . Esto permite anticiparse al error en el futuro, al ser proporcional a la velocidad con la que este varía. La acción derivativa permite reducir la sobreoscilación y lograr respuestas más suaves, pues si prevé que la salida va a sobrepasar la referencia puede restar acción de control antes de que suceda.

En definitiva, la acción proporcional se calcula con el valor del error presente, la acción integral, con la acumulación de errores pasados y la acción derivativa, con la tendencia del error futuro.

La acción de control se puede expresar de la siguiente forma:

$$u(t) = K_p \cdot e(t) + K_I \cdot \int_0^t e(t) \cdot dt + K_D \cdot \frac{d}{dt} \cdot e(t)$$

Ecuación 36. Acción de control del regulador PID [14].

Normalmente, se expresa de esta manera:

$$u(t) = K_p \cdot \left(e(t) + \frac{1}{T_I} \cdot \int_0^t e(t) \cdot dt + T_D \cdot \frac{d}{dt} \cdot e(t) \right)$$

Ecuación 37. Acción de control del regulador PID en función del tiempo integral y del tiempo derivativo [14].

donde T_I y T_D son el tiempo integral y el tiempo derivativo respectivamente.

En el dominio de Laplace, la función de transferencia del regulador queda como:

$$C(s) = K_p \cdot \left(1 + \frac{1}{T_I \cdot s} + T_D \cdot s \right)$$

Ecuación 38. Función de transferencia de un regulador PID [14].

Cabe destacar que los reguladores tipo PID no tienen que contener necesariamente las tres acciones simultáneamente. Pueden ser tipo P (solo Proporcional), PI (Proporcional-Integral), PD (Proporcional-Derivativo) o PID (Proporcional-Integral-Derivativo).

3.7. Seguimiento de trayectorias

El problema del seguimiento de trayectorias implica implementar un algoritmo que proporcione las referencias de velocidad para ambas ruedas para que el robot siga una determinada trayectoria, que estará formada por un conjunto de posiciones (coordenadas x e y) en el plano.

El seguimiento de trayectorias se encargará de calcular las consignas de la velocidad para ambas ruedas con el fin de alcanzar el siguiente punto de la ruta, corrigiendo el error en la orientación, en x y en y . Por otro lado, cada motor estará controlado mediante un regulador tipo PID (o alguna de sus variantes), que se ocupará de que la velocidad de las ruedas alcance la referencia con una respuesta aceptable. No obstante, estos reguladores quedan fuera del problema de seguimiento de trayectorias.

Existen numerosos algoritmos para el seguimiento de trayectorias. Uno de los más populares, debido a su sencillez, es el conocido como “*Follow the carrot*”.

Este algoritmo consiste en, dado un punto objetivo, corregir la orientación del robot, para que apunte hacia la meta y moverlo en esa dirección hasta alcanzarla. Cuando el vehículo llega al punto de destino, el punto objetivo pasa a ser el siguiente en la trayectoria.

En primer lugar, se determina el error entre la orientación deseada para llegar al punto objetivo desde el actual y la orientación real actual del robot [16]. Ese error se envía a un regulador de tipo PID (o alguna de sus variantes). Cada una de las ruedas recibe una referencia de velocidad igual. A esta referencia se suma en un caso y se resta en el otro, la acción de control resultante del regulador anterior multiplicada por una constante. De este modo, cuando existe un error en la orientación, se envía una acción de control mayor a una rueda y menor a la otra, lo que provoca el giro de robot corrigiéndose la desviación. Cuando el error de orientación es cero, ambas ruedas giran con idéntica velocidad, por lo que el robot avanza en línea recta.

El algoritmo descrito anteriormente se puede expresar como:

$$\omega_{i_{ref}}(s) = \frac{v_{ref}(s)}{r} + k_d \cdot R_\theta(s) \cdot E_\theta(s)$$
$$\omega_{d_{ref}}(s) = \frac{v_{ref}(s)}{r} - k_d \cdot R_\theta(s) \cdot E_\theta(s)$$

Ecuación 39. Velocidades de referencia de las ruedas con el algoritmo "follow the carrot".

donde $\omega_{i_{ref}}$ es la velocidad de referencia de la rueda izquierda, $\omega_{d_{ref}}$ es la velocidad de referencia de la rueda derecha, v_{ref} es la velocidad de referencia del robot, r es el radio de las ruedas, k_d es una constante, R_θ es la función de transferencia del regulador PID de la orientación y E_θ es el error de la orientación.

4. Diseño y selección de piezas mecánicas

En este apartado se explica el diseño en *SolidWorks* o la selección de las diferentes piezas que componen el robot, así como su ensamblaje.

4.1. Esfera

En primer lugar, para la esfera que constituye la parte externa del robot y que contiene en su interior el resto de los elementos mecánicos y electrónicos, es preciso tener en cuenta la necesidad de poder abrirla por la mitad, puesto que dentro de ella se han de disponer todos los componentes y se ha de poder acceder a ellos, no solo para su construcción sino también para su programación, recarga de batería, etc.

En la Figura 28 se puede observar la esfera seleccionada. Tiene un diámetro exterior de 30 cm e interior de 29.4 cm y puede desmontarse en dos semiesferas iguales. La unión entre ambas se hace de tal forma que la esfera queda lisa por fuera y por dentro, lo que la hace apta para este proyecto. Es además transparente, lo que permite ver su contenido. Esta esfera ha sido ideada para funciones decorativas, para lo cual, consta de un gancho para ser colgada. Para esta aplicación, se cortará y limará esta parte.



Figura 28. Esfera divisible en dos semiesferas seleccionada.

4.2. Chasis del vehículo

En el interior de la esfera se colca un vehículo de dos ruedas, que será el responsable de provocar el movimiento de la esfera. La primera parte del vehículo que se va a exponer es el chasis, que es el encargado de sostener y mantener unidas todas las piezas y componentes tanto mecánicos como electrónicos en general. La Figura 29 muestra una vista isométrica de la pieza.

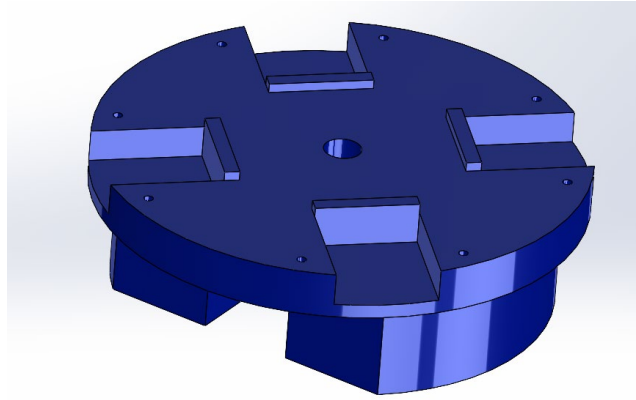


Figura 29. Vista isométrica del chasis del vehículo diseñado en SolidWorks.

Se procede ahora a explicar de manera más detallada algunas partes del chasis.

Primeramente, en la parte inferior existe un hueco de forma rectangular que atraviesa el chasis de lado a lado. En esta concavidad irán ubicados los dos motores de las ruedas. La Figura 30 muestra esta parte con más detalle.

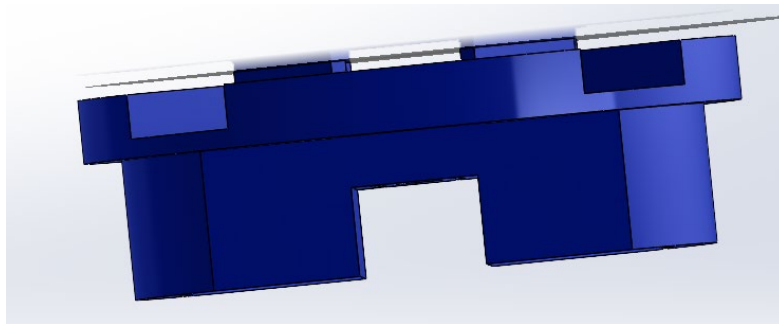


Figura 30. Hueco para albergar los motores de las ruedas en el chasis diseñado en SolidWorks.

Además, a ambos lados, en el lugar donde irán las ruedas existen, en la cara inferior de la parte superior del chasis, que tiene forma circular, unas hendiduras para garantizar que las ruedas no rocen, tal y como puede apreciarse en la Figura 31.

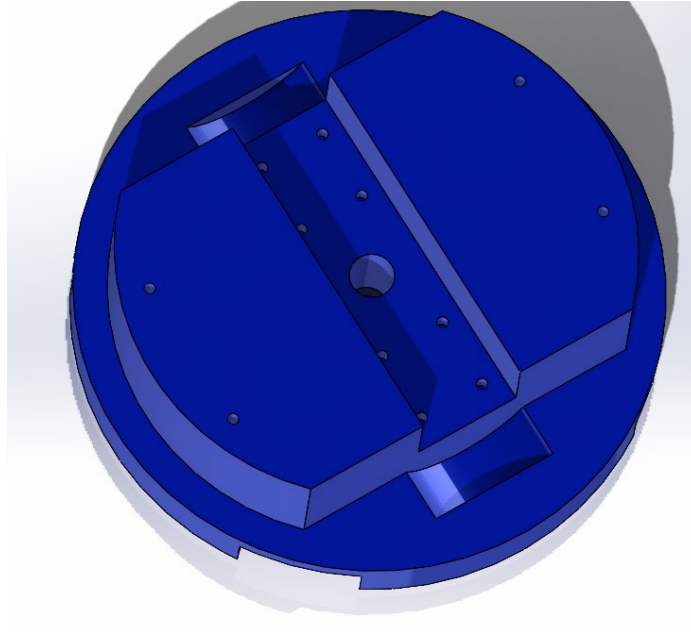


Figura 31. Vista inferior del chasis diseñado en SolidWorks.

Tal y como se observa en la Figura 31, la parte superior del chasis es circular, mientras que la inferior tiene partes circulares delante y detrás y está recortada en los lados para dar cabida a las ruedas.

Por otro lado, en la parte cara superior, existen cuatro hendiduras rectangulares dispuestas cada 90°. En ellas, se insertarán cuatro piezas que sujetarán ruedas esféricas que rodarán sobre la superficie interna de la esfera. En la parte interna de estas hendiduras, existe un borde con una altura superior a la del resto de la base. Entre esos bordes y las piezas que sujeten las ruedas esféricas se situará un muelle que garantizará en todo momento el contacto de estas ruedas con la superficie interior de la esfera. La Figura 32 muestra esta parte del chasis. Cabe resaltar que se han colocado a 45° del eje de las ruedas y no a 0° y 90° para que la hendidura realizada en la parte inferior para que las ruedas no toquen la parte superior del chasis no interfiera con estos huecos.

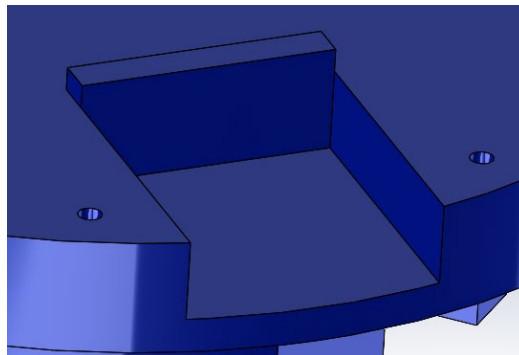


Figura 32. Hendidura para las ruedas esféricas en el chasis diseñado en SolidWorks.

Por último, existe un agujero circular en el centro del chasis cuya función es permitir el paso de los cables desde el microcontrolador, que se situará sobre el vehículo, y los motores y la batería, situados abajo. En la Figura 33 se observa este conducto.

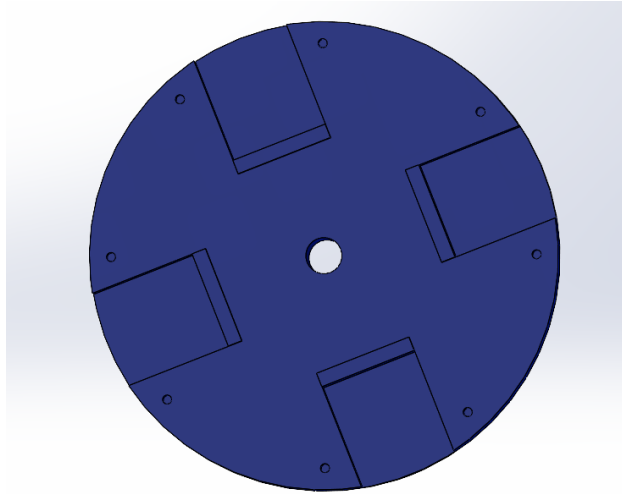


Figura 33. Vista superior del chasis diseñado en SolidWorks.

4.3. Parte inferior del vehículo

La pieza mostrada en la Figura 34 y la Figura 35 ha sido diseñada para colocarse bajo el chasis. Sobre ella se ubicarán los motores y, en el hueco existente en su interior, la batería que alimentará al robot.

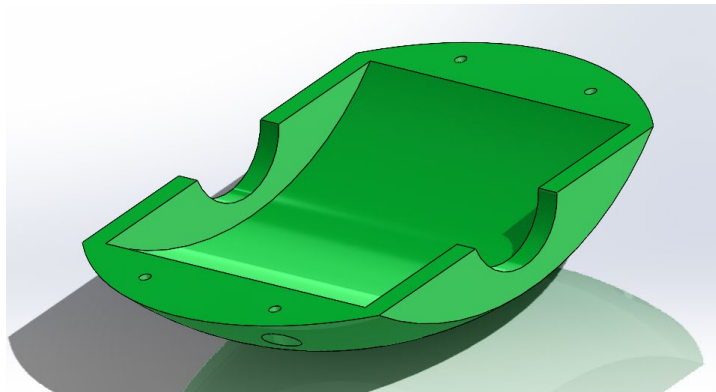


Figura 34. Vista isométrica de la parte inferior del vehículo diseñada en SolidWorks.

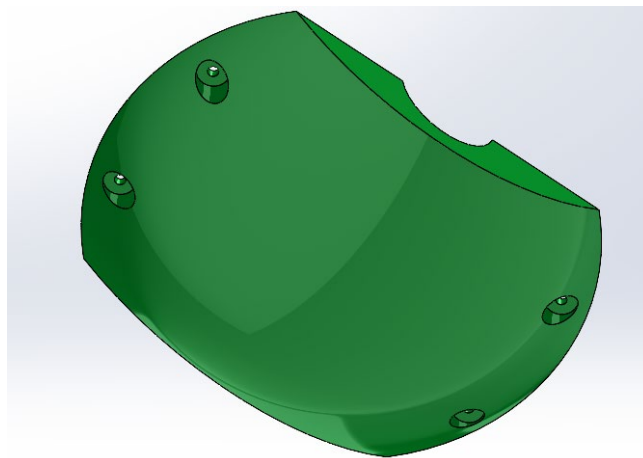


Figura 35. Vista inferior de la parte inferior del vehículo diseñada en SolidWorks.

4.4. Motor y encoder

Para realizar el ensamblaje en SolidWorks se ha utilizado un modelo de CAD disponible en la web del motor EMG30 con encoder, que se puede ver en la Figura 36. El robot consta de dos motores, uno para cada rueda.

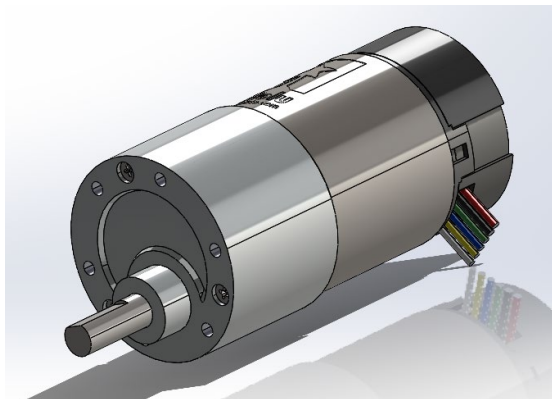


Figura 36. Modelo CAD del motor EMG30 con encoder.

4.5. Soporte para motor

Para la sujeción de los motores al chasis se ha diseñado el soporte mostrado en la Figura 37, al que irá atornillado el motor y que se fijará a su vez al chasis del vehículo.

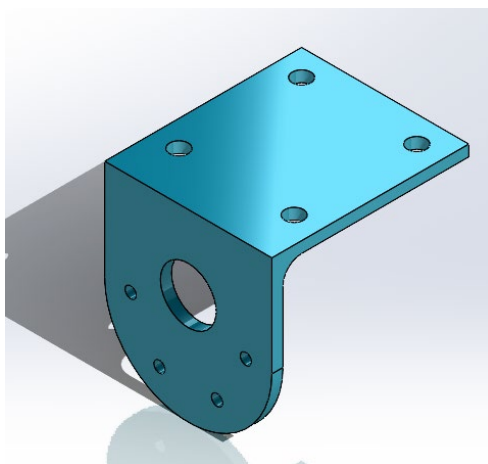


Figura 37. Soporte para motor.

4.6. Rueda esférica

Para fijar el vehículo en el interior de la esfera se ha optado por adquirir ruedas esféricas como las mostradas en la Figura 38, que rodarán sobre la superficie interna de la esfera.



Figura 38. Ruedas esféricas.

4.7. Soporte para rueda esférica

Para sujetar las ruedas esféricas se han diseñado cuatro piezas como la mostrada en la Figura 39. En la parte delantera se atornilla la rueda esférica y en el cilindro situado en la parte de atrás se colocará el resorte que garantizará el contacto entre la rueda esférica y la esfera que recubre al robot. Esta parte puede observarse en detalle en la Figura 40.

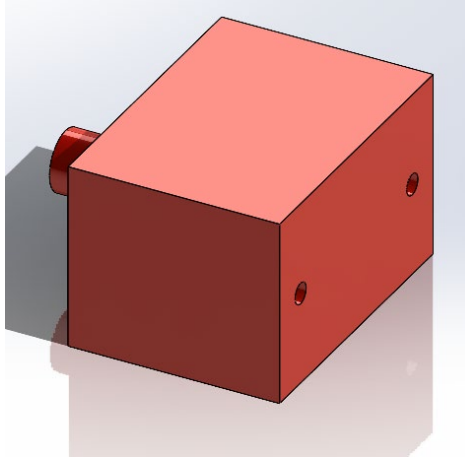


Figura 39. Pieza para sujetar una rueda esférica diseñada en SolidWorks.

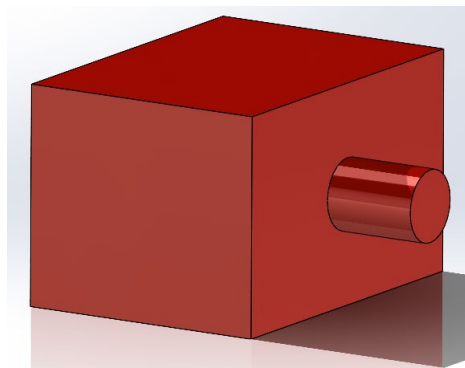


Figura 40. Parte trasera de la pieza para sujetar una rueda esférica diseñada en SolidWorks.

4.8. Sujeción para soporte de rueda esférica

Es necesaria una pieza que sujete el soporte de la rueda esférica permitiendo que se mueva hacia delante y hacia atrás por efecto del muelle, pero no hacia los lados o hacia arriba y abajo. Para tal efecto se ha diseñado la pieza mostrada en la Figura 41.

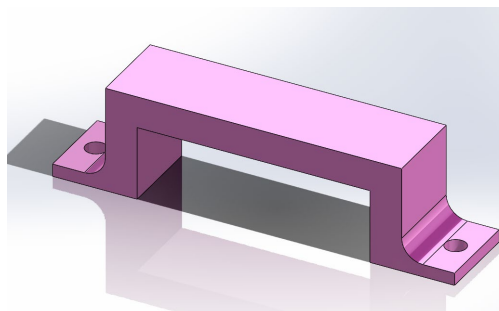


Figura 41. Pieza para sujetar el soporte de la rueda esférica diseñada en SolidWorks.

4.9. Resorte

Tal y como se ha comentado con anterioridad, se requieren cuatro muelles situados entre los soportes de las ruedas esféricas y el chasis que garanticen el contacto de estas con la esfera en todo momento. Se han comprado para ello resortes como el mostrado en la Figura 42.



Figura 42. Resorte para garantizar el contacto entre la esfera y las ruedas esféricas.

4.10. Rueda

Finalmente, se necesitan dos ruedas para el vehículo situado en el interior de la esfera. Para ello se han comprado dos ruedas como la mostrada en la Figura 43.



Figura 43. Rueda seleccionada.

4.11. Ensamblaje

En la Figura 44 se muestra el ensamblaje completo del robot incluyendo todas las piezas anteriores.

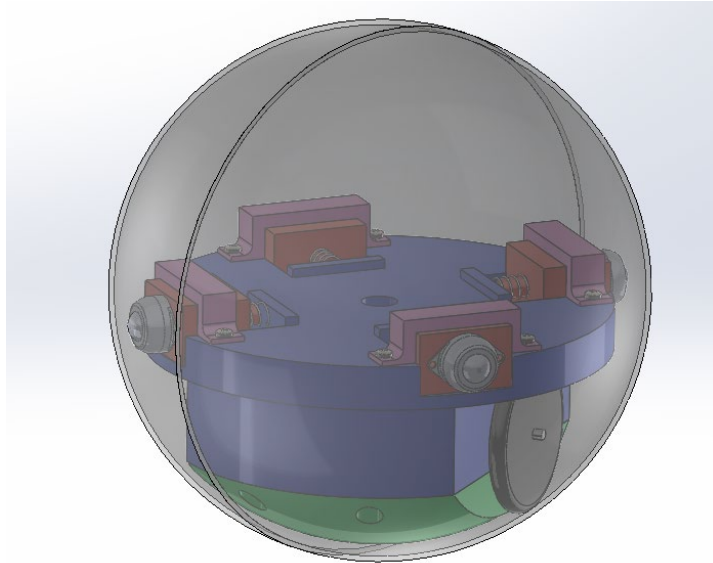


Figura 44. Ensamblaje del robot completo.

A continuación, se muestran en detalle algunas partes del ensamblaje.

En primer lugar, la Figura 45 muestra los motores colocados en sus soportes.

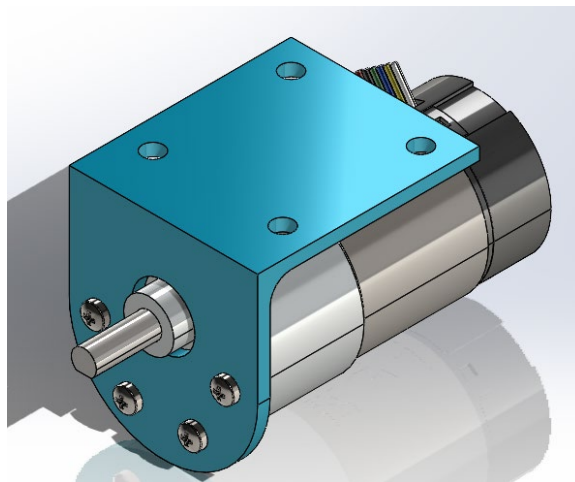


Figura 45. Motor montado en su soporte.

En segunda instancia, en la Figura 46 se observan los motores con ruedas en sus soportes atornillados al chasis.

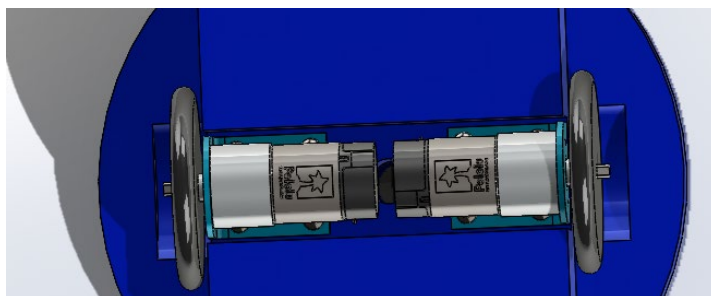


Figura 46. Motores con ruedas en sus soportes atornillados al chasis.

La Figura 47 y la Figura 48 muestran el conjunto formado por la rueda esférica, su soporte, la pieza que lo sujeta al chasis y el muelle.

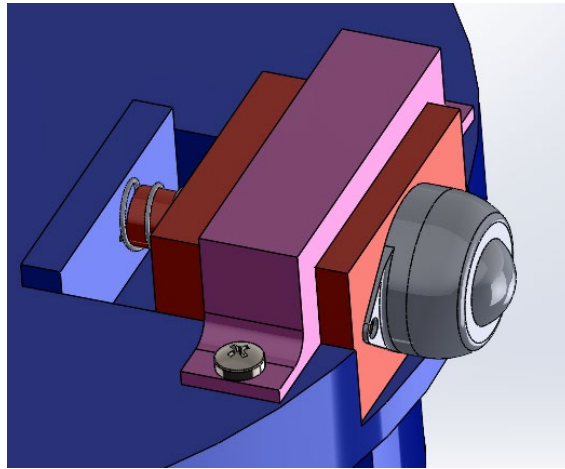


Figura 47. Soporte de rueda esférica montado en el chasis.

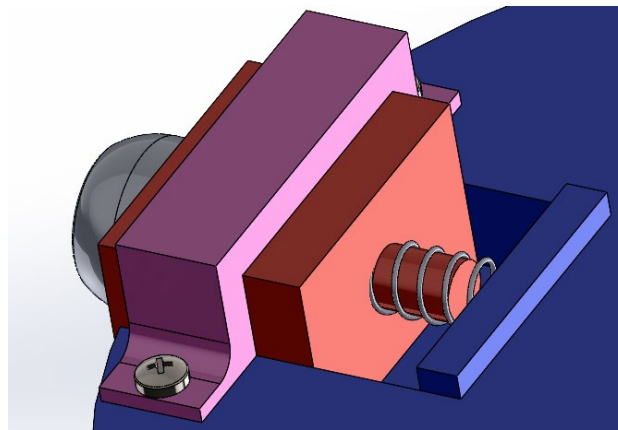


Figura 48. Parte trasera de soporte de rueda esférica montado en el chasis.

Por último, en la Figura 49 se puede apreciar el vehículo completo fuera de la esfera.

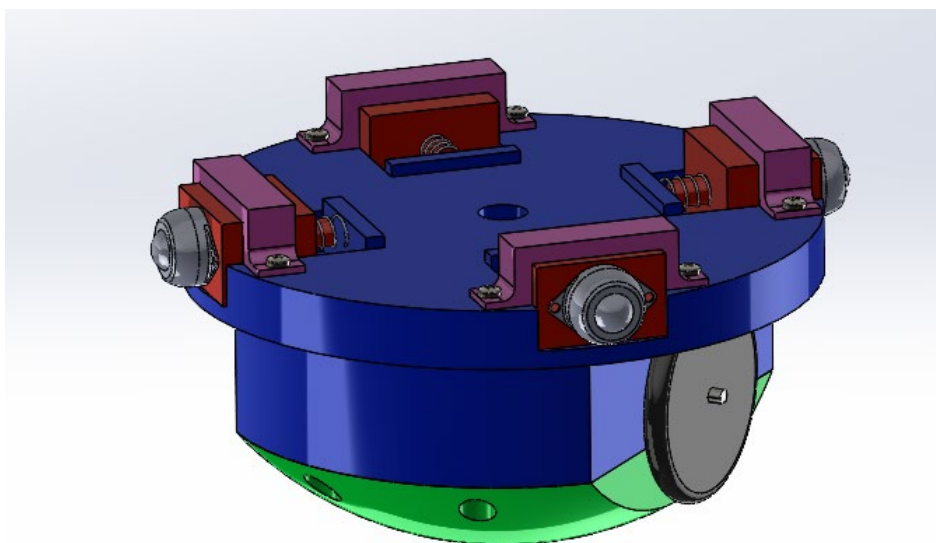


Figura 49. Ensamblaje del vehículo sin la esfera.

5. Simulación

5.1. Simscape Multibody

Simscape Multibody es una herramienta de MATLAB que ofrece un entorno de simulación para sistemas mecánicos en 3D, como pueden ser robots, suspensiones de vehículos, máquinas de construcción, trenes de aterrizaje, etc. Es posible modelar sistemas con múltiples cuerpos utilizando bloques que representan sólidos, articulaciones, restricciones, elementos de fuerza y sensores. Este software formula y soluciona las ecuaciones que rigen el movimiento del sistema mecánico. Además de ofrecer bloques con cuerpos de determinadas formas geométricas (esfera, prisma, cilindro) permite la importación de modelos CAD de piezas y ensamblajes, incluyendo las masas, inercias y articulaciones de las mismas. También se puede visualizar una simulación en 3D de la dinámica del sistema [17].

Cabe destacar que los bloques de *Simscape Multibody* se utilizan en el entorno de Simulink, por lo que, mediante los conversores de señales pertinentes, son compatibles con los elementos de otras librerías de *Simulink*, como los bloques *Scope* o *Step*.

A continuación, se detallan brevemente algunos de los bloques de *Simscape Multibody* utilizados en el desarrollo de este proyecto.

En primer lugar, se indica el bloque *Solver Configuration*, que puede verse en la Figura 50, en el que se especifican los parámetros de *solver* necesarios para realizar los cálculos y, por tanto, la simulación.

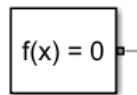


Figura 50. Bloque Solver Configuration de Simscape Multibody.

El bloque *World Frame*, en la Figura 51, representa el sistema de referencia fijo. Todos los sistemas están unidos a él mediante transformaciones y/o articulaciones que definen su posición y grados de libertad.

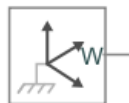


Figura 51. Bloque World Frame de Simscape Multibody.

El Bloque *Mechanism Configuration*, mostrado en la Figura 52, contiene un vector de aceleraciones en el que se puede introducir, eliminar o cambiar de eje la aceleración de la gravedad. En este bloque se decide si se desea tener en cuenta o no la gravedad en las simulaciones.

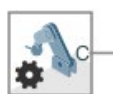


Figura 52. Bloque Mechanism Configuration en Simscape Multibody.

El bloque *Rigid Transform*, que puede verse en la Figura 53, establece una relación de posición y orientación entre dos elementos mediante un desplazamiento en los ejes cartesianos y uno o varios giros con respecto a ellos, definiendo así una matriz de transformación.

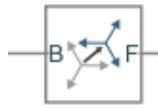


Figura 53. Bloque *Rigid Transform* en *Simscape Multibody*.

Tal y como se ha comentado previamente, existen bloques que definen cuerpos geométricos, cuyas dimensiones, y cuya masa o densidad pueden especificarse. Los cuerpos geométricos que proporciona *Simscape Multibody* son la esfera, el cilindro y el prisma en los bloques *Spherical Solid*, *Cylindrical Solid* y *Brick Solid* respectivamente. Todos ellos aparecen representados en la Figura 54.



Figura 54. Bloques *Spherical Solid*, *Cylindrical Solid* y *Brick Solid* en *Simscape Multibody*.

Por otra parte, también existe el bloque *File Solid*, visible en la Figura 55, que no tiene una geometría predefinida, sino que permite importar un modelo CAD.



Figura 55. Bloque *File Solid* en *Simscape Multibody*.

Para definir los grados de libertad del mecanismo existen bloques que modelan articulaciones de distintos tipos: de revolución, prismática, esférica, planar, entre otras. La Figura 56, la Figura 57, la Figura 58 y la Figura 59 muestran las articulaciones anteriormente citadas. Todas las articulaciones ofrecen la opción de añadir una entrada para modificar su valor y salidas para obtener medidas.



Figura 56. Articulación de revolución en *Simscape Multibody*.

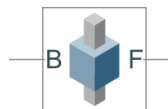


Figura 57. Articulación prismática en *Simscape Multibody*.

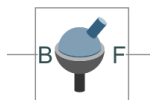


Figura 58. Articulación esférica en *Simscape Multibody*.



Figura 59. Articulación planar en Simscape Multibody.

Para conocer la posición y/u orientación de un cuerpo con respecto a otro se puede utilizar el bloque *Transform Sensor*, que permite seleccionar distintas magnitudes de posición, velocidad y aceleración para medirlas. En la Figura 60 se observa un bloque *Transform Sensor* que mide la posición en el eje x , la velocidad lineal en el eje y y la velocidad angular en el eje z .

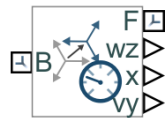


Figura 60. Bloque *Transform Sensor* en Simscape Multibody.

Los conectores *PS-Simulink Converter* y *Simulink-PS Converter*, mostrados en la Figura 61 y en la Figura 62 respectivamente, son convertidores que permiten unir elementos de *Simulink* con elementos de *Simscape Mutibody* y viceversa.



Figura 61. Conector *PS-Simulink Converter* en Simscape Multibody.



Figura 62. Conector *Simulink-PS Converter* en Simscape Multibody.

Por último, el bloque *Spatial Contact Force*, que se puede apreciar en la Figura 63, toma las envolventes convexas de dos cuerpos, para lo cual previamente debe activarse la opción *Convex Hull* en los bloques de dichos cuerpos, y crea una fuerza de contacto que impide que un cuerpo atraviese al otro. En aplicaciones de robótica móvil y de vehículos en general, es necesario para que las ruedas del vehículo se mantengan sobre la superficie del suelo y no lo atraviesen y caigan indefinidamente por acción de la gravedad.



Figura 63. Bloque *Spatial Contact Force* en Simscape Multibody.

En cuanto a la simulación, cabe decir que puede verse desde distintas vistas (isométrica, planta, alzado, perfil, etc.), se puede partir la pantalla para ver distintas vistas simultáneamente e incluso se pueden colocar cámaras en elementos móviles y visualizar la simulación desde dichas cámaras.

5.2. Librería *Contact Forces*

El bloque *Spatial Contact Force* presenta un problema en este proyecto. Es necesario definir una fuerza de contacto entre la superficie interna de la esfera y las ruedas del vehículo en su interior. No obstante, dado que este bloque toma un recubrimiento convexo de los cuerpos, no permite tomar la superficie interna de las semiesferas. Por esta razón no es posible simular el robot con el bloque *Spatial Contact Force*.

Para solucionar este problema se ha optado por utilizar una librería existente: *Contact Force Library*. Esta librería contiene numerosos bloques que modelan fuerzas de contacto entre cuerpos de diferentes formas con o sin fricción, tanto en 2D como en 3D. La Figura 64 y la Figura 65 muestran estos bloques para dos y tres dimensiones respectivamente.

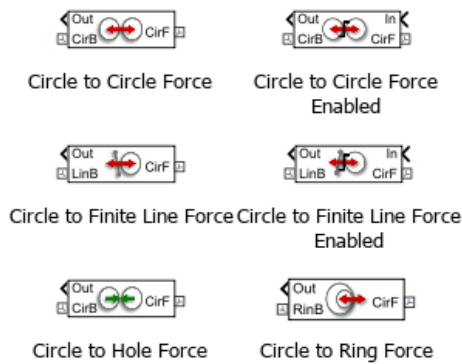


Figura 64. Bloques de fuerzas de contacto para cuerpos 2D en la librería *Contact Forces Library*.

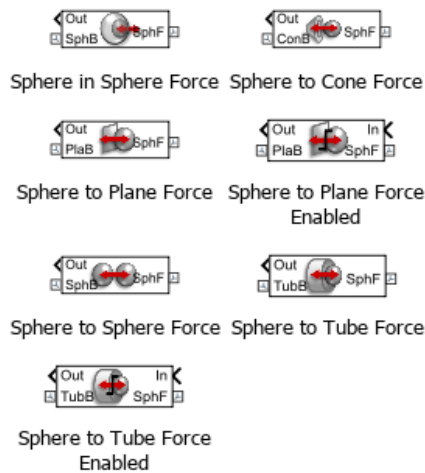


Figura 65. Bloques de fuerzas de contacto para cuerpos 3D en la librería *Contact Forces Library*.

En este proyecto se ha utilizado el bloque *Sphere in Sphere Force*, visible en la Figura 66, para modelar el contacto entre las ruedas del vehículo situado en el interior de la esfera y la superficie interna de esta.



Figura 66. Bloque *Sphere in Sphere Force* de la librería *Contact Forces Library*.

5.3. Modelo en Simscape Multibody

5.3.1. Modelo del robot

Con el fin de poder simular el comportamiento del robot es necesario importar las piezas diseñadas en *SolidWorks* a *Simscape Multibody* y relacionarlas mediante transformaciones y articulaciones.

En este caso, para evitar una carga computacional excesiva y optimizar los tiempos de cálculo, se ha decidido considerar todas las piezas impresas del vehículo, junto con los muelles y ruedas esféricas como un único cuerpo, mientras que los motores, sus soportes, las ruedas y la esfera se han importado como cuerpos independientes. Cabe resaltar que las dos semiesferas se han ensamblado y se han considerado como un único cuerpo para mayor simplicidad.

La Figura 67 muestra el esquema en *Simulink* con el modelo del robot que a continuación se explica con mayor detalle.

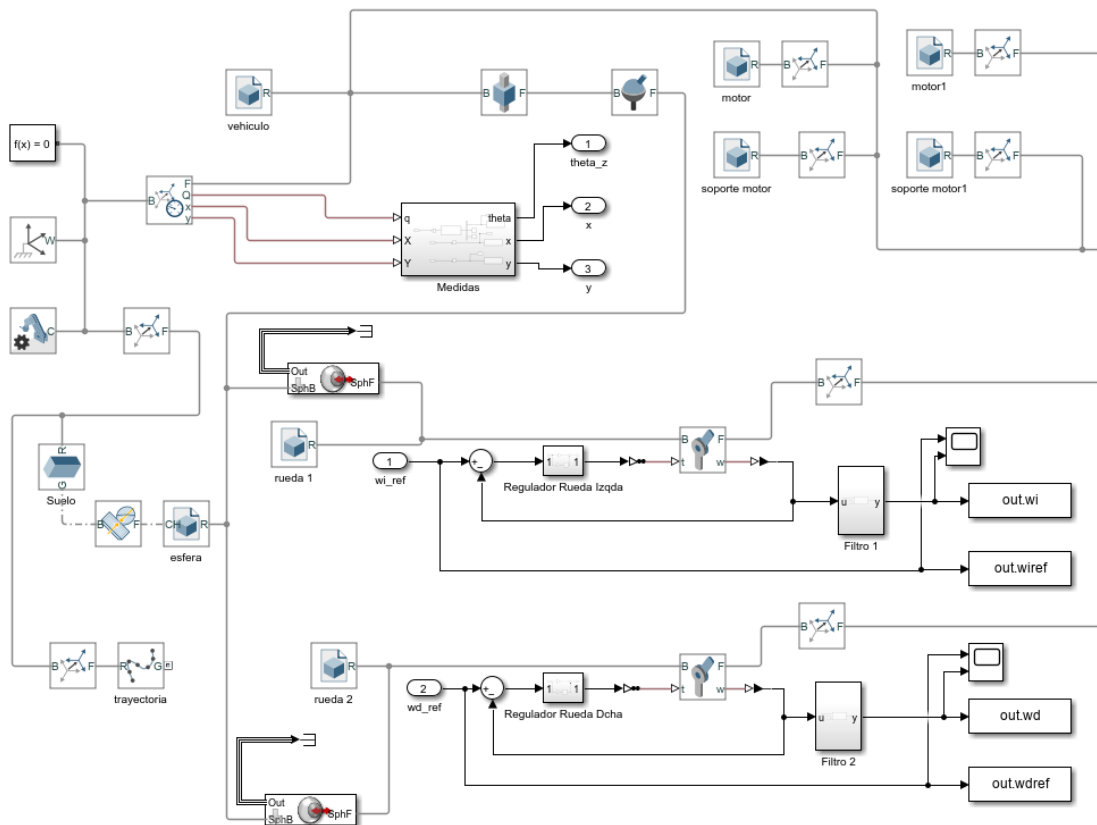


Figura 67. Esquema en Simulink del modelo del robot.

En el diagrama de la Figura 67, en primer lugar, se colocan los bloques *Mechanism Configuration* y *World Frame* junto con el bloque *Solver Configuration*. En *Mechanism Configuration* se le ha dado a la aceleración de la gravedad un valor de 9.81 m/s^2 en el sentido negativo del eje z. Esta parte del esquema se puede ver en detalle en la Figura 68.

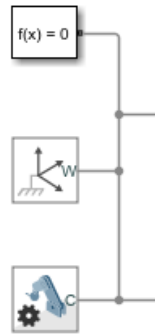


Figura 68. Detalle de los bloques Solver Configuration, World Frame y Mechanism Configuration en el esquema del modelo del robot en Simulink.

En segundo lugar, se ha utilizado un bloque tipo *Brick Solid* (prisma) para modelar el suelo y se ha conectado, mediante una transformación (bloque *Rigid Transformation*) al sistema de referencia global (bloque *World Frame*). El suelo se une a la esfera, que ha sido importada mediante un bloque *File Solid*, a través de un bloque *Spatial Contact Force*. Tal y como se ha comentado en el apartado 5.1, este bloque requiere activar la opción *Convex Hull* tanto en el bloque del suelo como en el de la esfera, y los recubre con una envolvente convexa para aplicar una fuerza de contacto que impide que los cuerpos se atraviesen. De este modo se logra que la esfera se mueva sobre la superficie. Además, se ha utilizado un bloque *Spline* que permitirá dibujar trayectorias sobre el suelo. Este bloque se une al del suelo mediante una transformación. Todos estos bloques se muestran en detalle en la Figura 69.

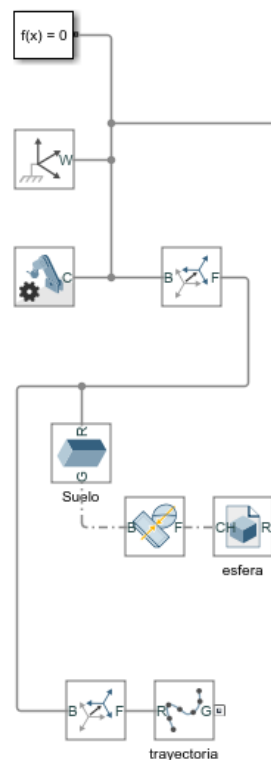


Figura 69. Detalle de los bloques correspondientes al suelo y a la esfera en el esquema del modelo del robot en Simulink.

En el bloque de la esfera se especifica su densidad, que es la del metacrilato (1.18 g/cm^3). Además, se selecciona su color y opacidad. En este caso interesa una opacidad baja para

poder ver los elementos que se van a colocar en su interior. En la Figura 70 se puede ver el CAD de la esfera importado.

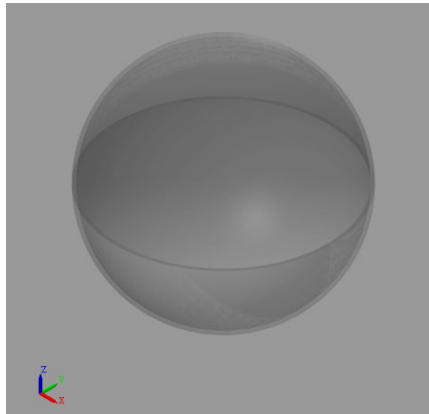


Figura 70. CAD de la esfera importado en Simscape Multibody.

La esfera se une mediante un bloque *Sphere to Sphere Force* de la librería *Contact Forces Library* con cada una de las ruedas, que también se importan mediante el bloque *Solid File*, especificándose su densidad. En la Figura 71 se observa en detalle la parte del esquema en la que se unen las ruedas y la esfera y la Figura 72 muestra el CAD de una de las ruedas importado en *Simscape Multibody*.

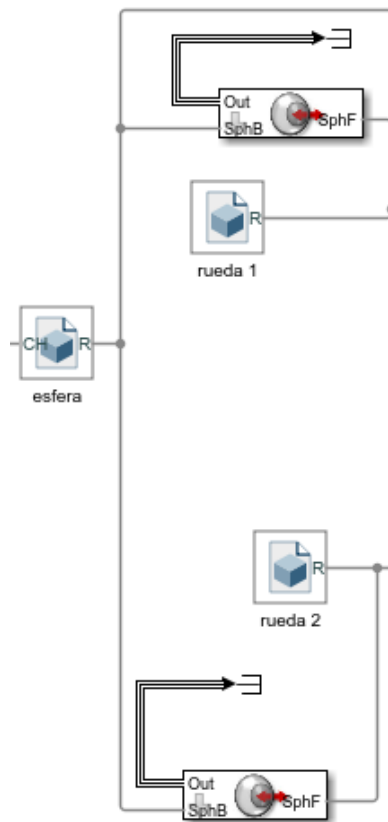


Figura 71. Detalle de la unión entre la esfera y las ruedas en el modelo del robot en Simulink.

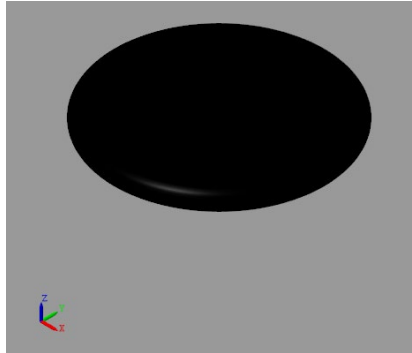


Figura 72. CAD de la rueda importado en Simscape Multibody.

A continuación, las ruedas se unen al vehículo mediante articulaciones de revolución que estarán accionadas por los motores. Estas articulaciones, además de hacer de conexión entre el CAD de las ruedas y el de los motores, reciben una entrada de par, que proviene de un regulador tipo PID. Dado que se trata de un bucle cerrado de control, también es necesaria una salida que permita medir el valor de la velocidad de la articulación, para compararla con la referencia y enviar el error entre ambas a la entrada del regulador. Para pasar la medida de *Simscape Multibody* a *Simulink* se requiere un conector *PS-Simulink Converter* y para pasar la acción de control de *Simulink* a *Simscape Multibody*, un conector *Simulink-PS Converter*. Se han añadido además bloques *Scope* para visualizar la velocidad de referencia y la velocidad real de ambas ruedas y bloques *To Workspace* para enviarlas al *Workspace* de MATLAB. Se han usado también filtros para eliminar ruido en las medidas. En la Figura 73 se puede apreciar esta parte del esquema.

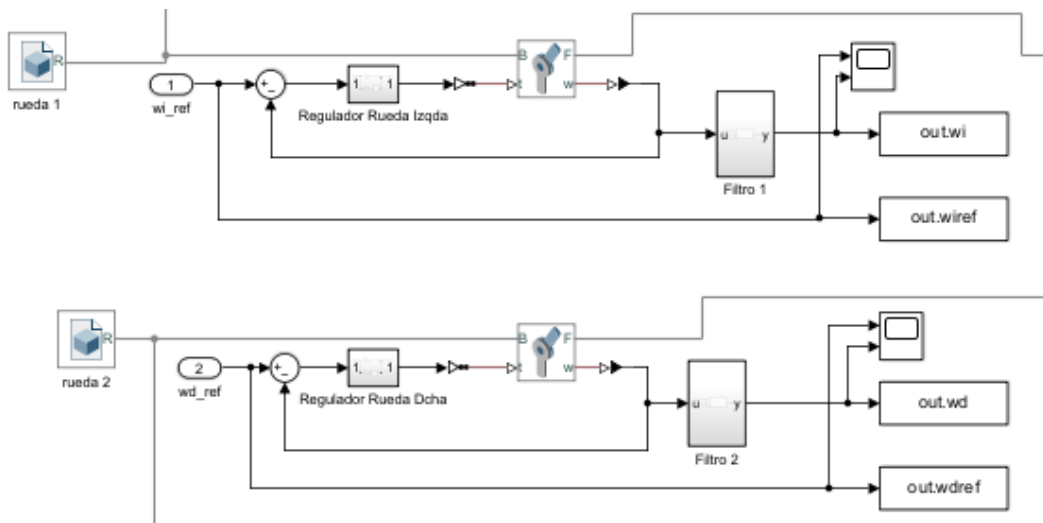


Figura 73. Articulaciones de revolución de las ruedas y reguladores de los motores en el modelo del robot en Simulink.

En cuanto al filtro, se muestra en la Figura 74 el contenido del subsistema, que no es más que un bloque *Varying Lowpass Filter*, es decir, un filtro paso-bajo con frecuencia de corte de 20 Hz. La Figura 75 muestra la señal antes de pasar por el filtro y la Figura 76, después, lo que permite apreciar en primer lugar la gran cantidad de ruido presente en la medida y en segunda instancia la efectividad del filtro.

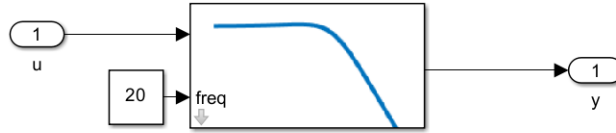


Figura 74. Filtro paso-bajo en la medida de la velocidad de las ruedas en el modelo del robot en Simulink.

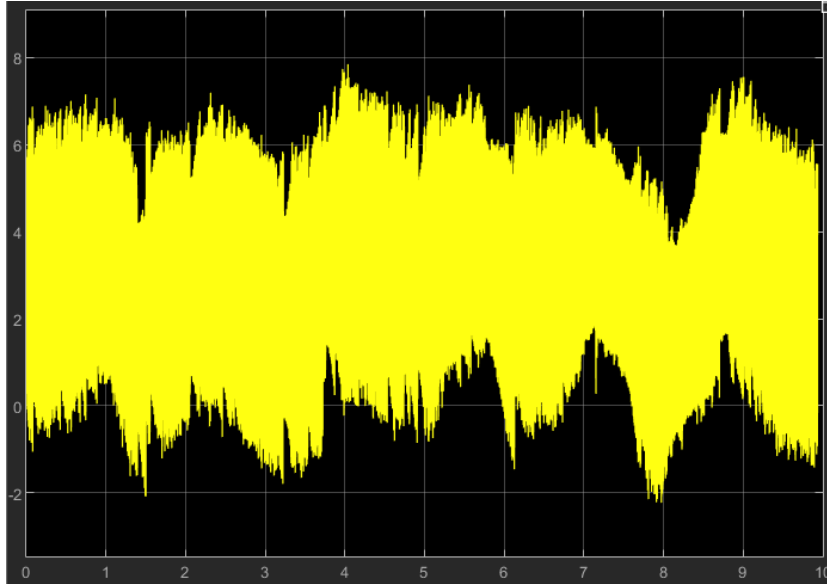


Figura 75. Medida de la velocidad de una rueda sin filtrar en el modelo del robot en Simulink.

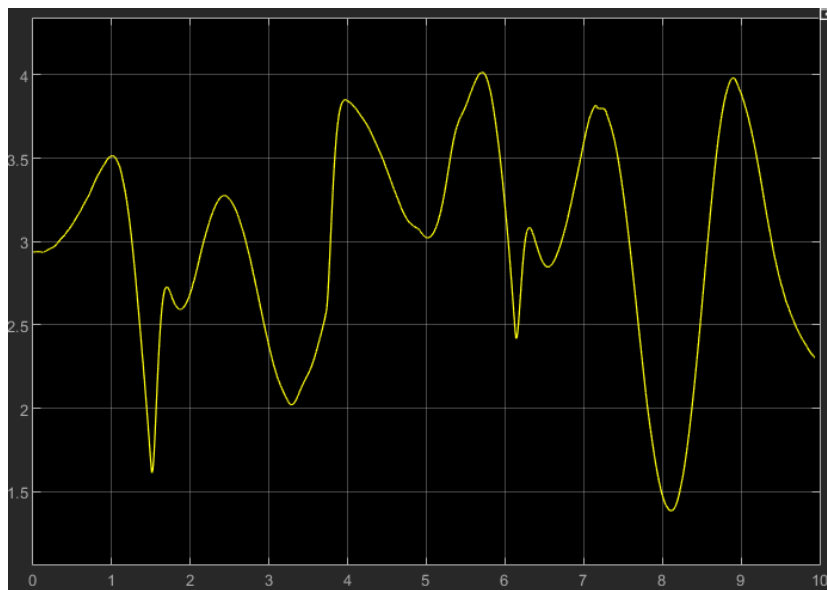


Figura 76. Medida de la velocidad de una rueda filtrada en el modelo del robot en Simulink.

Seguidamente, las articulaciones de revolución se han de unir también al CAD de los motores, pasando previamente por las transformaciones precisas (bloque *Rigid Transform*) para que los ejes y las distancias coincidan con el diseño. Los motores están unidos rígidamente a los soportes y estos al vehículo, en ambos casos también mediante transformaciones. Estas relaciones se muestran en la Figura 77.

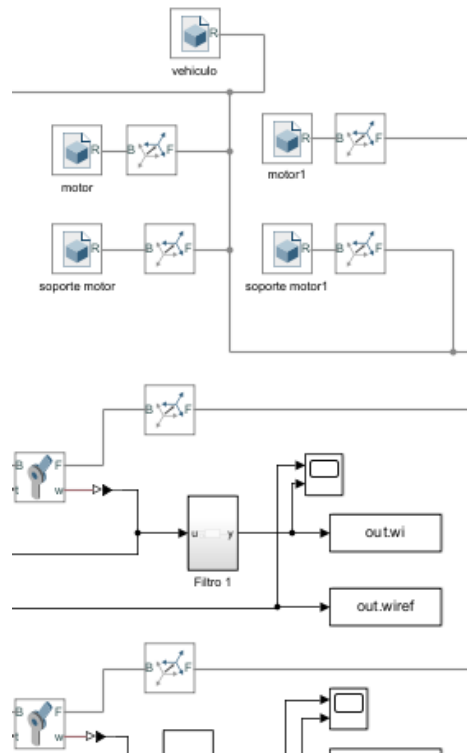


Figura 77. Detalle de la conexión entre las articulaciones de las ruedas con los motores y de estos con el vehículo en el modelo del robot en Simulink.

Naturalmente, los motores y sus soportes han sido importados a un bloque *File Solid*, y puede verse el CAD importado en la Figura 78 y la Figura 79 respectivamente. En el bloque del motor se ha especificado su peso (200 g) y en el del soporte la densidad del plástico PLA (1.24 g/cm³).



Figura 78. CAD del motor importado en Simscape Multibody.

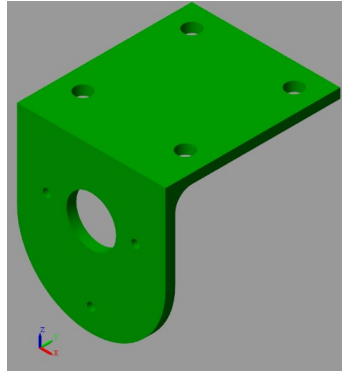


Figura 79. CAD del soporte del motor importado en Simscape Multibody.

Por último, el vehículo debe conectarse a la esfera mediante las articulaciones correspondientes. Concretamente se ha utilizado una articulación esférica, ya que el vehículo puede girar en todas las direcciones dentro de la esfera. Además, se ha colocado una articulación prismática, que permite al vehículo moverse verticalmente y es la gravedad la que lo mantiene en la parte inferior de la esfera, de forma análoga a lo que ocurre en el mundo real. Además, entre el vehículo y el sistema global (bloque *World Frame*) se ha colocado un bloque *Sensor Transform* para medir la posición en coordenadas x e y del robot, así como su orientación con respecto al eje z . En la Figura 80 se puede apreciar en detalle esta parte del modelo.

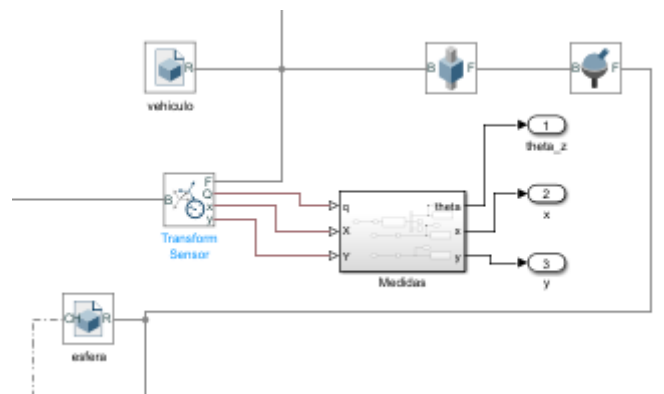


Figura 80. Detalle de la conexión del vehículo a la esfera y de las medidas de la posición y orientación del robot en el modelo del robot en Simulink.

La Figura 81 muestra el CAD del vehículo importado en un bloque *File Solid*. La densidad es la del plástico PLA (1.24 g/cm^3). Este CAD incluye también las ruedas esféricas y los resortes, que son de otros materiales, pero no se han considerado para no incrementar la carga computacional de la simulación.

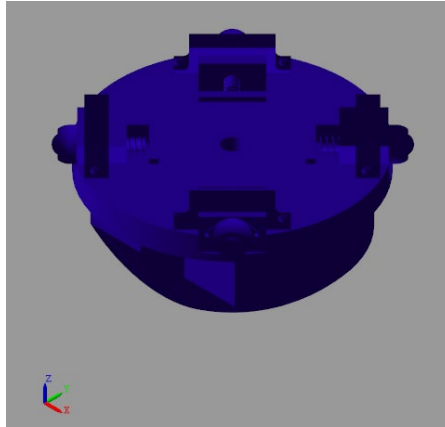


Figura 81. CAD del vehículo importado en Simscape Multibody.

El contenido del subsistema “Medidas” está en la Figura 82. Se usa un bloque *Rotation Order*, que mide la orientación en los tres ejes a partir de la salida *Q* del bloque *Transform Sensor*, para obtener el ángulo girado respecto al eje *z*, y se utilizan conectores *PS-Simulink Converter* para enviar la información al *Workspace* de MATLAB.

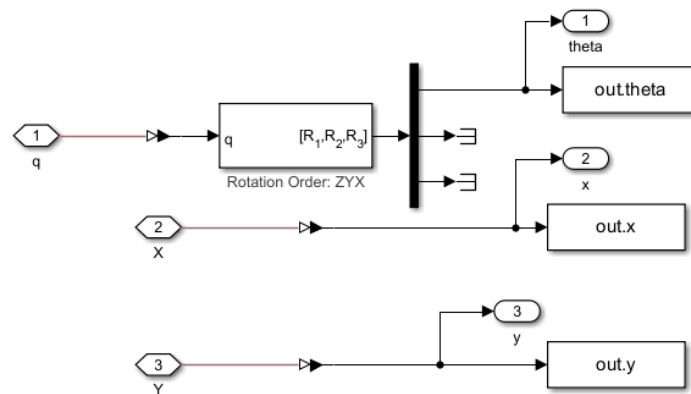


Figura 82. Contenido del subsistema de medida en el modelo del robot en Simulink.

Con esto se tiene el robot ensamblado en *Simscape Multibody* con sus correspondientes grados de libertad y articulaciones definidos. Con los reguladores adecuados para la velocidad de las ruedas y proporcionando las referencias de velocidad para cada rueda se pueden realizar simulaciones del movimiento de la esfera.

5.3.2. Reguladores de los motores

En este apartado se explica el ajuste de los controladores PID de los motores, que se sitúan en los subsistemas “Regulador Rueda Izqda” y “Regulador Rueda Dcha” en la Figura 67.

El objetivo es lograr un regulador que alcance la velocidad de referencia con error de posición nulo, con la menor sobreoscilación y en el menor tiempo posible.

De manera experimental se han ido ajustando los parámetros del regulador (ganancia proporcional, derivativa e integral) con el fin de lograr los objetivos. Se ha observado que con una ganancia proporcional de 0.5 se logra un tiempo de respuesta rápido y sin sobreoscilación, aunque con un pequeño error. Para ganancias menores el error aumenta y con valores mayores comienza a sobreoscilar. Con una ganancia proporcional de 0.5 se ha añadido una acción integral con una ganancia de 0.1, y se ha observado que se elimina

el error en régimen permanente. Con ganancias integrales mayores aumenta la inestabilidad y con menores el cambio no es significativo. Dado que con acción proporcional e integral la respuesta obtenida es aceptable al ser rápida y sin oscilaciones, no se ha considerado necesaria la acción derivativa, por lo que el regulador es de tipo PI. Cabe destacar que el controlador es el mismo para ambos motores.

La Figura 83 muestra el regulador diseñado, donde puede verse la ganancia proporcional e integral, así como el integrador. En la Figura 84 se muestra el seguimiento de la referencia con el controlador PI diseñado.

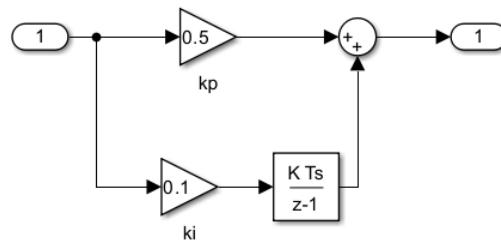


Figura 83. Regulador PI de los motores en el modelo del robot en Simulink.

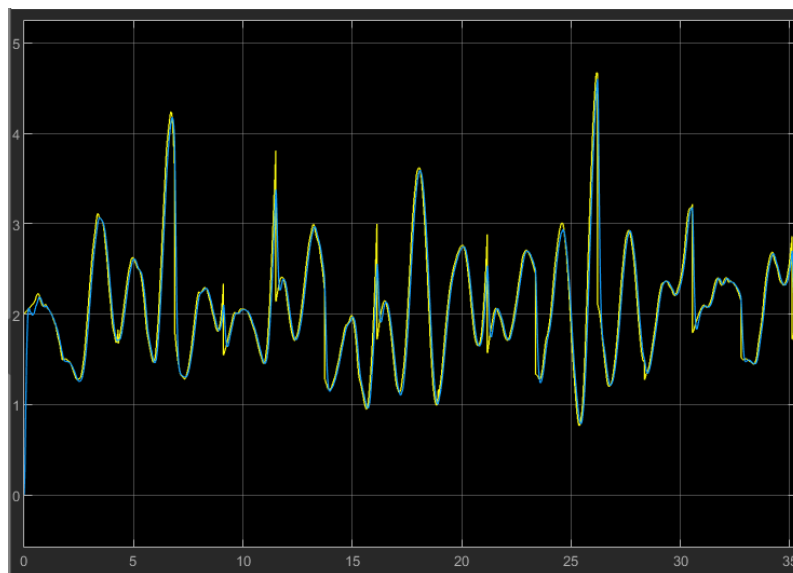


Figura 84. Seguimiento de la referencia de velocidad con el regulador diseñado en la simulación.

5.3.3. Implementación de la simulación a partir de velocidades de referencia

En primer lugar, se ha realizado una simulación en la que simplemente se da un valor de referencia a la velocidad de cada una de las ruedas mediante dos bloques *knob*. En la Figura 85 se observa el esquema en Simulink, donde el subsistema Robot contiene el modelo de la esfera explicado en el apartado 5.3.1.

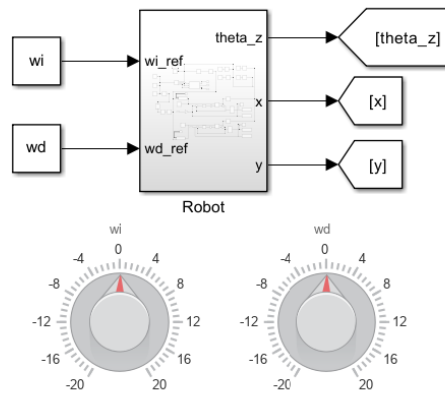


Figura 85. Simulación del robot a partir de las referencias de velocidad.

En la simulación se verifica que el robot permanece parado si la velocidad de referencia es cero en ambas ruedas, se mueve en línea recta si la velocidad de ambas es igual, describe curvas cuando una rueda gira a mayor velocidad que la otra y gira sobre sí mismo cuando las velocidades tienen idéntico valor, pero en sentidos opuestos. Estos resultados son acordes con el modelo cinemático del apartado 3.5.

5.3.4. Implementación de la simulación del desplazamiento a un punto

Como paso previo a la simulación del seguimiento de trayectorias se simula el posicionamiento del robot en un punto dado partiendo de un punto inicial. Una vez resuelto este problema, el seguimiento de trayectorias no es más que una sucesión de puntos en la que, cuando se alcanza un punto objetivo, el siguiente punto en la trayectoria pasa a ser el nuevo punto objetivo.

En la Figura 86 se puede observar el esquema realizado en Simulink para alcanzar un punto de destino. Se utilizan dos constantes para fijar las coordenadas x e y del punto objetivo. El subsistema “Robot” contiene el modelo del robot descrito en el apartado 5.3.1 con los reguladores que se especifican en el apartado 5.3.2. El subsistema “Distancia y ángulo”, que se detalla a continuación, calcula a partir de los valores x e y de referencia y los medidos en el bloque *Transform Sensor*, el ángulo y la distancia existente entre ambos puntos. Por último, el subsistema “Velocidades de referencia”, que se explica más adelante, obtiene las velocidades de referencia de los motores a partir del ángulo de orientación de referencia, calculado en el subsistema anterior, y de la orientación medida. Además, cuando la distancia entre el punto objetivo y el punto en el que se encuentra el robot es menor que cierto valor de tolerancia r se detiene la simulación.

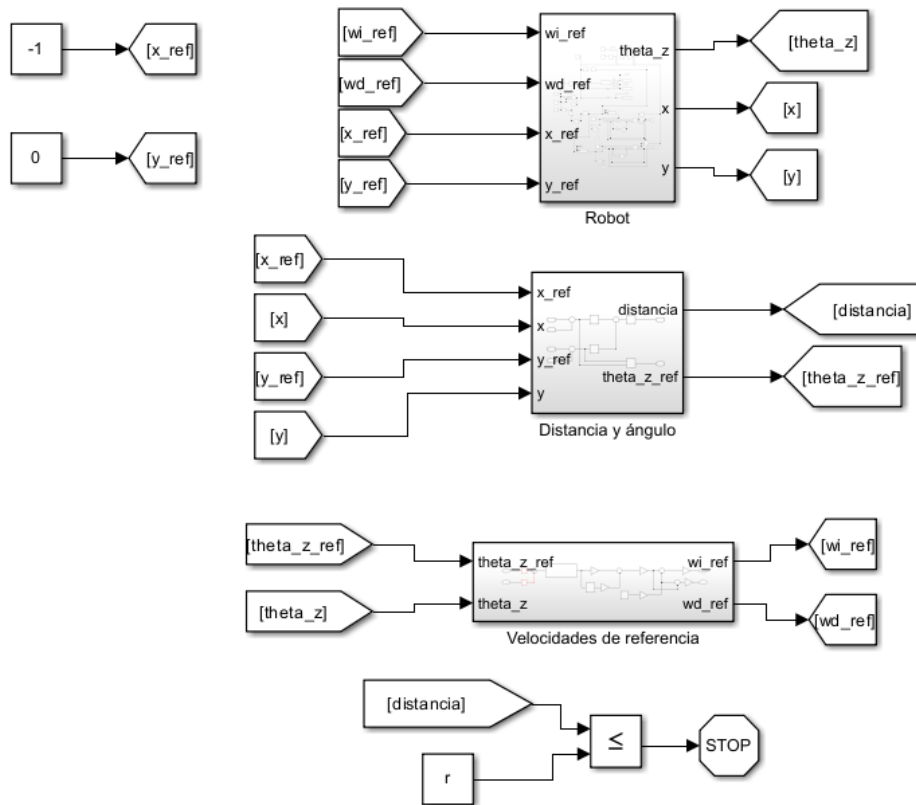


Figura 86. Esquema en Simulink para el desplazamiento del robot a un punto objetivo.

En la Figura 87 se aprecia el contenido del subsistema “Distancia y ángulo”, en el que simplemente se calcula la distancia entre el punto de referencia y el que ocupa realmente el robot como la raíz cuadrada de la suma de los cuadrados de las distancias en x y en y , calculadas como la diferencia entre los valores reales y el de referencia, según el Teorema de Pitágoras. El ángulo de la recta que une el punto de referencia y el medido se calcula mediante el bloque *atan2*, que calcula el arco tangente teniendo en cuenta los signos de las distancias en y y en x , que recibe como entradas.

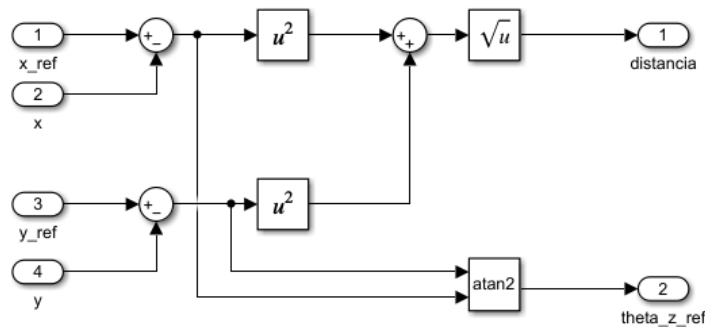


Figura 87. Contenido del bloque “Distancia y ángulo” del esquema de Simulink para el desplazamiento del robot a un punto objetivo.

La Figura 88 muestra el contenido del subsistema “Velocidades de referencia”. En primer lugar, se calcula el error de la orientación como la diferencia entre el ángulo formado por la recta que une el punto actual y el punto objetivo, y la orientación medida mediante el

bloque *Trasnform Sensor*. El subsistema “Límites ángulo” limita el ángulo de error a valores en el intervalo $[-\pi, \pi]$. El error de orientación es la entrada a un regulador tipo PI, cuyas constantes proporcional e integral, ajustadas de forma experimental, tienen respectivamente valores de 0.8 y 0.2. La acción de control se multiplica a su vez por una ganancia K_d .

Por otra parte, se utiliza una constante K_v , que es una velocidad lineal de referencia para el robot, que se pasa a velocidad angular de las ruedas mediante el modelo matemático del apartado 3.5. La acción de control del regulador PI de la orientación multiplicada por K_d se suma a la rueda izquierda y se resta a la derecha, con lo que se consigue que la existencia de un error en la orientación dé lugar a velocidades distintas en las ruedas que provoquen un giro que corrija la orientación hasta alinear el robot con la recta que une su posición con el punto de destino. El valor de K_v se ha fijado en 0.1 m/s y el de K_d se ha ajustado mediante prueba y error a lo largo de varias simulaciones hasta un valor de 50.

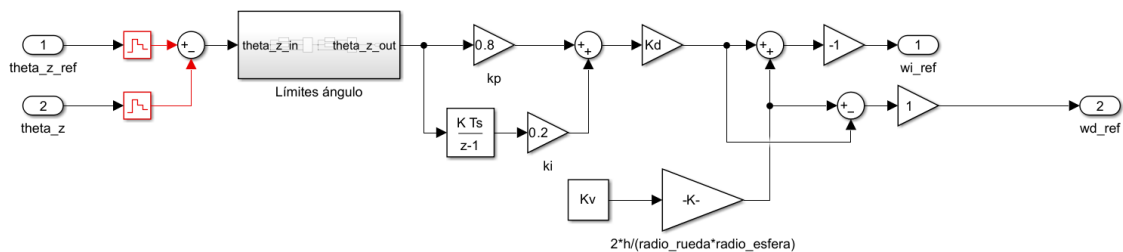


Figura 88. Contenido del bloque “Velocidades de referencia” del esquema de Simulink para el desplazamiento del robot a un punto objetivo.

El contenido del subsistema “Límites ángulo” se puede apreciar en la Figura 89. Tal y como se ha comentado previamente, se comprueba si el valor de entrada sobrepasa los límites del intervalo $[-\pi, \pi]$, en cuyo caso se resta o se suma $2\cdot\pi$, para que el ángulo quede dentro del rango comprendido entre $-\pi$ y π .

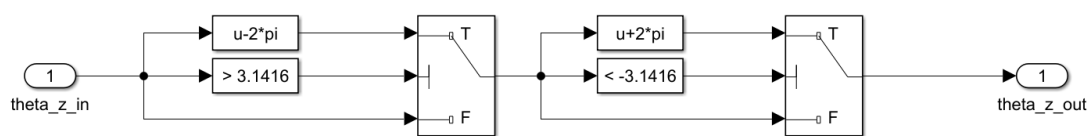


Figura 89. Contenido del subsistema “Límites ángulo”.

Por último, cabe destacar que, dentro del subsistema “Robot” se halla el modelo del robot explicado en el apartado 5.3.1, con dos pequeñas diferencias. En primer lugar, se ha eliminado la *Spline*, puesto que no hay una trayectoria en este caso y, en segundo lugar, se ha añadido un cilindro conectado al sistema de referencia mediante una articulación planar cuyos parámetros de entrada son la posición en x y en y , correspondientes al punto de destino. Este cilindro marca el punto objetivo en la simulación. En la Figura 90 se puede ver la implementación del cuerpo cilíndrico que marca la meta.

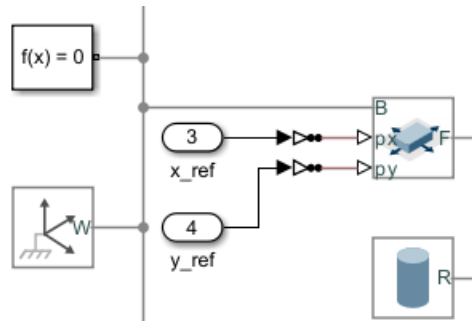


Figura 90. Cilindro añadido al subsistema "Robot" para marcar el punto objetivo en Simulink.

5.3.5. Verificación de la simulación del desplazamiento a un punto

Con el fin de comprobar el correcto funcionamiento del modelo creado en Simulink para el desplazamiento a un punto se han realizado cuatro simulaciones para cuatro puntos objetivos diferentes. La Figura 91 muestra una captura de la visualización de la simulación en la que se observa el robot y el cilindro que marca el destino. Los resultados obtenidos se muestran a continuación mediante gráficos que representan la trayectoria descrita por el robot.



Figura 91. Captura de la visualización de la simulación del desplazamiento a un punto en Simscape Multibody.

El primer punto objetivo tiene coordenadas (1,0). En el Gráfico 1 se aprecia la trayectoria descrita por el robot. Nótese que se le ha dado al parámetro r , que es la distancia a partir de la cual se considera alcanzado el punto, un valor de 0.1 m, por lo que no llega a alcanzar el destino, sino que finaliza la simulación a una distancia de 10 cm.

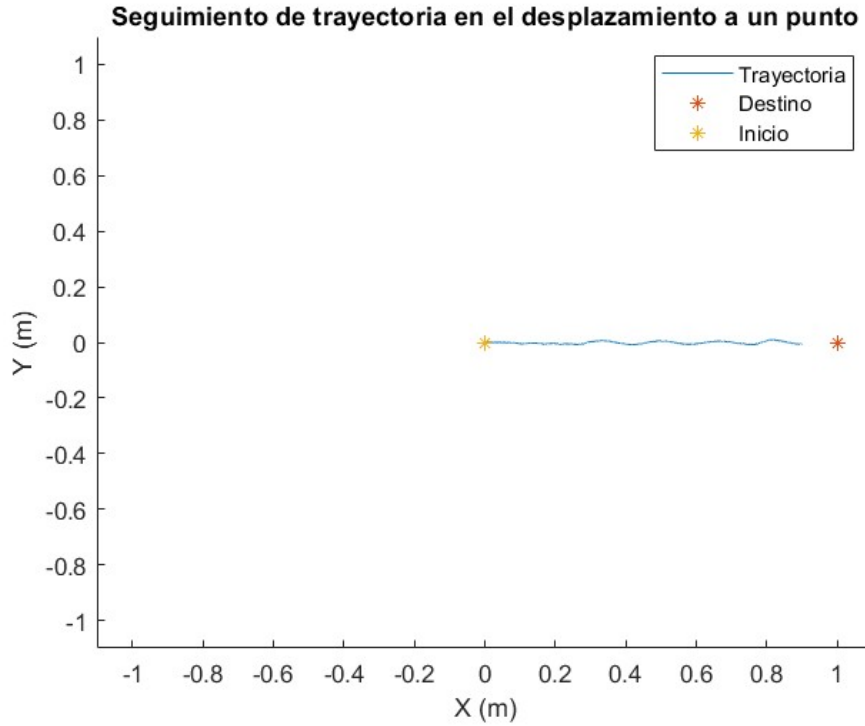


Gráfico 1. Trayectoria seguida en el desplazamiento al punto (1,0).

El segundo punto objetivo tiene coordenadas (0,1). En el Gráfico 2 se muestra la trayectoria seguida por el robot.

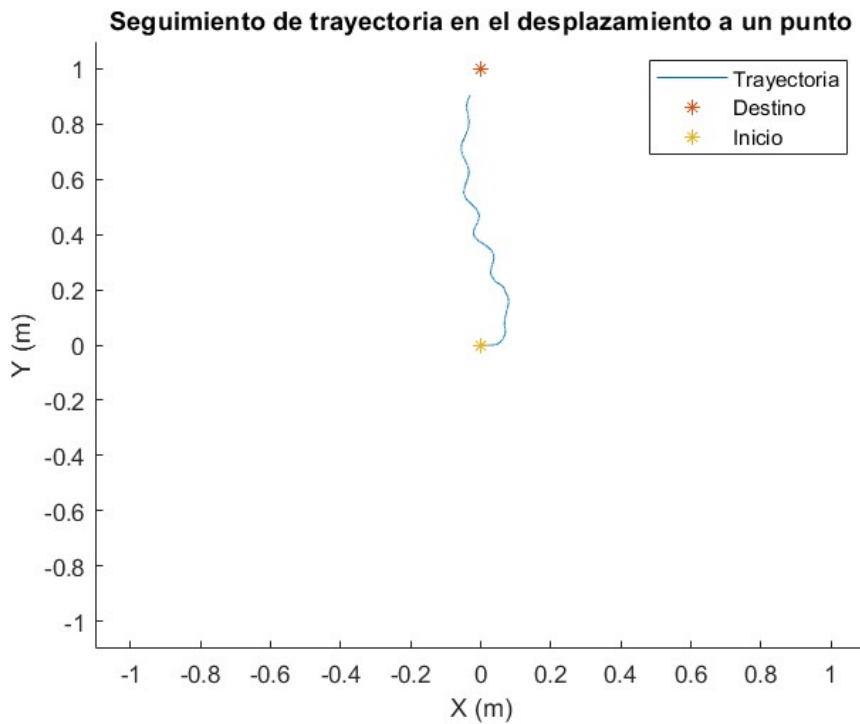
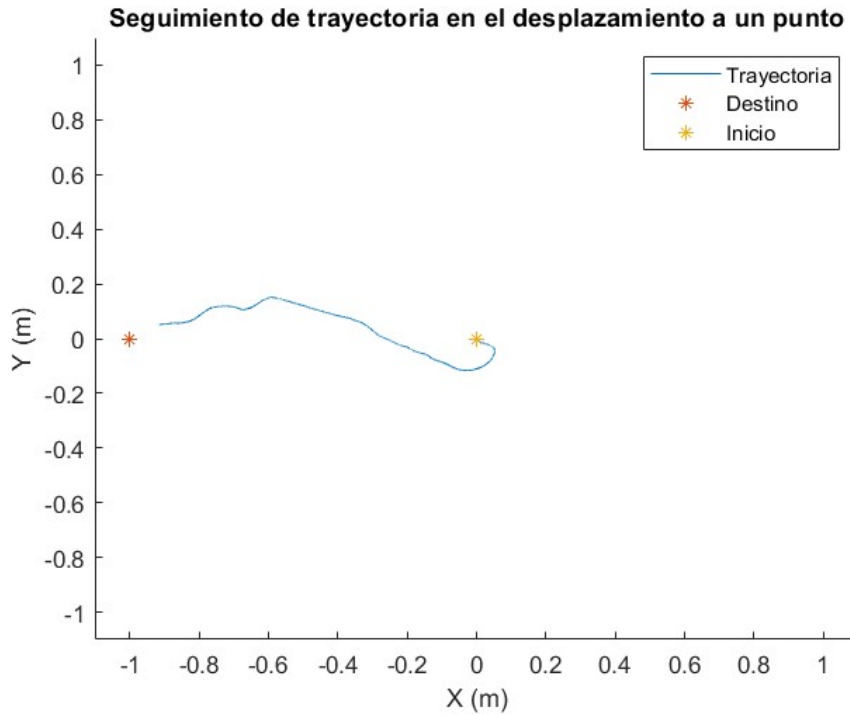
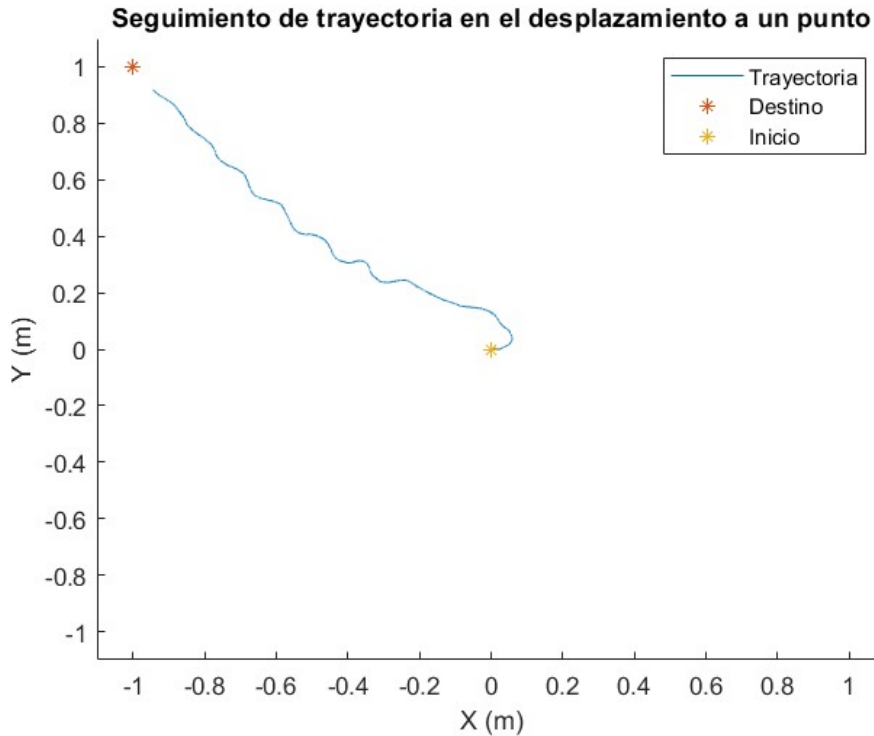


Gráfico 2. Trayectoria seguida en el desplazamiento al punto (0,1).

El tercer punto objetivo tiene coordenadas (-1,0). En el Gráfico 3 se observa la trayectoria obtenida.



El último punto objetivo tiene coordenadas (-1,1). En el Gráfico 4 se aprecia la trayectoria descrita.



En el Gráfico 1, el Gráfico 2, el Gráfico 3 y el Gráfico 4 se puede observar cómo, en todos los casos, el robot, gracias al algoritmo implementado, corrige su orientación hasta

situarse frente al punto y avanza hasta alcanzarlo. También se aprecia como durante su avance se corrige nuevamente la orientación en caso de desviarse.

5.3.6. Implementación de la simulación del seguimiento de trayectorias

En este apartado se explica el esquema creado en *Simulink* para simular el seguimiento de trayectorias. Es idéntico al indicado en el apartado 5.3.4 con una salvedad. En este caso no se trata de un punto de destino, sino de una sucesión de puntos objetivo que conforman la trayectoria que se ha de seguir. Por esta razón, ahora, al alcanzar un punto objetivo no finalizará la simulación, sino que habrá un nuevo punto de destino al que llegar y así sucesivamente hasta completar el trayecto. La Figura 92 muestra el esquema en Simulink.

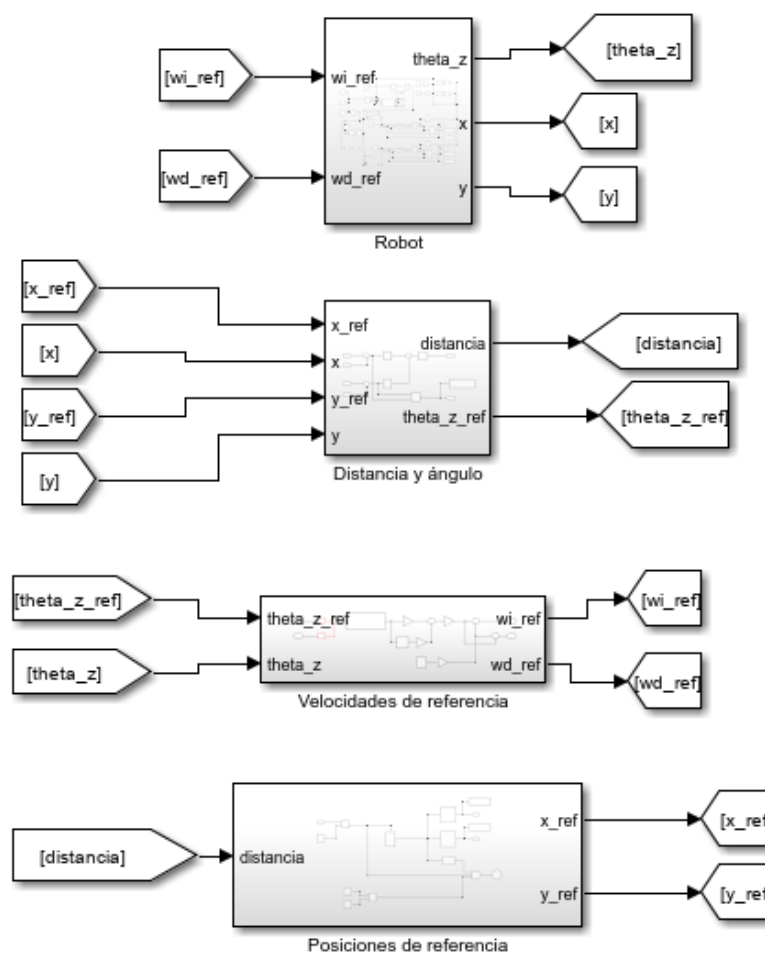


Figura 92. Esquema en Simulink para el seguimiento de trayectorias.

Los bloques “Velocidades de referencia”, “Distancia y ángulo” y “Robot” son los mismos que en el caso del desplazamiento a un punto del apartado 5.3.4, si bien es cierto que en “Robot” se ha retirado el cilindro y vuelto a colocar el bloque *Spline* como en el apartado 5.3.1.

Se ha añadido un nuevo subsistema llamado “Posiciones de referencia”. En su interior se halla el esquema visible en la Figura 93, en el que, en primer lugar, se compara la distancia entre el robot y el punto objetivo con un valor de tolerancia r . Cuando la distancia es inferior

a este umbral, el subsistema “Contador” recibe un pulso, que se traduce como un incremento de una unidad a su salida, ya que, en definitiva, cuenta los pulsos recibidos. El valor de esta cuenta se envía a dos *Look-up tables* que contienen las coordenadas x e y respectivamente de todos los puntos de la trayectoria. Las salidas de las *Look-up tables* son las coordenadas del punto de referencia actual, que pasarán a ser las del siguiente punto cuando se incremente el contador, esto es, cuando la distancia del robot al punto objetivo sea menor que r .

La simulación acaba cuando se dan simultáneamente tres condiciones, de ahí la puerta AND de tres entradas en la Figura 93. La primera de ellas es que la distancia al punto objetivo sea menor que la tolerancia. La segunda es que el punto objetivo de ese momento sea el último de la trayectoria, lo cual, siendo np el número de puntos de la trayectoria, se da cuando el contador vale np menos una unidad. La última condición es que se desee dar una única vuelta, pues en caso contrario tras el último punto volverá a comenzar por el primero y la simulación no se interrumpirá.

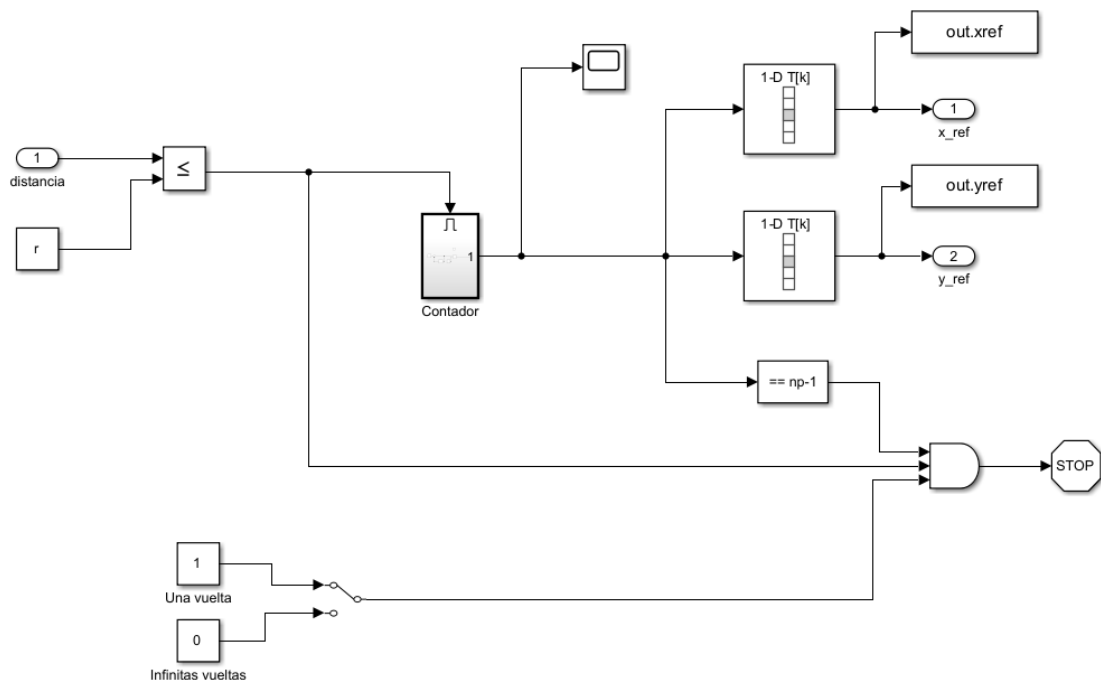


Figura 93. Contenido del subsistema “Posiciones de referencia” del esquema en Simulink para el seguimiento de trayectorias.

Por último, resta explicar el subsistema “Contador”, cuyo esquema interno aparece en la Figura 94. Primeramente, se requiere un bloque *Enable* para que solo actúe al recibir un pulso en la entrada. Cuando esto sucede se suma una unidad al valor anterior, para lo cual se usa un retardo. La salida es el resto de la división entera del contador entre el valor total de puntos en la trayectoria, de forma tal que, si se selecciona la opción de vueltas infinitas, al llegar el contador a np la salida vuelve a cero y se reinicia la cuenta.

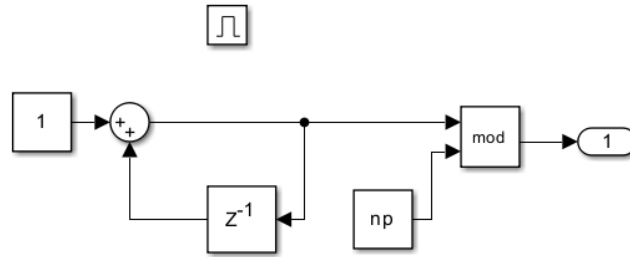


Figura 94. Contenido del subsistema "Contador" dentro del subsistema "Posiciones de referencia" del esquema en Simulink para el seguimiento de trayectorias.

Por lo que al entorno de simulación de *Simscape Multibody* se refiere, cabe destacar que se ha configurado una visualización con tres pantallas. En una de ellas, situada a la izquierda, se ve una vista isométrica del entorno, en otra, en la parte inferior derecha, se ve una vista de planta y en la tercera, arriba a la derecha, se visualiza una vista desde una cámara virtual móvil situada 1200 mm por detrás y 300 mm por encima del robot. Para esto último se ha añadido una transformación rígida unida al bloque del vehículo dentro del subsistema "Robot" y se ha tomado la salida de la transformación como referencia para la cámara. En la Figura 95 se puede observar la configuración elegida y el robot moviéndose sobre una trayectoria circular.

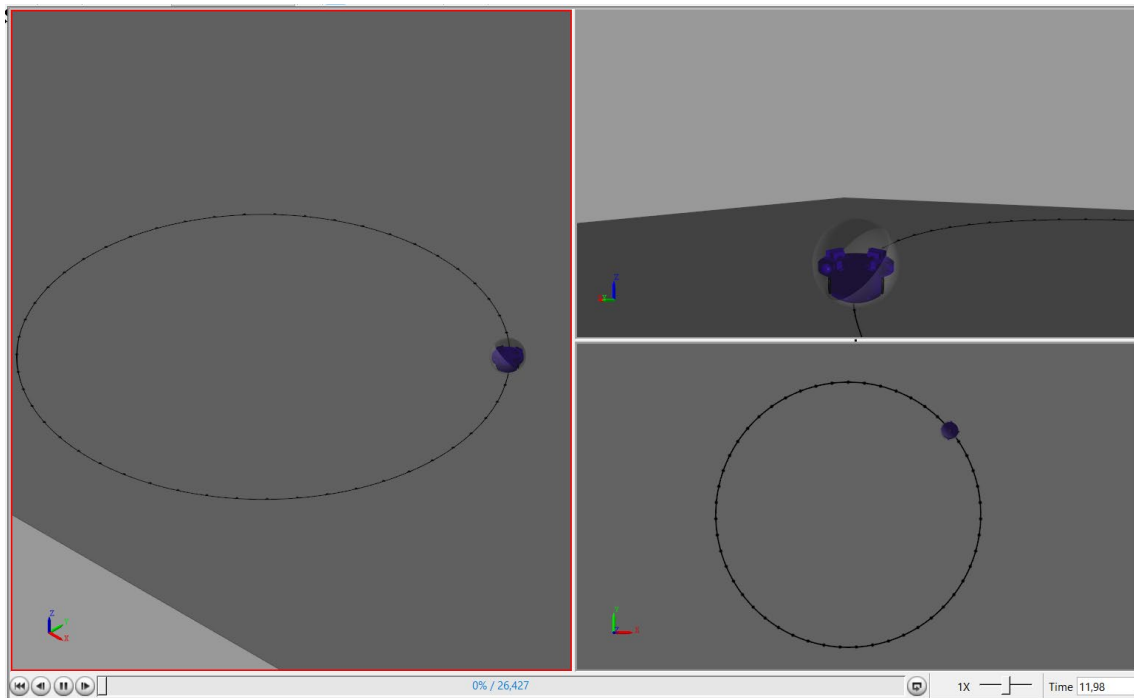


Figura 95. Configuración del entorno de simulación de *Simscape Multibody*.

5.4. Resultados de la simulación

En este apartado se exponen los resultados obtenidos en la simulación del seguimiento de trayectorias. Mediante un script de MATLAB se han generado vectores de puntos para tres trayectorias distintas.

La primera trayectoria tiene forma circular, tal y como se aprecia en la Figura 95, en el apartado 5.3. Los resultados obtenidos para esta trayectoria se encuentran a continuación en forma de gráficos. Primeramente, el Gráfico 5 y el Gráfico 6 muestran el seguimiento de la velocidad de referencia en las ruedas izquierda y derecha respectivamente. A continuación, en el Gráfico 7 y el Gráfico 8 se visualiza el seguimiento de la referencia de las coordenadas x e y del robot. Seguidamente, en el Gráfico 9 se puede ver el seguimiento de la orientación de referencia y, por último, en el Gráfico 10 se observa la trayectoria descrita, junto con la trayectoria deseada.

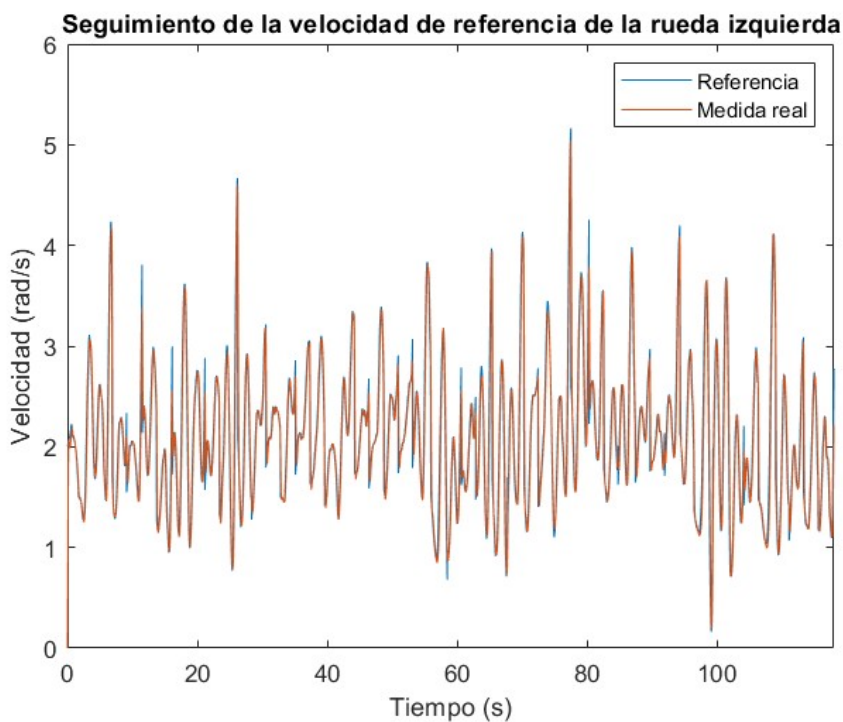


Gráfico 5. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria circular.

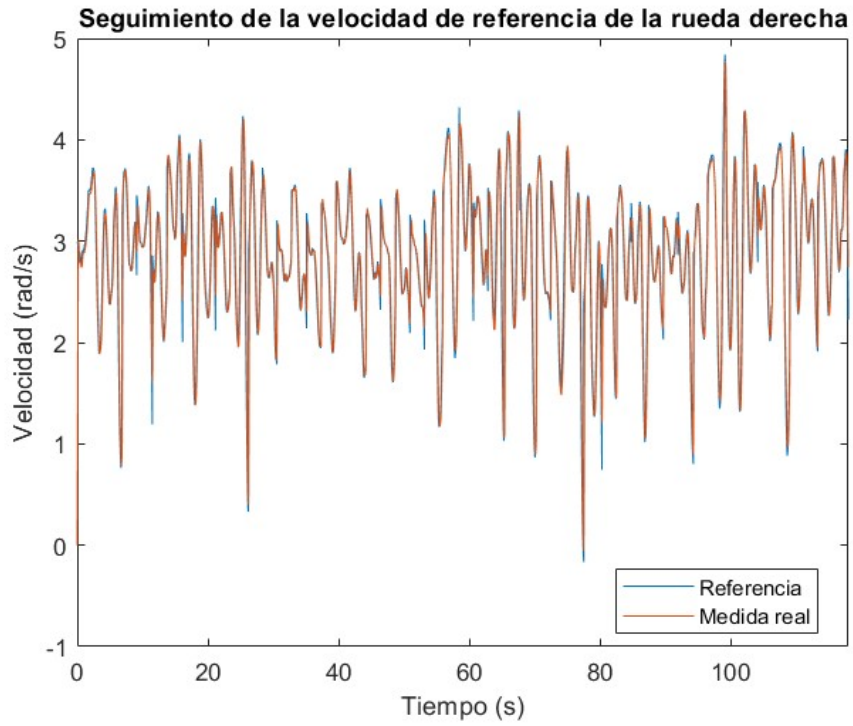


Gráfico 6. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria circular.

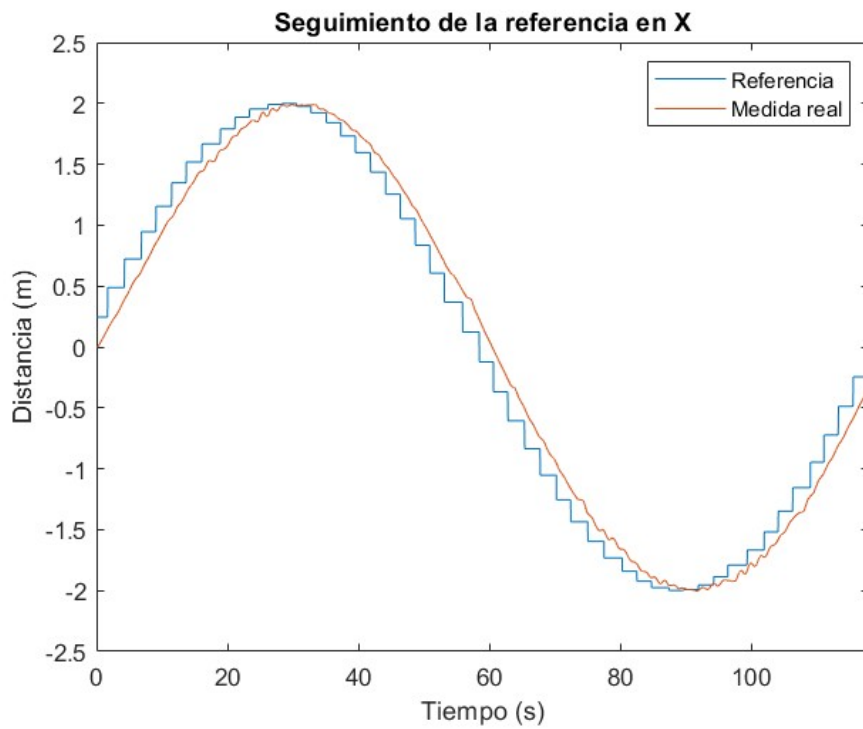


Gráfico 7. Seguimiento de la referencia de la coordenada X para una trayectoria circular.

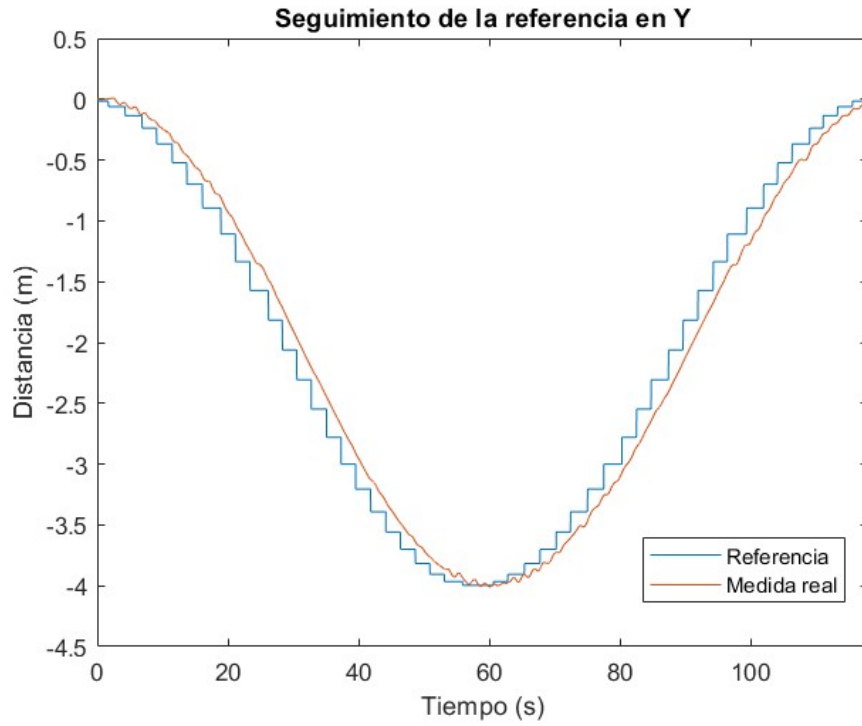


Gráfico 8. Seguimiento de la referencia de la coordenada Y para una trayectoria circular.

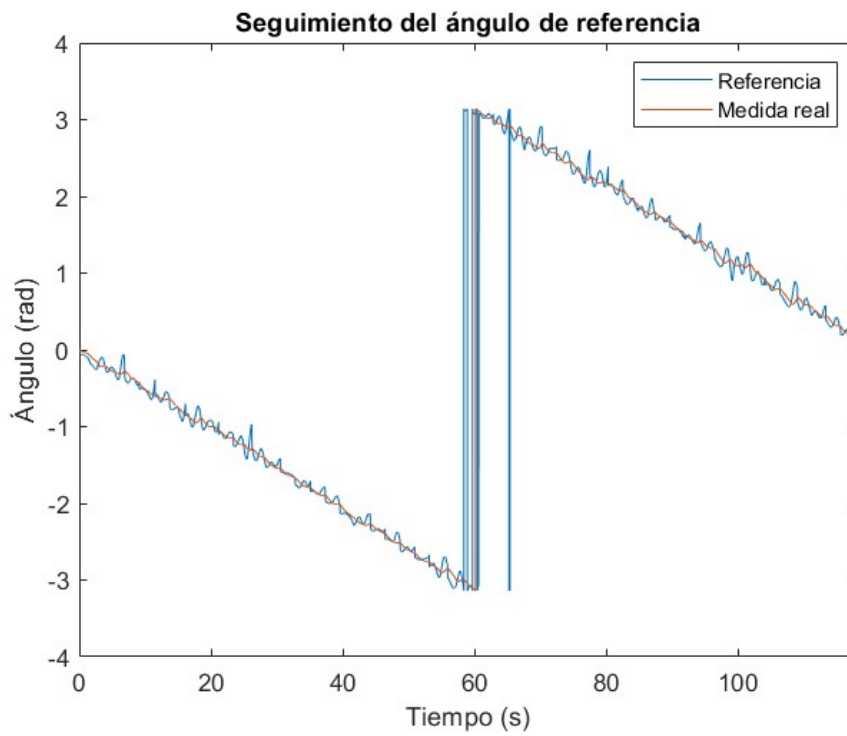


Gráfico 9. Seguimiento de la referencia de la orientación para una trayectoria circular.

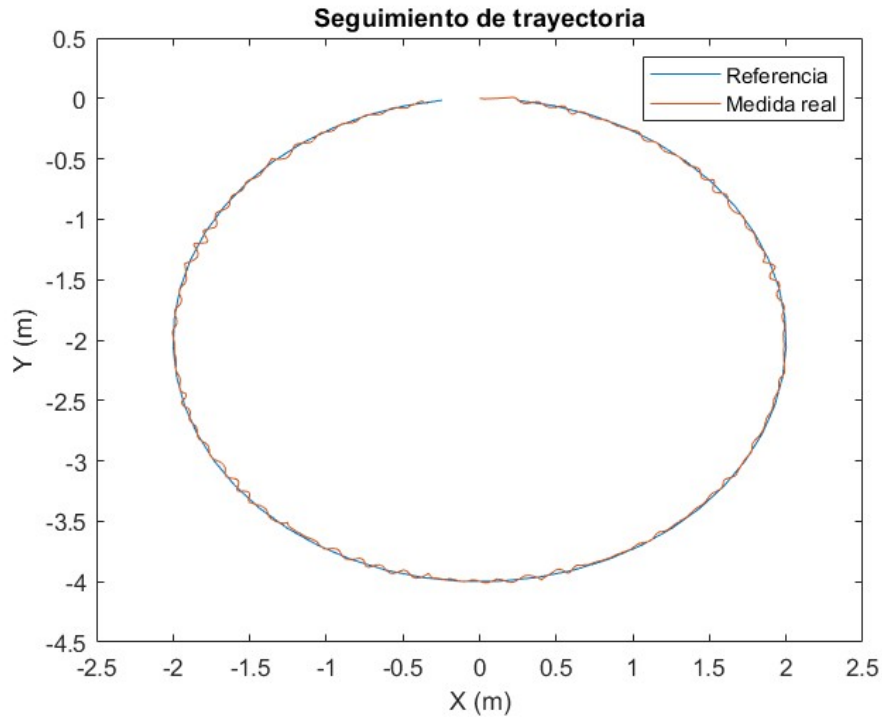


Gráfico 10. Seguimiento de una trayectoria circular.

La segunda trayectoria tiene la forma del símbolo del infinito. Los resultados obtenidos para esta trayectoria se encuentran a continuación en forma de gráficos. Primeramente, el Gráfico 11 y el Gráfico 12 muestran el seguimiento de la velocidad de referencia en las ruedas izquierda y derecha respectivamente. A continuación, en el Gráfico 13 y el Gráfico 14 se visualiza el seguimiento de la referencia de las coordenadas x e y del robot. Seguidamente, en el Gráfico 15 se puede ver el seguimiento de la orientación de referencia y , por último, en el Gráfico 16 se observa la trayectoria descrita, junto con la trayectoria deseada.

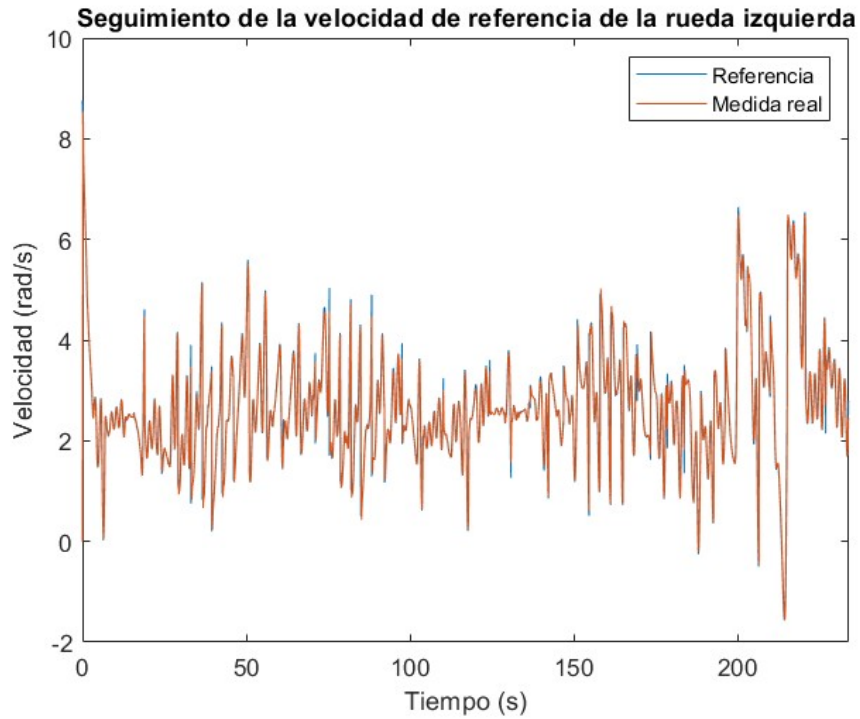


Gráfico 11. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria con forma del símbolo del infinito.

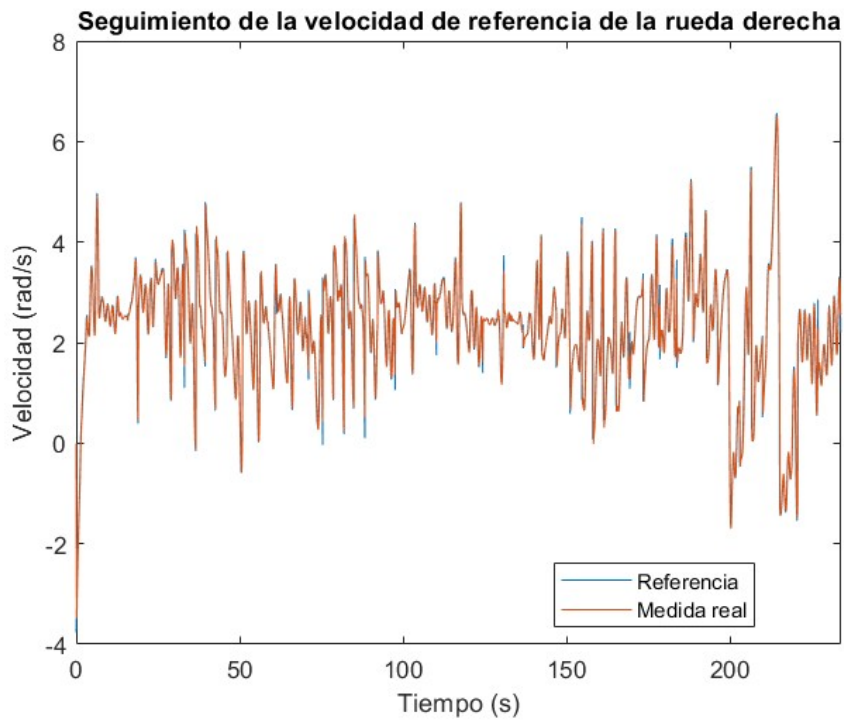


Gráfico 12. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria con forma del símbolo del infinito.

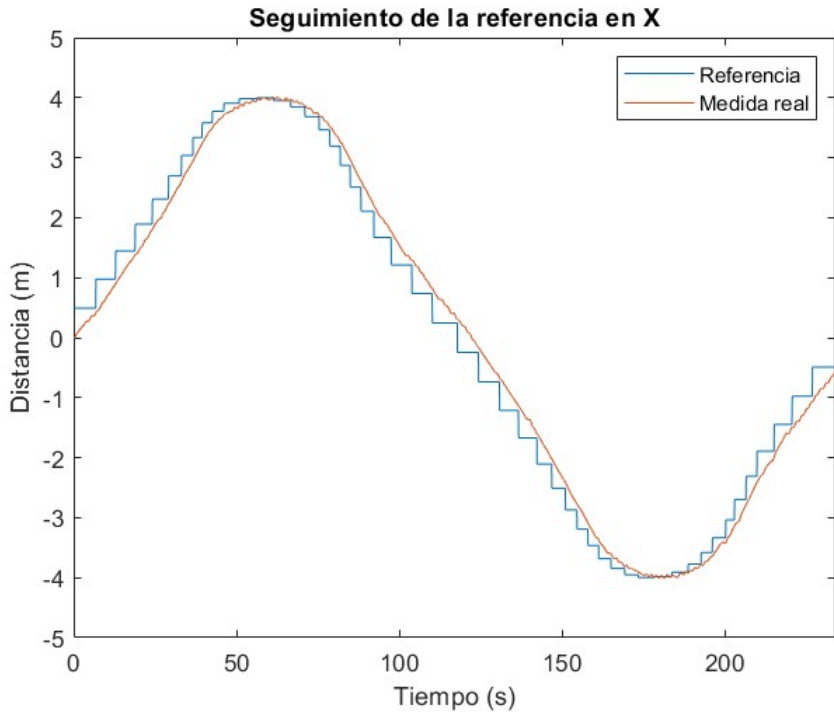


Gráfico 13. Seguimiento de la referencia de la coordenada X para una trayectoria con forma del símbolo del infinito.

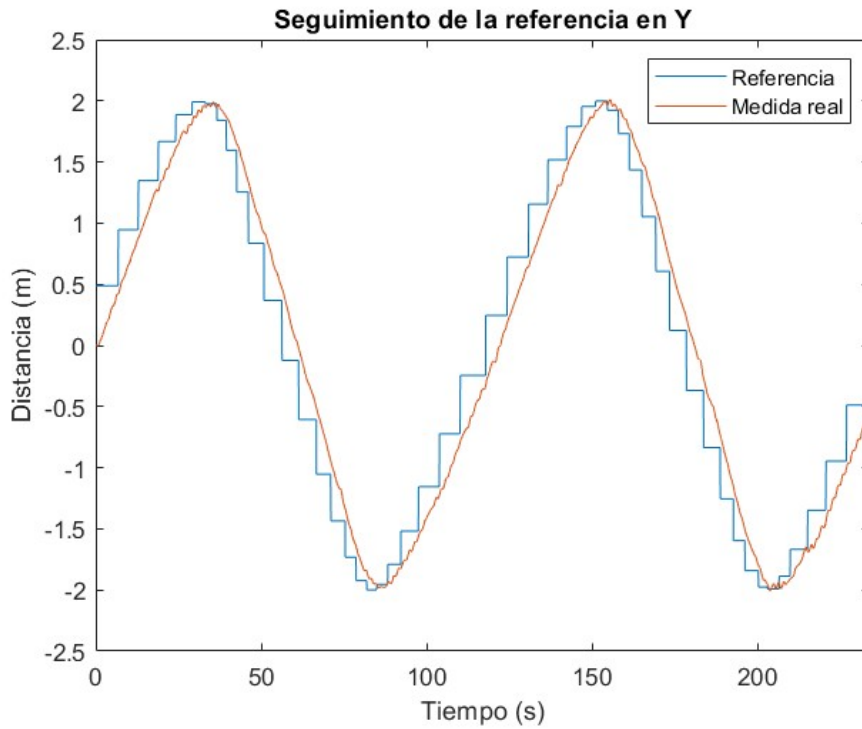


Gráfico 14. Seguimiento de la referencia de la coordenada Y para una trayectoria con forma del símbolo del infinito.

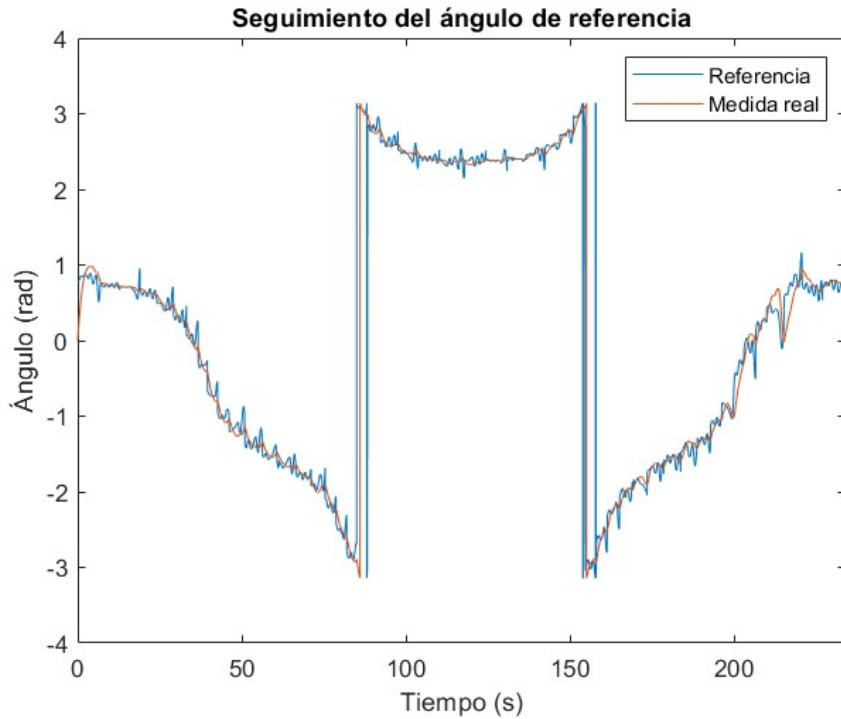


Gráfico 15. Seguimiento de la referencia de la orientación para una trayectoria con forma del símbolo del infinito.

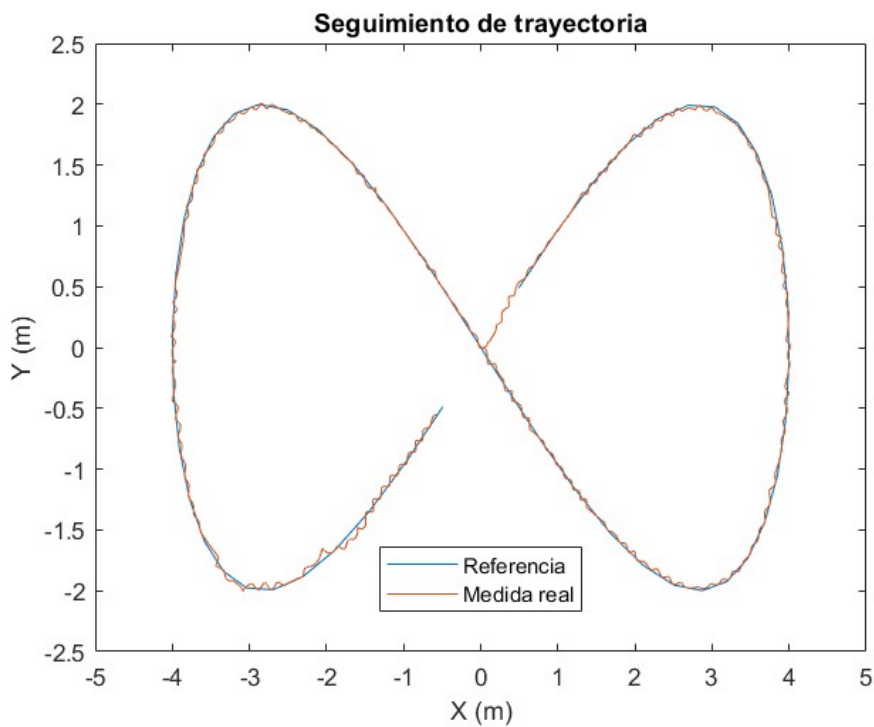


Gráfico 16. Seguimiento de una trayectoria con forma del símbolo del infinito.

Para acabar, la tercera y última trayectoria es cuadrada. Los resultados obtenidos para esta trayectoria se encuentran a continuación en forma de gráficos. Primeramente, el Gráfico 17 y el Gráfico 18 muestran el seguimiento de la velocidad de referencia en las ruedas izquierda y derecha respectivamente. A continuación, en el Gráfico 19 y el Gráfico 20 se visualiza el seguimiento de la referencia de las coordenadas x e y del robot. Seguidamente,

en el Gráfico 21 se puede ver el seguimiento de la orientación de referencia y, por último, en el Gráfico 22 se observa la trayectoria descrita, junto con la trayectoria deseada.

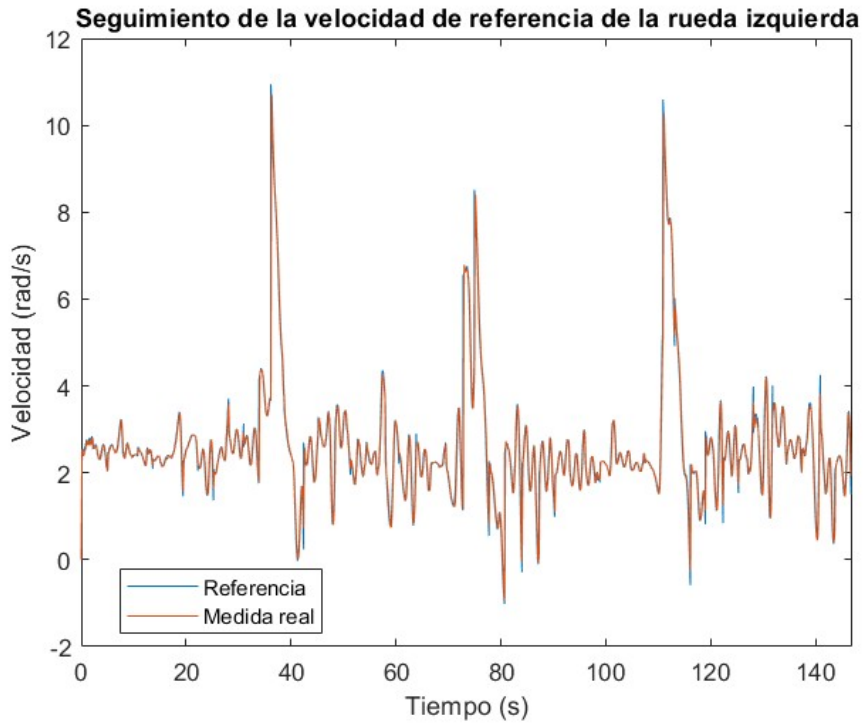


Gráfico 17. Seguimiento de la velocidad de referencia de la rueda izquierda para una trayectoria cuadrada.

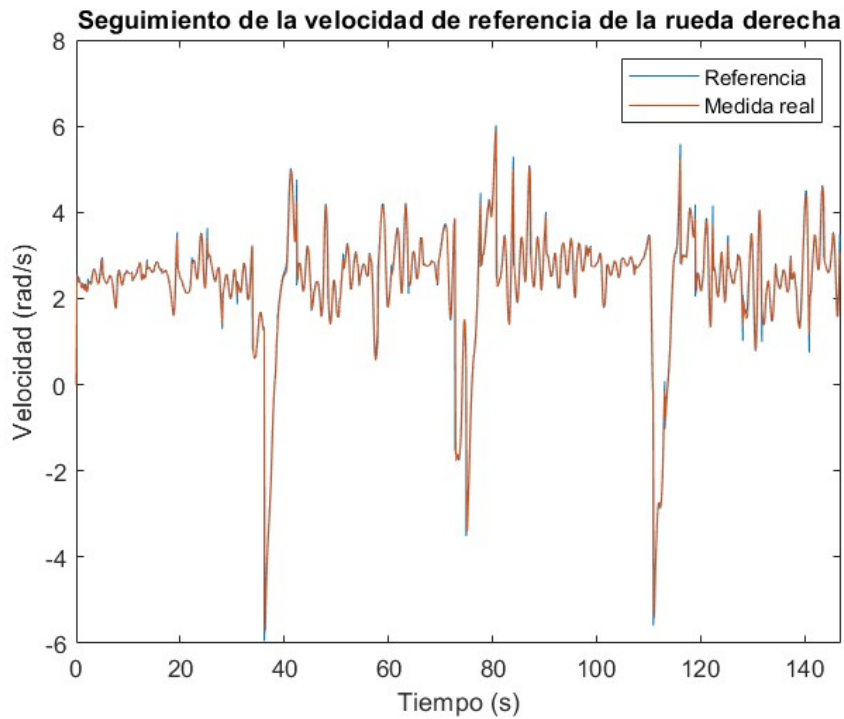


Gráfico 18. Seguimiento de la velocidad de referencia de la rueda derecha para una trayectoria cuadrada.

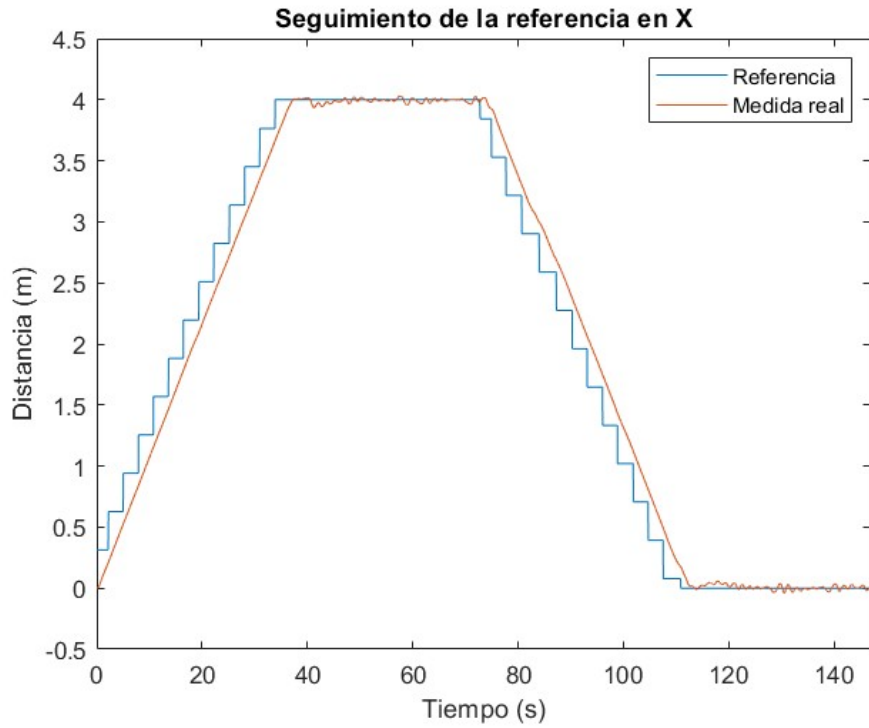


Gráfico 19. Seguimiento de la referencia de la coordenada X para una trayectoria cuadrada.

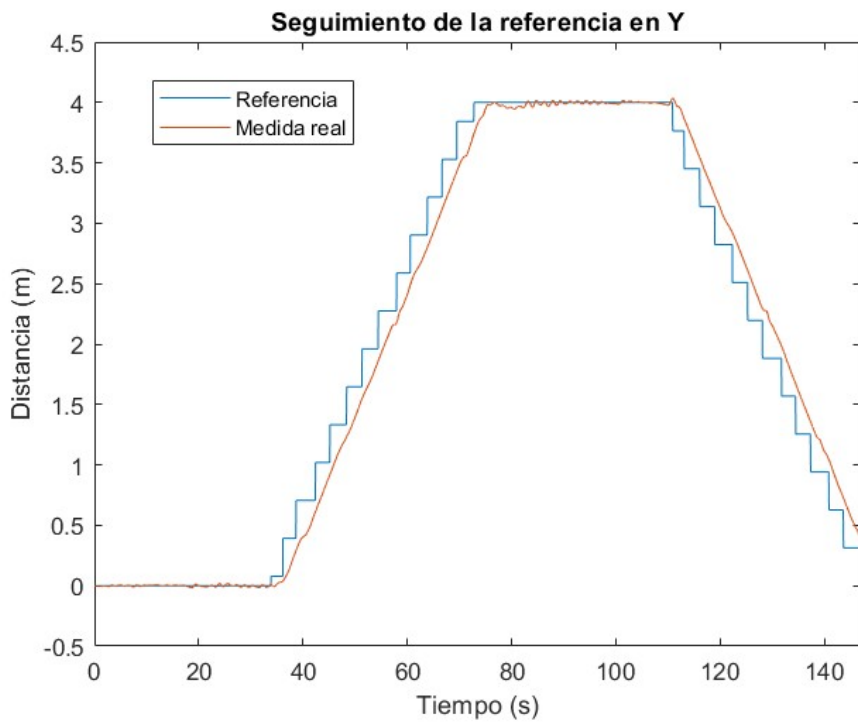


Gráfico 20. Seguimiento de la referencia de la coordenada Y para una trayectoria cuadrada.

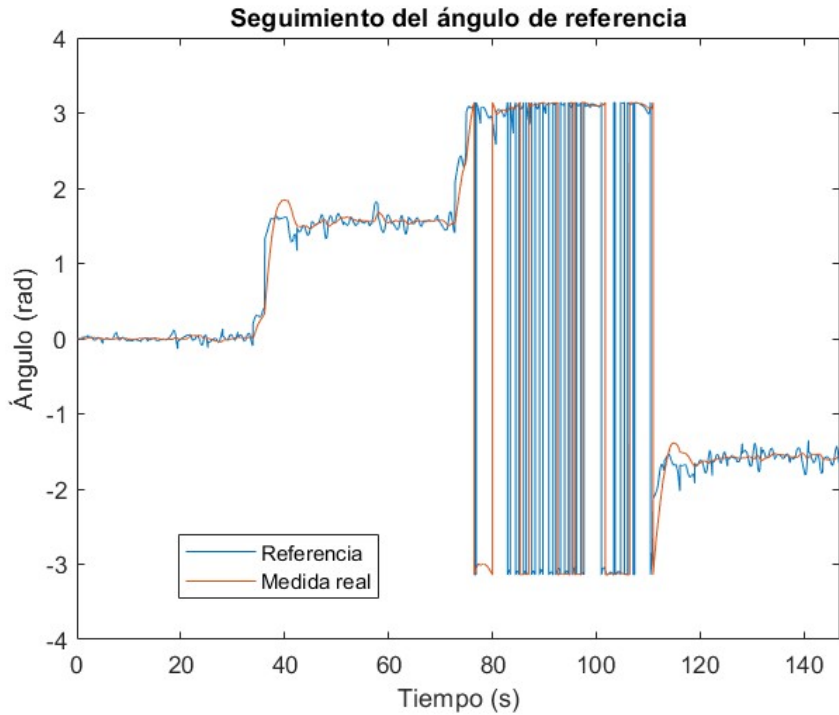


Gráfico 21. Seguimiento de la referencia de la orientación para una trayectoria cuadrada.

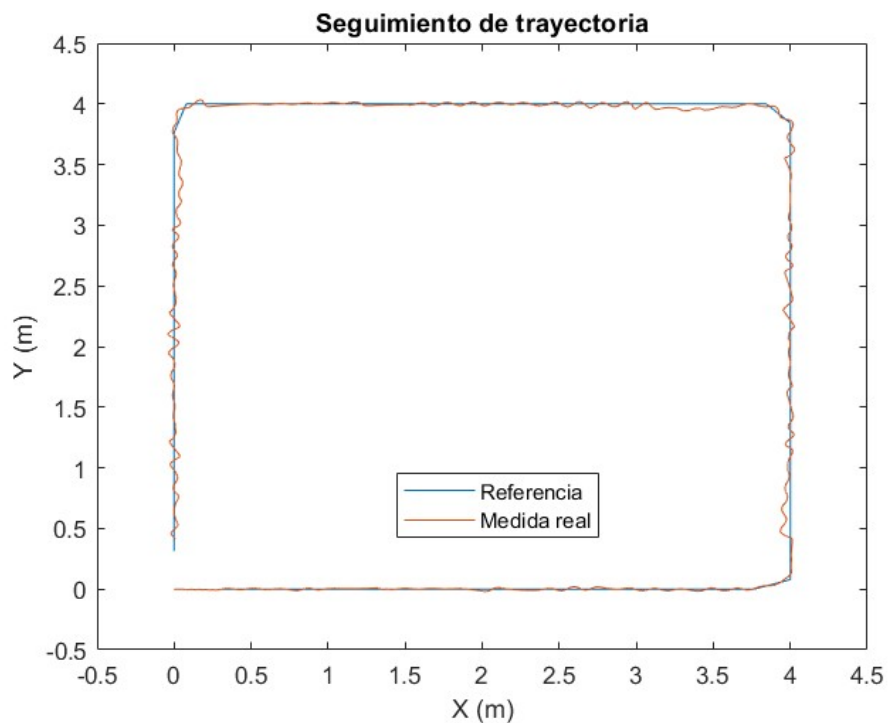


Gráfico 22. Seguimiento de una trayectoria cuadrada.

A la vista de los resultados mostrados en los gráficos del Gráfico 5 al Gráfico 22, cabe resaltar que el seguimiento de las velocidades de referencia (Gráfico 5, Gráfico 6, Gráfico 11, Gráfico 12, Gráfico 17 y Gráfico 18) es aceptable, pues no se observan discrepancias significativas entre la referencia y los valores medidos. Resultan llamativos los picos de velocidad observados en la trayectoria cuadrada (Gráfico 17 y Gráfico 18), que son debidos

a los giros bruscos necesarios para seguir la trayectoria en los vértices del cuadrado. El seguimiento de la referencia en las coordenadas x e y (Gráfico 7, Gráfico 8, Gráfico 13, Gráfico 14, Gráfico 19 y Gráfico 20) también es correcto. Nótese que la referencia es escalonada debido a que la trayectoria se ha tomado como una sucesión de puntos con una cierta distancia entre ellos. El seguimiento de la orientación (Gráfico 9, Gráfico 15 y Gráfico 21) es igualmente válido, pues la medida se ajusta con bastante precisión a la referencia, si bien es cierto que esta oscila ligeramente. Los saltos que se aprecian en el seguimiento de la orientación cerca de los valores de π y $-\pi$ son debidos simplemente a los límites del ángulo, pues al sobrepasar el valor $-\pi$ pasa a π y viceversa. Finalmente, al observar el Gráfico 10, el Gráfico 16 y el Gráfico 22, que muestran el seguimiento de la trayectoria, que era el objetivo perseguido en esta simulación, se puede concluir que el robot sigue la trayectoria programada con una precisión aceptable.

6. Implementación real

En este apartado se explica la construcción y programación del robot diseñado.

6.1. Componentes electrónicos

Además de las piezas y componentes mecánicos detallados en el apartado 4, son necesarios los componentes electrónicos que se indican a continuación.

6.1.1. Motores con encoder

Los motores utilizados son del modelo 37Dx70L de Pololu. Se trata de motores de corriente continua de alimentación máxima de 12 V. Estos motores llevan incorporada una caja reductora y existe además la opción de incorporar también un encoder. En este proyecto, el control de los motores es un control en bucle cerrado, por lo que es imprescindible contar con un encoder en cada uno de los dos motores. La Figura 96 muestra el motor seleccionado, cuyo CAD ya aparece en la Figura 36 del apartado 4.4. Consta de seis cables de diferentes colores. El rojo y el negro son los cables de alimentación del motor, el azul y verde son la alimentación (5 V y tierra) del encoder y los cables amarillo y blanco son la salida del encoder.



Figura 96. Motor Pololu 37D con encoder.

6.1.2. Driver L298N

El puente en H L298N, mostrado en la Figura 97, es un *driver* bidireccional para motores, que permite el control de dos motores de 12 V en ambos sentidos de giro con un microcontrolador alimentado a 5V.

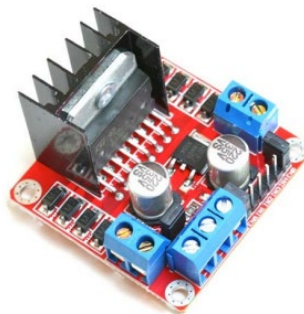


Figura 97. Puente en H L298N.

6.1.3. Batería de 12 V

Para la alimentación de los motores se ha utilizado una batería recargable de 12 V y 3000 mAh como la que se puede ver en la Figura 98.



Figura 98. Batería recargable de 12 V.

6.1.4. Microcontrolador

Naturalmente, es necesario un microcontrolador que se encargue de ejecutar el programa, recibir las órdenes de control vía Bluetooth, leer la posición de los motores mediante los encoder y enviar las acciones de control precisas a los motores.

Para este proyecto se ha optado por utilizar una placa Arduino MEGA, como la que puede verse en la Figura 99. Entre sus principales características cabe resaltar su CPU de 16 MHz y sus 256 KB de memoria. Dispone de 54 pines de entradas o salidas digitales, 15 de los cuales pueden utilizarse como salidas PWM; 16 entradas analógicas y 6 pines de interrupciones.

La principal razón por la que se ha elegido esta placa y no otras como el Arduino UNO, es que se requieren dos pines de interrupciones para cada uno de los encoder. Por ende, hacen falta cuatro pines de interrupciones como mínimo. La placa Arduino MEGA cuenta con seis pines de este tipo, tal y como se ha indicado anteriormente, mientras que otras como Arduino UNO o Arduino NANO solo cuentan con dos.



Figura 99. Placa Arduino MEGA.

6.1.5. Módulo Bluetooth

Para poder controlar el robot desde una aplicación Android es necesario un módulo Bluetooth que permita la comunicación entre ambos dispositivos. En este caso se ha elegido el módulo Bluetooth HC-06, que se observa en la Figura 100. Este módulo envía y

recibe los datos desde un dispositivo móvil a Arduino a través de los pines de comunicación USART: RX y TX.

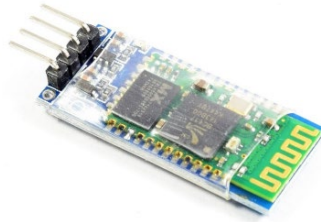


Figura 100. Módulo Bluetooth HC-06.

6.1.6. Módulo MPU-9250

Para el seguimiento de trayectorias es necesario un sensor que permita medir el movimiento y la orientación del robot. Este módulo, visible en la Figura 101, cuenta con un acelerómetro, un giroscopio y un magnetómetro, lo que permite medir con precisión las aceleraciones lineales, velocidades angulares y la orientación en los tres ejes cartesianos. Concretamente, incorpora una IMU MPU-6050, que cuenta con un acelerómetro y un giroscopio, y un magnetómetro AK8963. Se comunica con Arduino mediante el protocolo I2C.

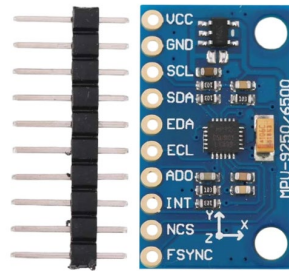


Figura 101. Módulo MPU-9250.

6.2. Montaje

Una vez impresas todas las piezas del robot y adquiridos el resto de los componentes mecánicos y electrónicos se procede a su montaje. A continuación, se indican las conexiones de la electrónica del robot y se muestra el resultado del ensamblaje de las partes mecánicas.

6.2.1. Conexiones

En este apartado se indican las conexiones realizadas entre la placa y los diferentes componentes electrónicos utilizados.

Módulo Bluetooth

El módulo Bluetooth dispone de cuatro pines: dos de alimentación, que se conectan a 5 V y a GND y dos de comunicación serie RX y TX que se conectan con los pines TX y RX de la placa de Arduino respectivamente (pines 1 y 0).

Puente en H L298N

La Figura 102 muestra el módulo L298N visto desde arriba e indicando sus pines.

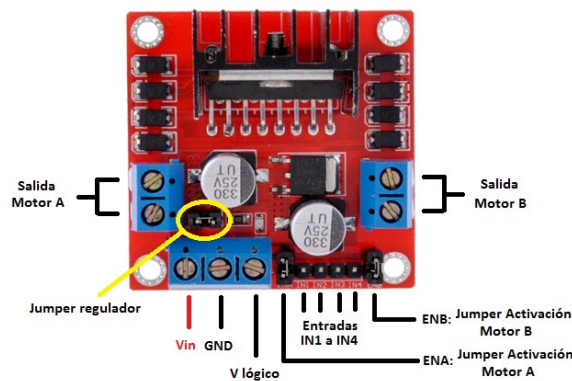


Figura 102. Pines del módulo L298N.

Los pines de alimentación Vín y GND deben conectarse a la batería de 12 V. Es importante tener en cuenta que el pin de tierra debe conectarse también al GND de la placa de Arduino. El terminal V lógico se debe conectar al pin de 5 V del controlador. Las salidas de los motores deben conectarse a los cables rojo y negro de los mismos. En cuanto a las entradas que controlan los motores, los pines IN1 e IN2 controlan el sentido de giro del motor A, mientras que IN3 e IN4 hacen lo propio con el motor B. Los pines ENA y ENB habilitan y deshabilitan el giro de los motores. Con los jumpers, permanecen a 5 V y retirando los jumpers se puede enviar desde Arduino una señal analógica que regule la velocidad de los motores. Así pues, los pines IN1, IN2, IN3 e IN4 se conectan a salidas digitales de la placa y los pines ENA y ENB, tras retirar los jumpers, a salidas PWM. Concretamente, las entradas IN1, IN2, IN3 e IN4 se han colocado en las salidas 9, 8, 5 y 4 de la placa de Arduino; y las entradas ENA y ENB a los pines 7 y 6 del controlador.

Motores

Los motores se conectan mediante sus seis pines, mostrados en la Figura 103, de la forma que se indica a continuación. Los cables rojo y negro se conectan a las salidas del puente en H, tal y como ya se ha comentado. Los cables azul y verde se conectan a 5 V y a GND

para alimentar los encoders. Por último, los cables blanco y amarillo, que envían los pulsos del encoder se conectan a pines de interrupción, para que cada cuenta del encoder genere una interrupción del programa en la que se medirá el movimiento del eje. La placa Arduino MEGA dispone de seis pines de interrupción, concretamente los pines 2, 3, 18, 19, 20 y 21. En este caso, se han elegido los cuatro primeros, utilizándose los pines 2 y 3 para el motor derecho y los pines 18 y 19 para el izquierdo.



t

Figura 103. Conector del motor Pololu 37D.

Módulo MPU-9250

El módulo MPU-9250 cuenta con 10 pines, pero para este caso, en el que se utilizará la comunicación I2C, basta con utilizar los cuatro primeros. Dos pines (Vcc y GND) se usan para alimentar el sensor, conectándose a 5 V y a tierra respectivamente. Los otros dos pines, son los que corresponden a la comunicación I2C: el pin de datos (SDA) y el del reloj (SCL). Los pines SDA y SCL de la placa Arduino Mega son respectivamente los números 20 y 21.

Diagrama de conexiones

La Figura 104 muestra un esquema en el que se aprecian las conexiones realizadas.

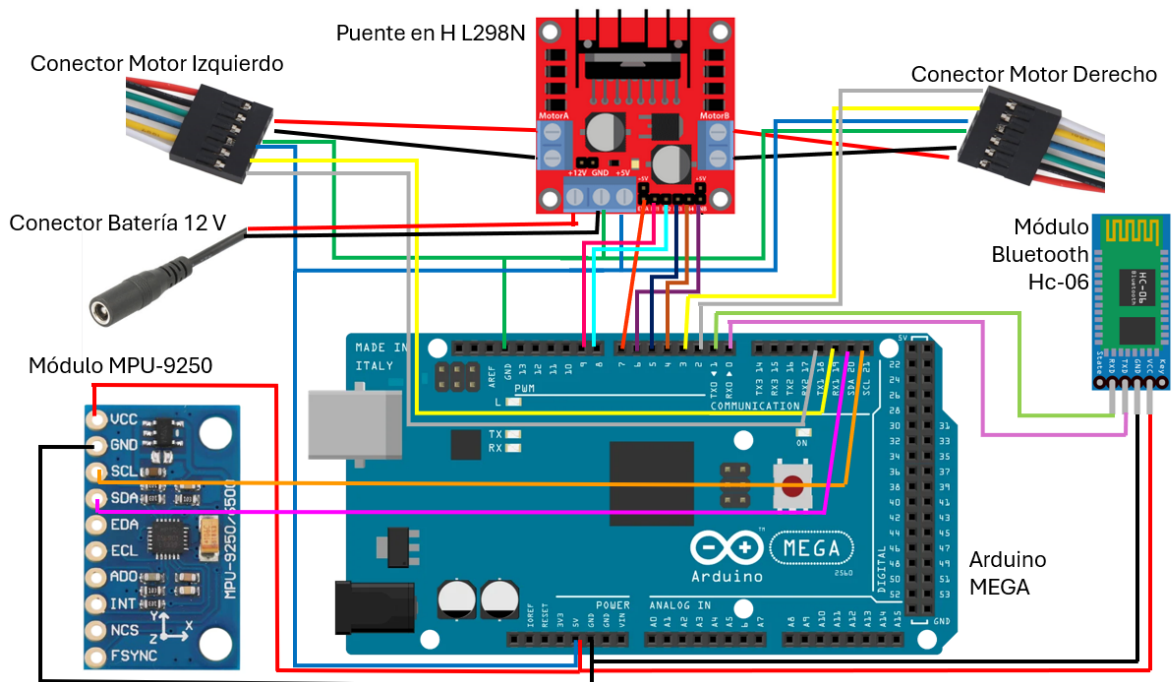


Figura 104. Diagrama de conexiones.

6.2.2. Ensamblaje mecánico

En este apartado se muestra el robot montado con todas sus piezas mecánicas ensambladas según el diseño del apartado 4. En la Figura 105 se puede observar un detalle de una de las ruedas ensamblada al motor, que a su vez está atornillado al soporte que lo sujeta al vehículo. En la Figura 106 y Figura 107 se puede observar un detalle de una de las piezas que soporta una de las ruedas esféricas, sujeta al vehículo mediante un puente y con un muelle en la parte trasera que garantiza el contacto de la rueda esférica con la esfera que envuelve al robot. Por último, en la Figura 108 se puede ver el robot completo montado, incluyendo también los elementos electrónicos. Nótese que la batería de 12 V está bajo los motores, en el interior del vehículo, tal y como se comentó en el apartado 4.3.

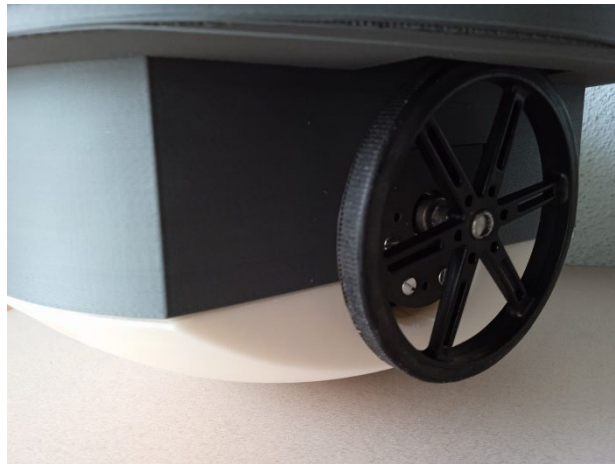


Figura 105. Detalle del ensamblaje de la rueda y el motor al vehículo.



Figura 106. Detalle de la sujeción de la rueda esférica.

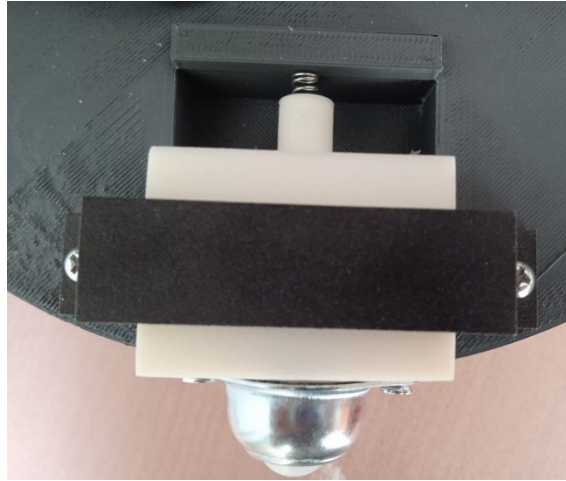


Figura 107. Detalle de la sujeción de la rueda esférica vista desde arriba.

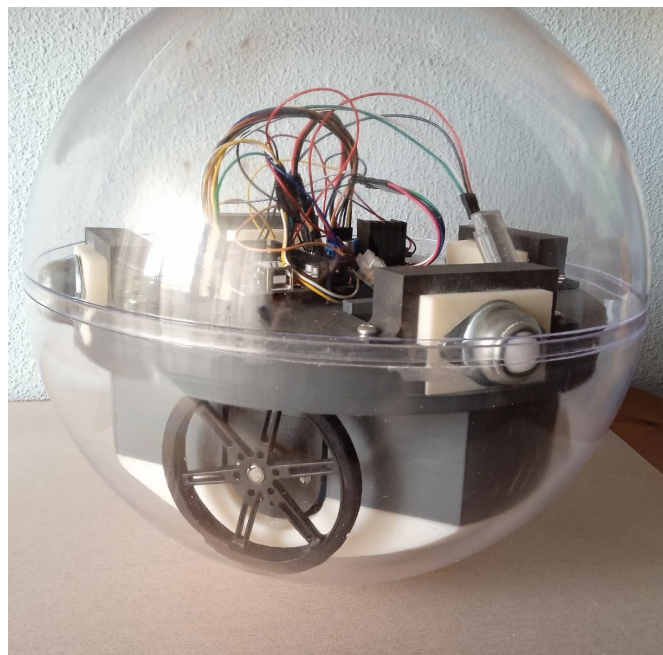


Figura 108. Prototipo de SpheroBot montado.

6.3. Programación del control remoto

Se ha desarrollado un programa en Arduino para el control del robot a través de órdenes recibidas vía Bluetooth mediante el módulo HC-06. En esencia, se envían desde un dispositivo móvil las referencias de velocidad de ambas ruedas y el programa implementado se encarga de recibirlas y de llevar a cabo un control de velocidad para ambas ruedas, que garantice que, en ambos casos, la velocidad real de los motores sea la deseada. Para esto último se utiliza un regulador tipo PI, lo cual implica cerrar el bucle de control, para lo que se usan los encoder de los motores.

En el programa desarrollado, en primer lugar, se inicializan las variables, entre las que se hallan las velocidades de referencia de ambas ruedas a cero. Seguidamente, se inicializa la comunicación serie, con una velocidad de 9600 baudios, que es la utilizada por el módulo HC-06; y se configuran las entradas y salidas, así como las interrupciones.

Dichas interrupciones se activarán cuando se produzca un cambio en alguno de los dos canales de cualquiera de los dos encoder de los motores, y llamarán a una función que incrementará o decrementará un contador para cada motor según el sentido de giro para poder así medir su posición.

En el bucle infinito (*loop*), cuando existen datos a la espera de ser leídos, se lee un carácter. La aplicación programada, que se detalla más adelante, envía instrucciones para aumentar o reducir la velocidad de cada una de las ruedas. Si el carácter recibido es “L” se aumenta en una unidad el valor de la velocidad de referencia de la rueda izquierda, y si es “l”, se reduce en una unidad. Por otra parte, si la letra es “R”, se incrementa la velocidad de referencia de la rueda derecha y se decrementa si es “r”. En todos los casos, tras aumentar o reducir la referencia correspondiente se llama a la función *Límites_ref*, que recibe como parámetro la referencia y la devuelve limitándola a unos valores máximo y mínimo. Tras esto, y también en todos los casos anteriores, se envía una trama para que la aplicación reciba el valor de las referencias y las muestre al usuario. Esta trama empieza y acaba con el carácter “*” y consta de dos parámetros. En primer lugar, una letra “D” o “I” para indicar si el valor que debe actualizarse en pantalla corresponde a la rueda derecha o izquierda, y un número, que es el valor de la referencia de la rueda (izquierda o derecha). Existe también una opción para detener el robot. En caso de recibirse el carácter “S” se ponen ambas referencias a cero y se envían dos tramas, una para cada motor, para que en la aplicación se actualice el valor de las dos ruedas, que será cero en este caso.

Una vez establecida la referencia mediante la comunicación Bluetooth, se lee el valor del encoder de cada rueda y se calcula su velocidad utilizando la diferencia entre el valor de posición actual y el anterior, y el periodo de muestreo. Una vez obtenida la velocidad se actualiza el valor de posición anterior para la siguiente iteración.

Posteriormente, se calcula para cada motor el error como la diferencia entre la referencia y el valor real de la velocidad. Se calcula también la integral del error como la acumulación del mismo, sumando el error actual al anterior en cada repetición del bucle. Con estos datos y con las constantes proporcional e integral del regulador, que se han ajustado de manera experimental, se obtiene la acción de control. Calculada la acción de control, se actualiza el valor del error integral anterior para la próxima iteración.

Luego, se limita la acción de control a unos valores límite, se establece el sentido de giro de cada uno de los motores según el signo de su acción de control y se envía dicha acción de control a través de dos salidas PWM.

Por último, se comprueba que no existe retraso respecto al periodo de muestreo, indicándose en caso contrario encendiendo el led de la placa asociado al pin 13, y se espera a que pase un periodo de muestreo completo.

A continuación, se muestra en la Figura 109, la Figura 110 y la Figura 111 el flujograma del programa utilizado para el control remoto del robot.

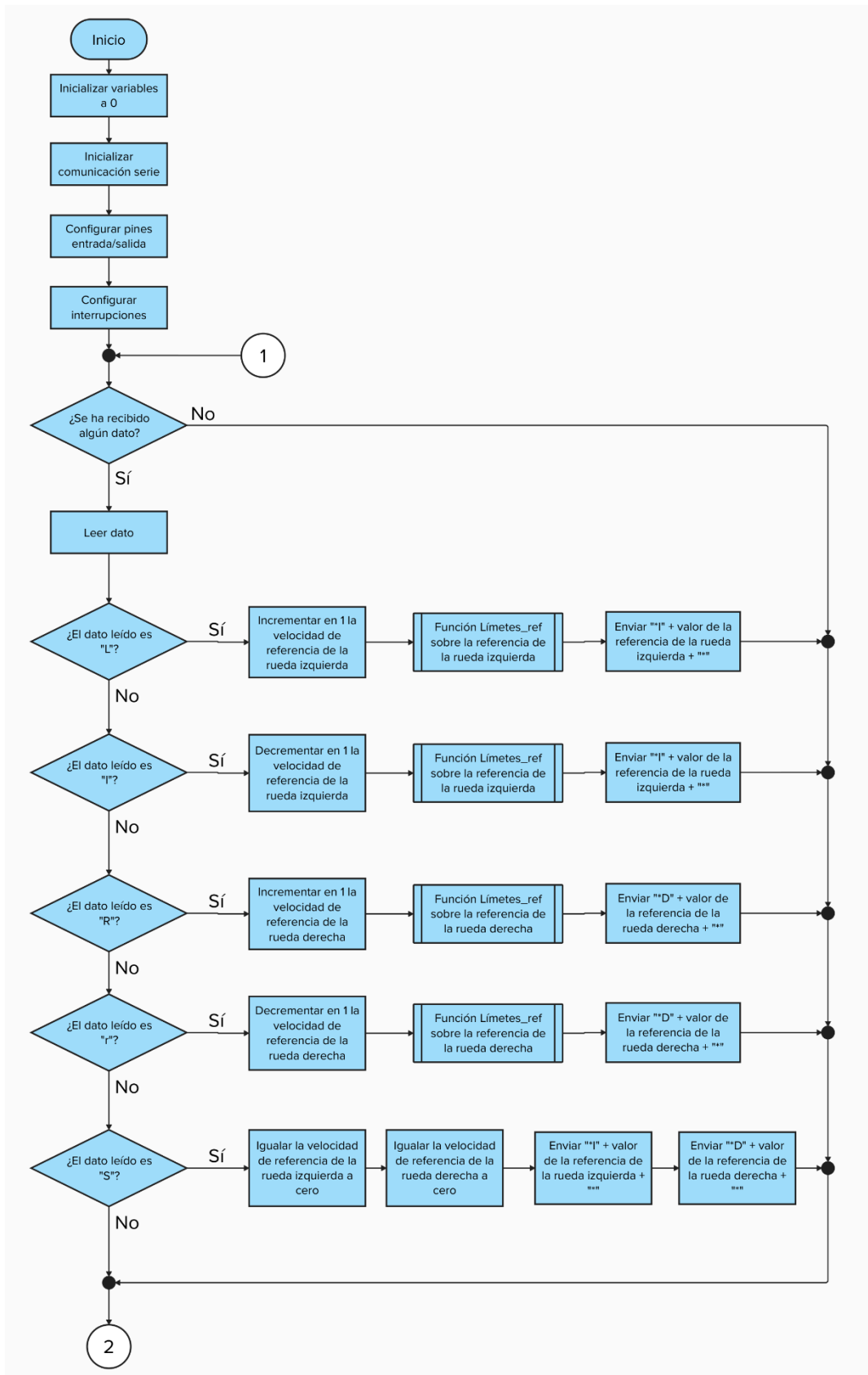


Figura 109. Flujograma del programa para el control remoto del robot (Parte 1).

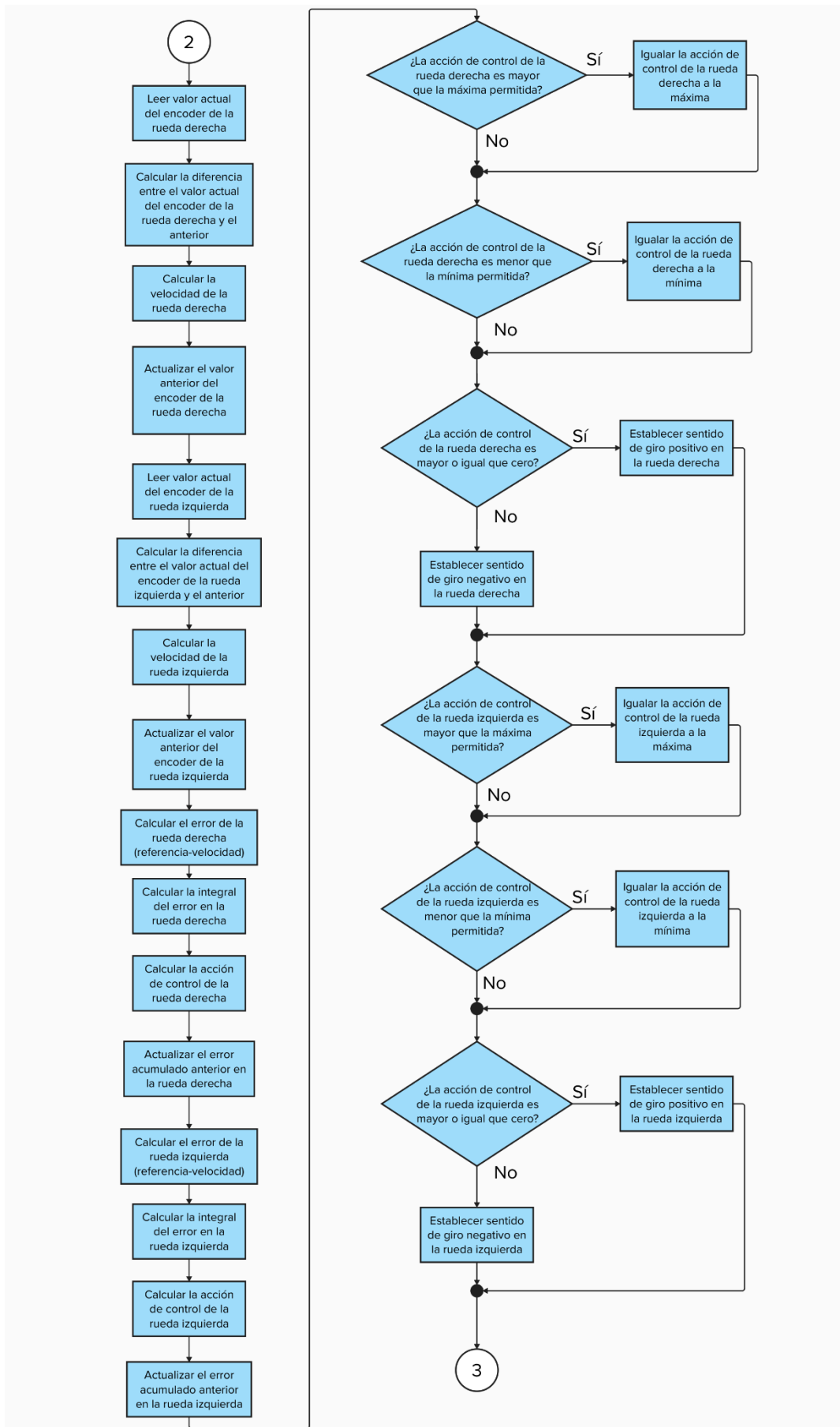


Figura 110. Flujoograma del programa para el control remoto del robot (Parte 2).

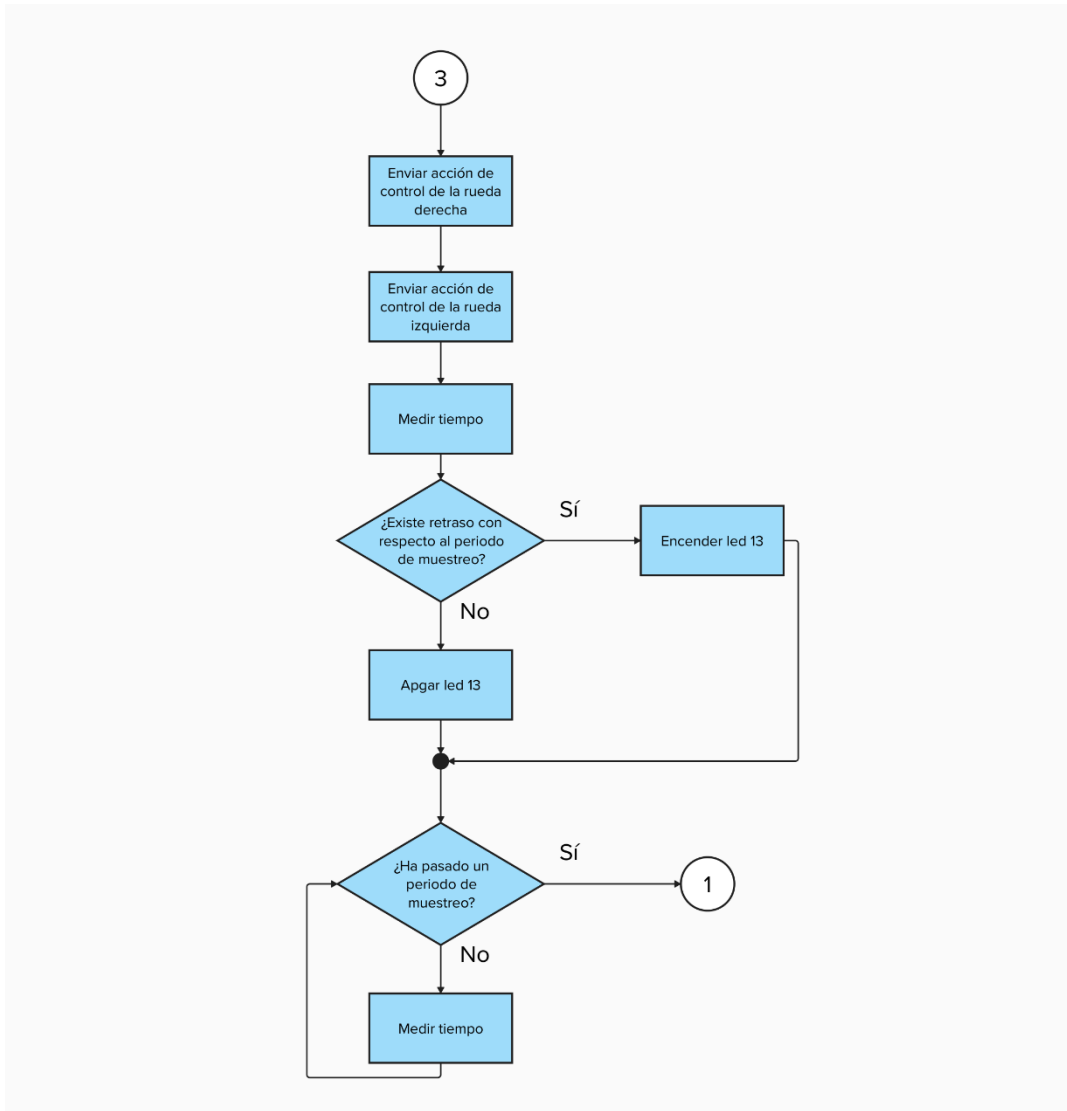


Figura 111. Flujograma del programa para el control remoto del robot (Parte 3).

El flujograma de la función Límites_ref se puede ver en la Figura 112.

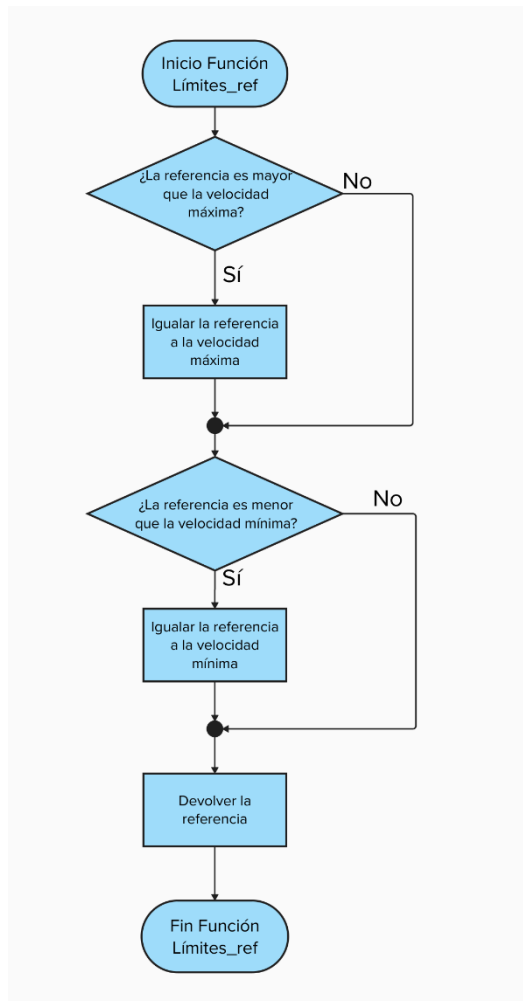


Figura 112. Flujograma de la función *Límites_ref* del programa para el control remoto del robot.

Por otra parte, la lectura de la posición de los motores para el cálculo de la velocidad se realiza mediante interrupciones, que se producen ante un cambio en los pines de los dos canales de cada uno de los encoders. Los flujogramas de las interrupciones se pueden apreciar en la Figura 113.

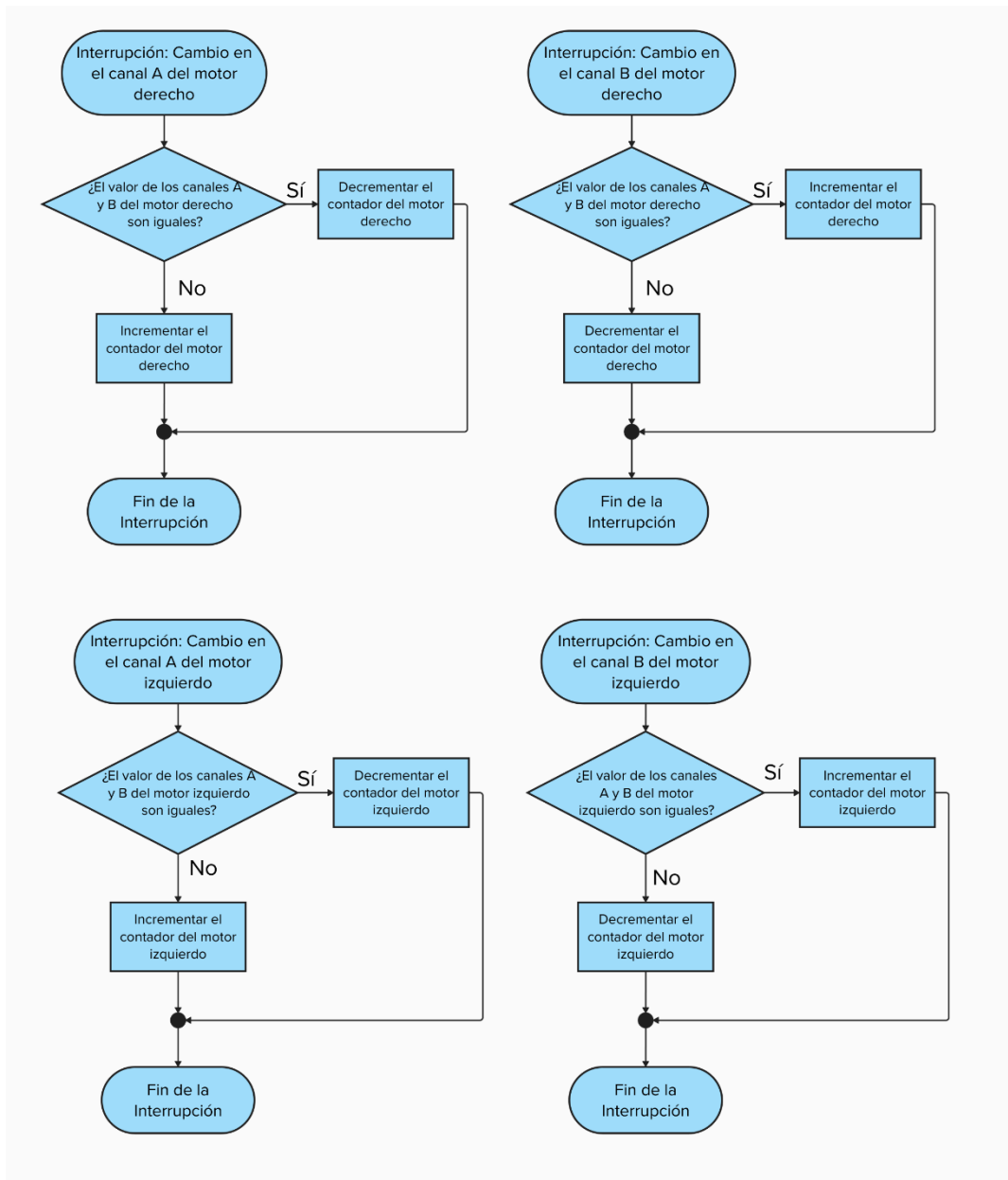


Figura 113. Flujo de las interrupciones del programa para el control remoto del robot.

6.4. Aplicación Android para control remoto

Con el fin de poder controlar remotamente el robot, se ha implementado una aplicación Android utilizando la herramienta *Bluetooth Electronics*. La aplicación, cuya interfaz puede verse en la Figura 114, consiste en dos botones por cada una de las ruedas para aumentar o reducir su velocidad. Además, hay un pulsador para detener el robot en caso de emergencia. En la parte inferior de los controles de las ruedas, se muestra el valor actual de la referencia de velocidad que se está enviando a cada una.



Figura 114. Interfaz de usuario de la aplicación de control remoto creada en *Bluetooth Electronics*.

Tal y como se ha comentado en el apartado 6.3, cada una de las flechas envía un carácter (“L”, “l”, “R” o “r”). Además, el botón rojo (STOP) envía “S”. En la Figura 115 puede observarse un ejemplo de la programación de uno de los pulsadores para enviar un carácter al ser pulsado.

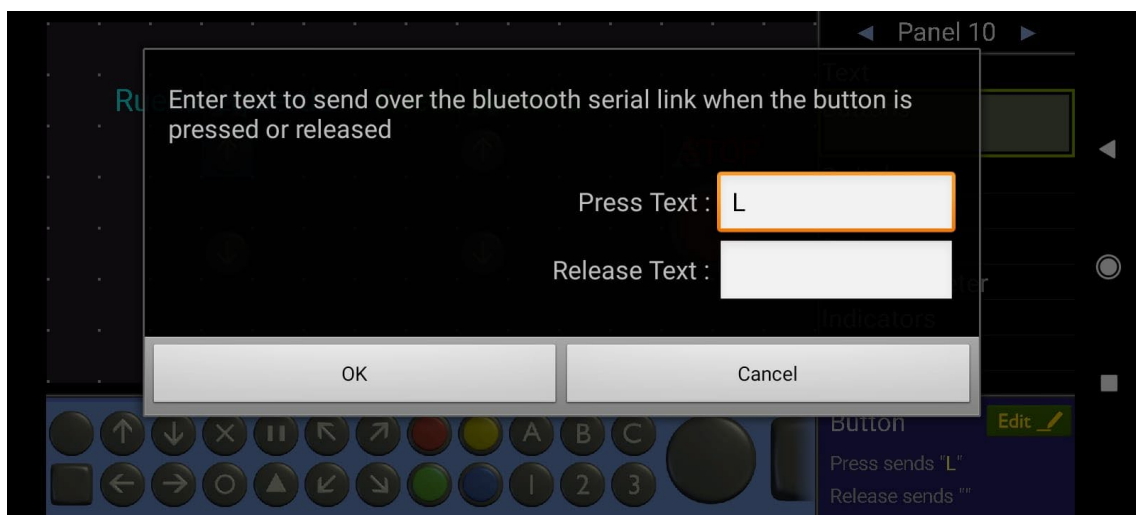


Figura 115. Programación en *Bluetooth Electronics* de un pulsador para enviar un carácter al ser pulsado.

Por otro lado, se han programado dos displays en la parte inferior para mostrar los valores de las referencias de ambas ruedas que la placa Arduino enviará por Bluetooth. En la Figura 116 se puede apreciar dicha programación, en la que se indica que el valor que se ha de

escribir irá precedido por la letra “I”. Este es el caso de la rueda izquierda. El de la rueda derecha es idéntico pero la letra que precede al valor numérico es “D”.

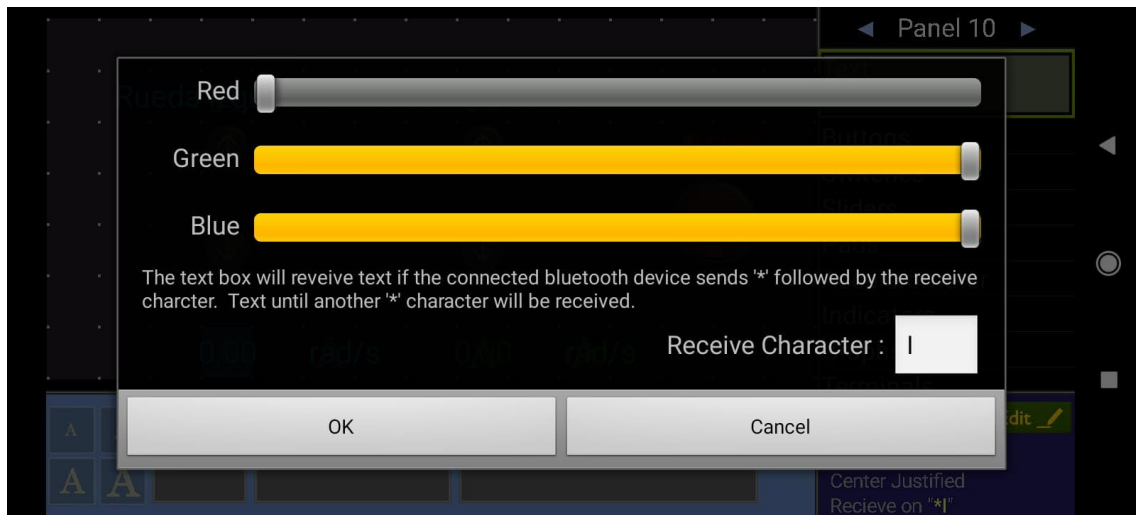


Figura 116. Programación en Bluetooth Electronics de un display para mostrar el valor de la referencia de la rueda izquierda.

6.5. Programación del seguimiento de trayectorias

Se ha desarrollado un programa en Arduino para el seguimiento de trayectorias predeterminadas por parte del robot. Además, se utiliza una aplicación Android que envía órdenes vía Bluetooth para elegir una trayectoria.

Para el seguimiento de trayectorias se ha hecho uso del módulo MPU-9250 para medir la velocidad angular de giro del robot y, a partir de ella, calcular su orientación. Conocida la orientación y las velocidades de ambas ruedas, es posible estimar la velocidad del robot y calcular por tanto su posición. Otra opción es utilizar el acelerómetro que incorpora este mismo módulo para medir la aceleración del robot y calcular su velocidad y posición. Sin embargo, el robot cabecea ligeramente durante su movimiento, lo que provoca que los ejes X e Y del sensor no siempre sean horizontales y la aceleración de la gravedad, junto con la aceleración vertical producida por el cabeceo, interfieran en la medida haciéndola inexacta. Por esta razón se ha optado por la estimación de la posición a partir de la orientación y de las velocidades de las ruedas en lugar de hacerlo a partir de la aceleración.

En el programa implementado, en primer lugar, se inicializan las variables, entre las que se hallan las velocidades de referencia de ambas ruedas a cero y la variable *booleana* “paro” que se inicializa a *TRUE*. También se inicializa a uno un contador, que se utilizará para leer los valores de dos vectores que contienen las coordenadas de los puntos que conforman la trayectoria que se debe seguir. Seguidamente, se inicializa la comunicación serie, con una velocidad de 9600 baudios, que es la utilizada por el módulo HC-06; se inicializa y configura el módulo MPU-9250 y se configuran las entradas y salidas, así como las interrupciones.

Al igual que en el programa para el control remoto, dichas interrupciones se activarán cuando se produzca un cambio en alguno de los dos canales de cualquiera de los dos encoder de los motores, y llamarán a una función que incrementará o decrementará un contador para cada motor según el sentido de giro para poder así medir su posición. Estas funciones son idénticas a las del programa anterior explicado en el apartado 6.3.

En el bucle infinito (*loop*), cuando existen datos a la espera de ser leídos, se lee un carácter. La aplicación programada, que se detalla más adelante, envía instrucciones para seleccionar una u otra trayectoria. Si el carácter leído es “S”, se ha seleccionado la opción *STOP*, y la variable “paro” se pone a *TRUE*. Si se lee “O” y “paro” está a *TRUE*, se pone el contador a uno, se envía la trama “*HC*” para limpiar el gráfico de la aplicación, como se detallará más adelante, se pone “paro” a *FALSE*, se inicializa el ángulo theta a cero y se guardan en dos vectores X e Y las coordenadas de la trayectoria circular. Lo mismo sucede si, estando “paro” a *TRUE*, se recibe el carácter “I” o “C”, pero en estos casos las trayectorias son en forma del signo del infinito o cuadrada respectivamente.

Acto seguido, se lee la velocidad angular en z mediante el módulo MPU-9250, se pasa a grados por segundo y se calcula el ángulo de orientación del robot, que posteriormente se pasa a radianes. Después, a partir de las coordenadas del robot y de las coordenadas leídas de los vectores que contienen la trayectoria se calcula la distancia y el ángulo entre ambos puntos (ángulo de referencia).

Conocido el ángulo de referencia y el ángulo medido se calcula el error de la orientación como la diferencia entre ambos, que se limita al intervalo $[-\pi, \pi]$ y se multiplica por una constante para obtener una acción de control mediante un regulador proporcional.

Si la variable “paro” es *FALSE*, se calcula la velocidad de referencia de la rueda derecha sumando a un cierto valor por defecto la acción de control de la orientación multiplicada por una constante. Para la rueda izquierda la acción de control de orientación multiplicada por la misma constante se resta al mismo valor por defecto, con lo que se logra una velocidad distinta en cada rueda ante un error en la orientación, lo que modifica el ángulo del robot corrigiendo la desviación. Después, se llama a la función *Límites_ref* con ambas referencias para limitarlas a los valores mínimos y máximos. Se ha fijado un valor de ganancia proporcional de 2, para el cálculo de la acción de control de la orientación, y una constante de 10 para multiplicar dicha acción de control y sumarla a la velocidad de una rueda y restarla a la otra. Estos valores se han fijado de manera experimental hasta lograr un resultado aceptable.

En caso de que “paro” esté a *TRUE*, las ruedas permanecen paradas.

A continuación, si la distancia entre el robot y el punto de la trayectoria que se pretende alcanzar es menor de 10 cm, se incrementa el contador, pasando así al siguiente punto. Además, en este momento, se envía una trama con las coordenadas del robot para ser representadas en la aplicación Android que se explica en el siguiente apartado.

El siguiente paso consiste en comprobar si se ha llegado al punto final de la trayectoria, en cuyo caso la variable “paro” se pone a *TRUE*.

Luego, se estima la posición del robot a partir del ángulo medido y de las velocidades de las ruedas, según el modelo explicado en el apartado 3.5.

A partir de aquí, el programa es idéntico al del control remoto del apartado 6.3. Una vez establecidas las velocidades de referencia se lee el valor del encoder de cada rueda y se calcula su velocidad utilizando la diferencia entre el valor de posición actual y el anterior, y el periodo de muestreo. Una vez obtenida la velocidad se actualiza el valor de posición anterior para la siguiente iteración.

Posteriormente, se calcula para cada motor el error como la diferencia entre la referencia y el valor real de la velocidad. Se calcula también la integral del error como la acumulación del mismo, sumando el error actual al anterior en cada repetición del bucle. Con estos datos y con las constantes proporcional e integral del regulador, ajustadas experimentalmente, se obtiene la acción de control. Calculada la acción de control, se actualiza el valor del error integral anterior para la próxima iteración.

Luego, se limita la acción de control a unos valores límite, se establece el sentido de giro de cada uno de los motores según el signo de su acción de control y se envía dicha acción de control a través de dos salidas PWM.

Por último, se comprueba que no existe retraso respecto al periodo de muestreo, indicándose en caso contrario encendiendo el led de la placa asociado al pin 13, y se espera a que pase un periodo de muestreo.

A continuación, se muestra en la Figura 117, la Figura 118, la Figura 119 y la Figura 120 el flujograma del programa utilizado para el seguimiento de trayectorias.

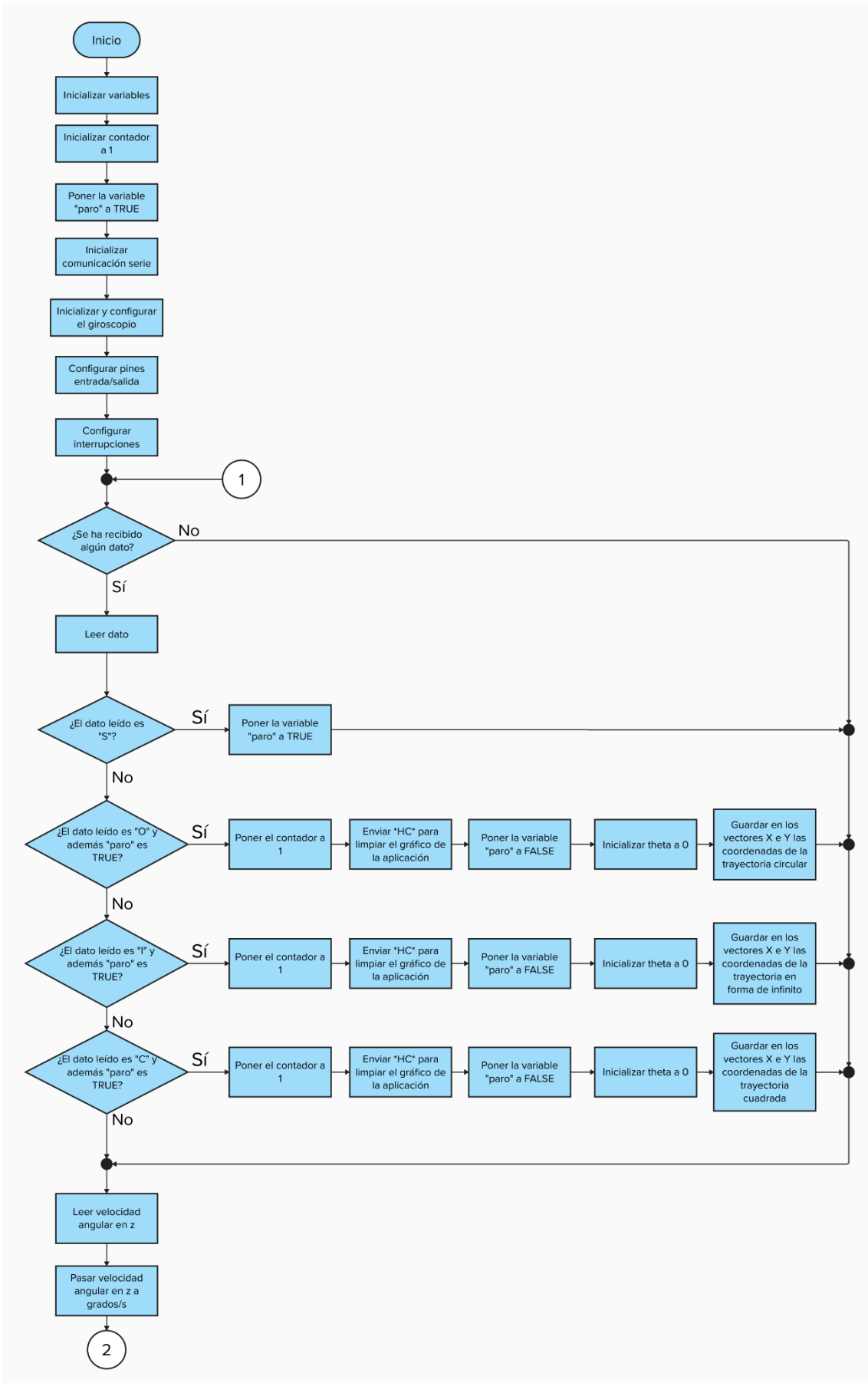


Figura 117. Flujograma del programa para el seguimiento de trayectorias (Parte 1).

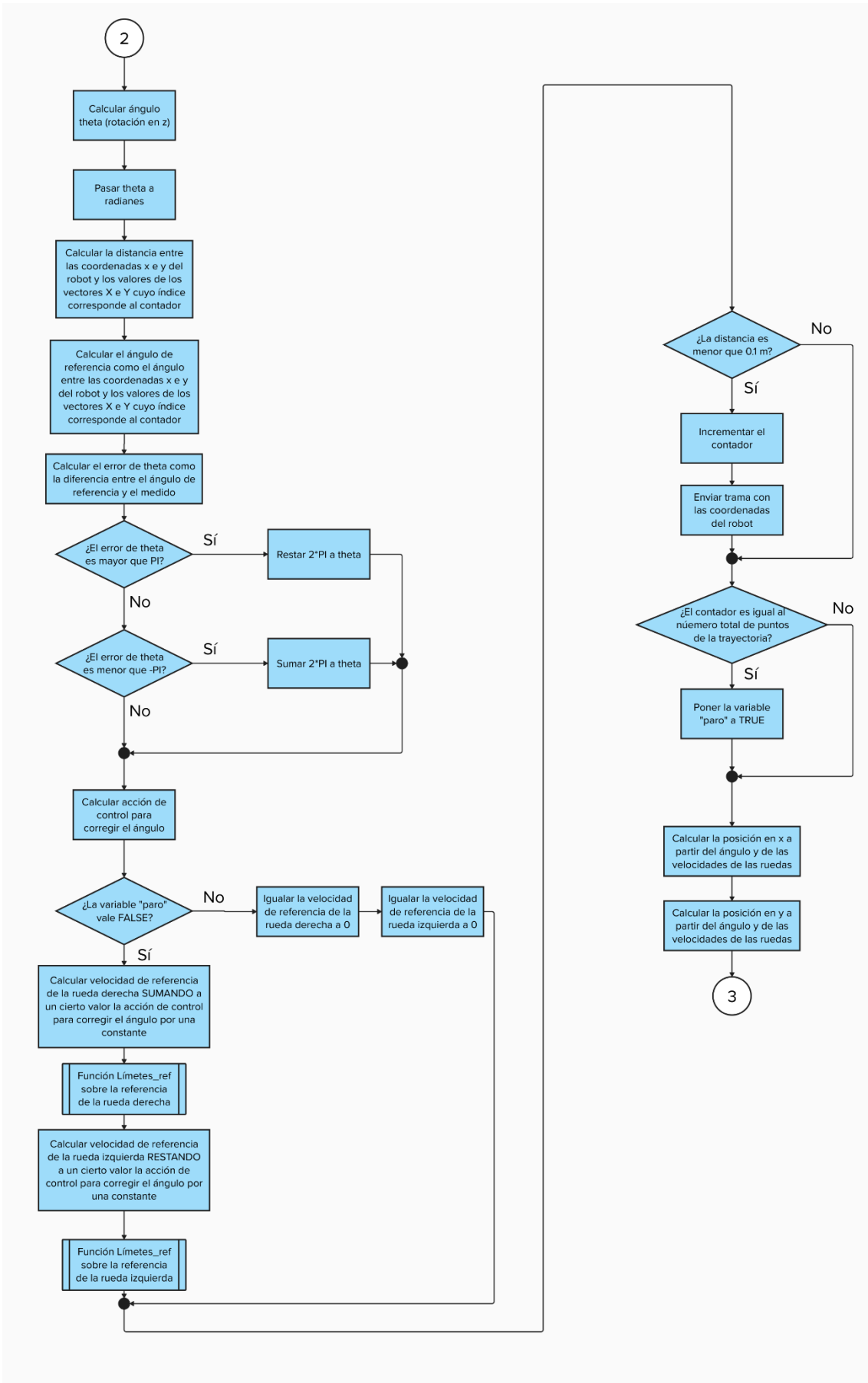


Figura 118. Flujograma del programa para el seguimiento de trayectorias (Parte 2).

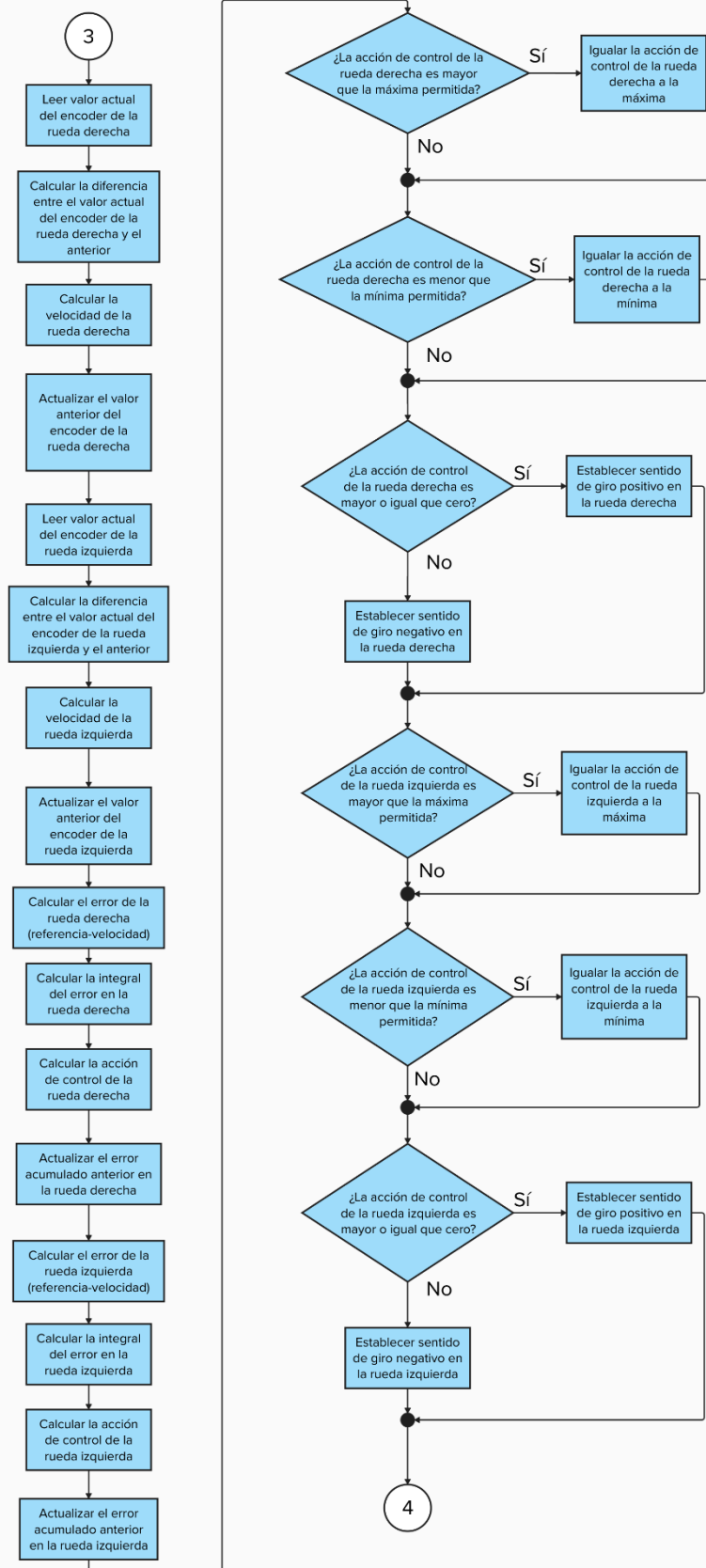


Figura 119. Flujograma del programa para el seguimiento de trayectorias (Parte 3).

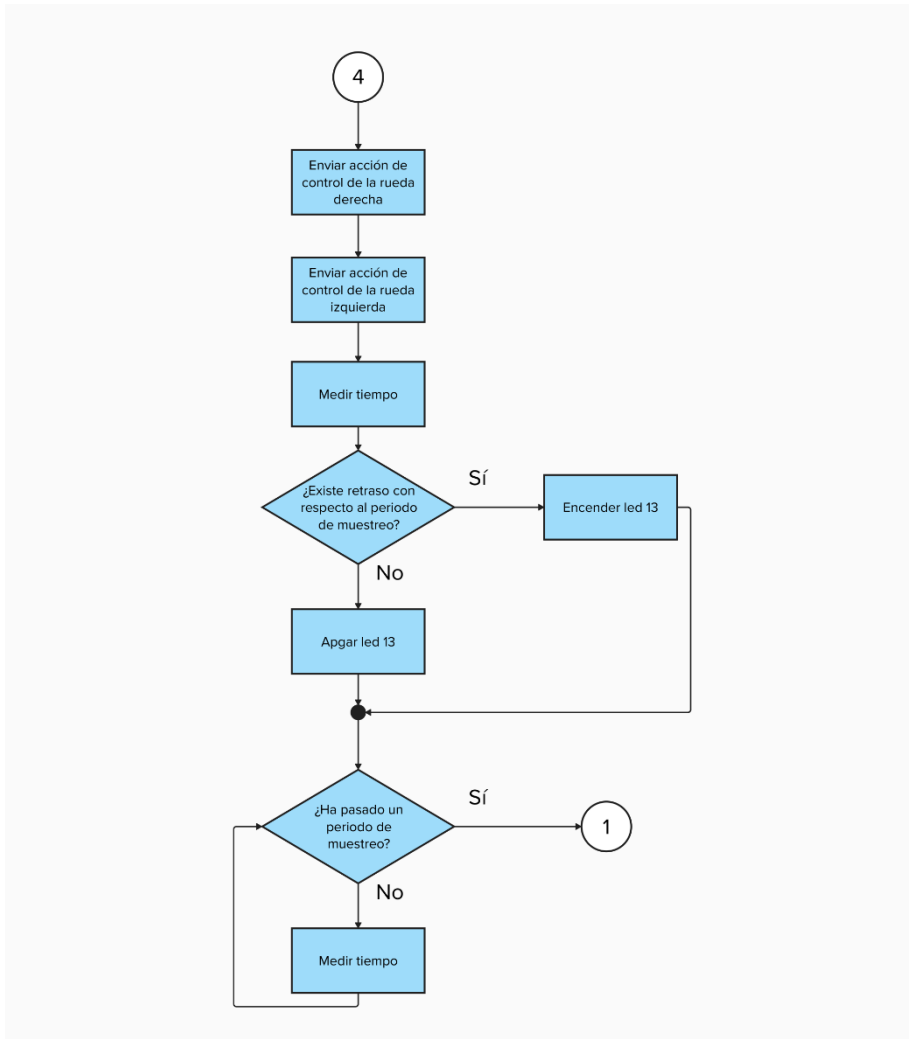


Figura 120. Flujograma del programa para el seguimiento de trayectorias (Parte 4).

Los flujogramas de la función Límites_ref y de las interrupciones utilizadas para la lectura de los encoder son idénticos a los del control remoto mostrados en la Figura 112 y la Figura 113 del apartado 6.3.

6.6. Aplicación Android para seguimiento de trayectorias

Para controlar y visualizar el seguimiento de trayectorias se ha desarrollado otra aplicación con la herramienta *Bluetooth Electronics*. La interfaz, mostrada en la Figura 121, contiene cuatro botones, cada uno con un texto que explica su función, y un gráfico. Tal y como se ha comentado en el apartado 6.5, el botón correspondiente al paro (STOP), de color rojo, envía un carácter “S”; el botón de la trayectoria circular, en amarillo, envía “O”; el pulsador de la trayectoria en forma de infinito, en verde, envía “I”, y el correspondiente a la trayectoria cuadrada, en azul, envía “C”. La configuración en *Bluetooth Electronics* de estos botones es idéntica a la indicada en el apartado 6.4.

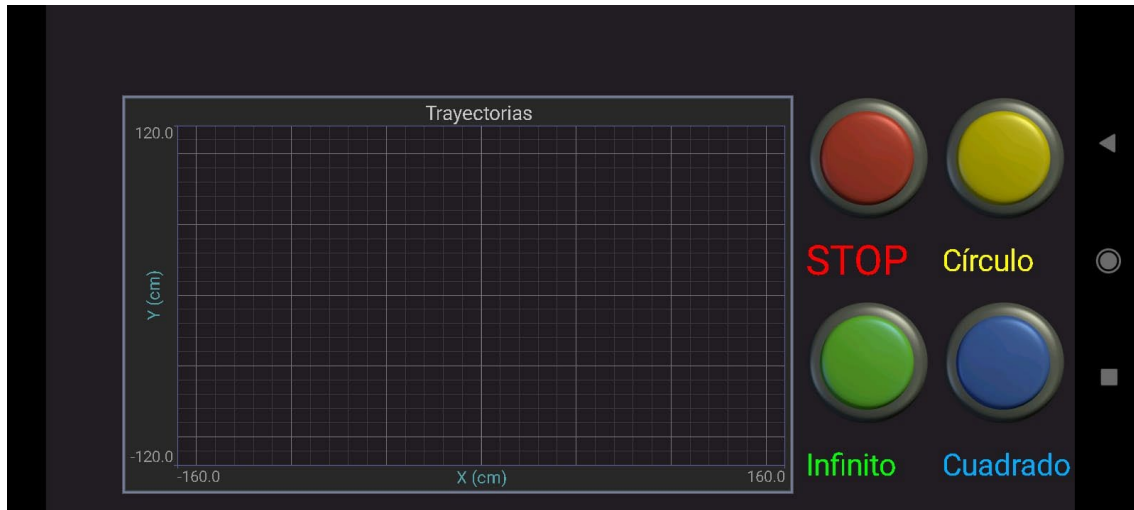


Figura 121. Interfaz de la aplicación para el seguimiento de trayectorias creada en *Bluetooth Electronics*.

Como ya se ha explicado en el apartado 6.5, el gráfico recibe los datos mediante una trama que comienza por el carácter “*” seguido de “H”. Después, se envía la letra “X” seguida del valor de la coordenada X y la letra “Y” seguida de la coordenada Y . La trama finaliza de nuevo con el carácter “*”. La configuración del gráfico se indica a continuación. En la Figura 122, se puede ver la descripción de la trama y el carácter que se recibe al comienzo de esta, en este caso “H”. Por otro lado, si recibe “*HC*”, borra las tramas del gráfico. En la Figura 123 y la Figura 124 se observa la configuración del gráfico, incluyendo el título, los títulos de los ejes, el número máximo de puntos que se representan a la vez y los límites de los ejes.

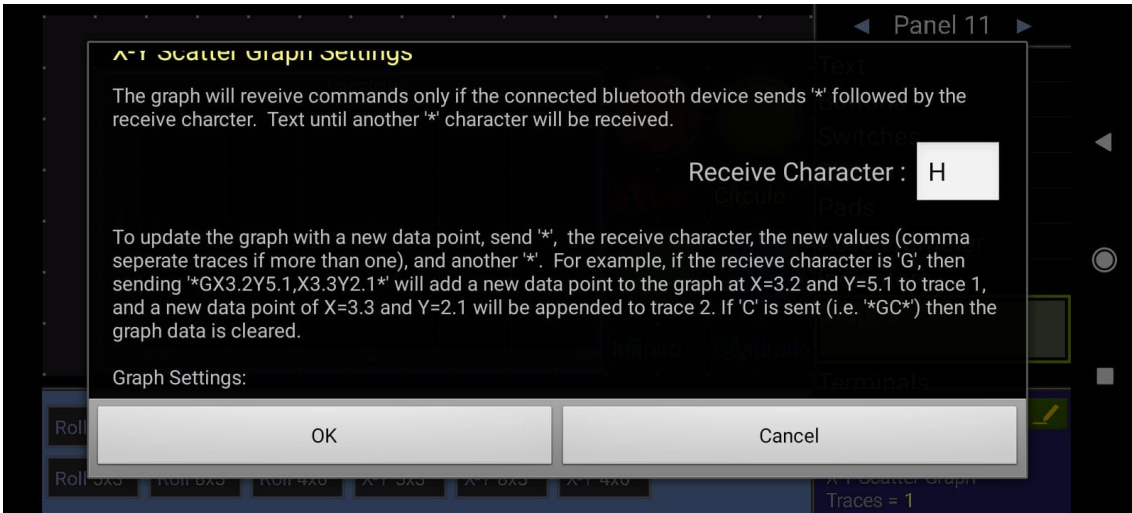


Figura 122. Configuración de gráfico en Bluetooth Electronics (Parte 1).

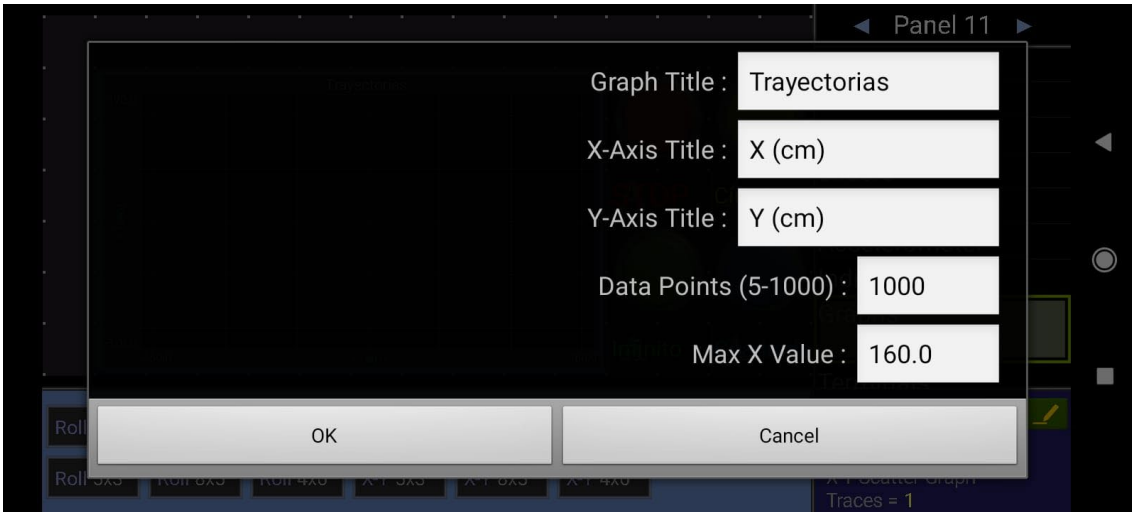


Figura 123. Configuración de gráfico en Bluetooth Electronics (Parte 2).

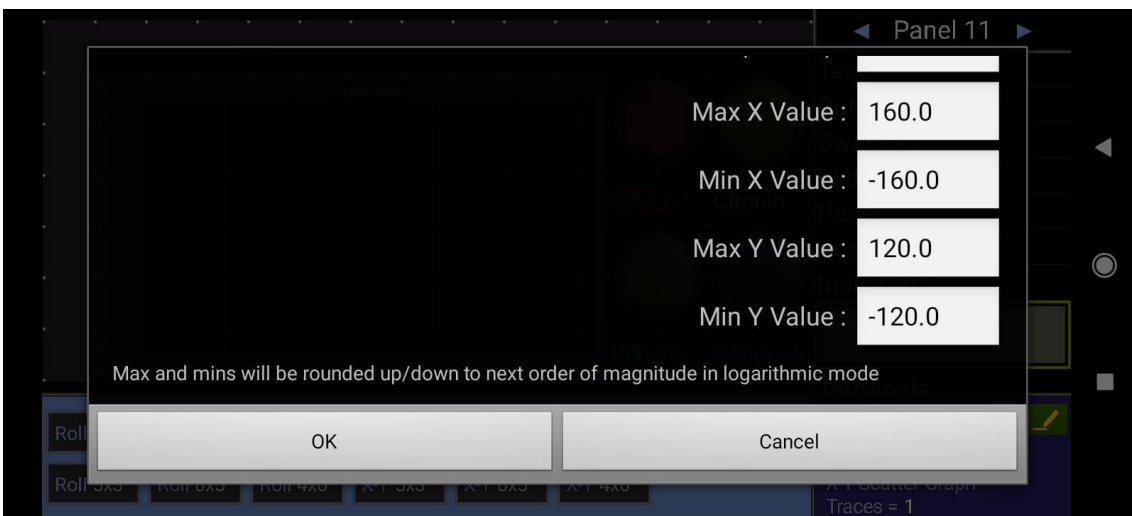


Figura 124. Configuración de gráfico en Bluetooth Electronics (Parte 3).

7. Resultados

En este capítulo se muestran y analizan los resultados obtenidos tras el montaje y programación del robot.

7.1. Control PI de Velocidad

De forma experimental, se ha ajustado un regulador tipo PI para el control de la velocidad de los motores. Se han ajustado las constantes proporcional e integral con el objetivo de lograr error de posición nulo con ausencia de sobreoscilación. Una constante proporcional de 0.5 y una constante integral de 1.5 satisfacen estas especificaciones. El tiempo de muestreo es de unos 100 ms. Este tiempo podría haberse reducido aún más, pero aceleraciones muy rápidas provocan un notable cabeceo en el robot, por lo que se ha considerado adecuado mantenerlo en torno al valor citado. Una sobreoscilación también provocaría un cabeceo en el robot, por lo que se ha buscado evitarla.

En el Gráfico 23 puede observarse la respuesta ante un escalón en la referencia de velocidad. La referencia se ha representado en color azul, la velocidad real, en rojo, y la acción de control, en verde. Se aprecia con claridad que se alcanza la referencia con un error estacionario nulo y que no existe sobreoscilación.

Dado que ambos motores son idénticos y no se observan diferencias entre el comportamiento de uno y otro, se ha empleado en ambos casos un regulador con iguales constantes.

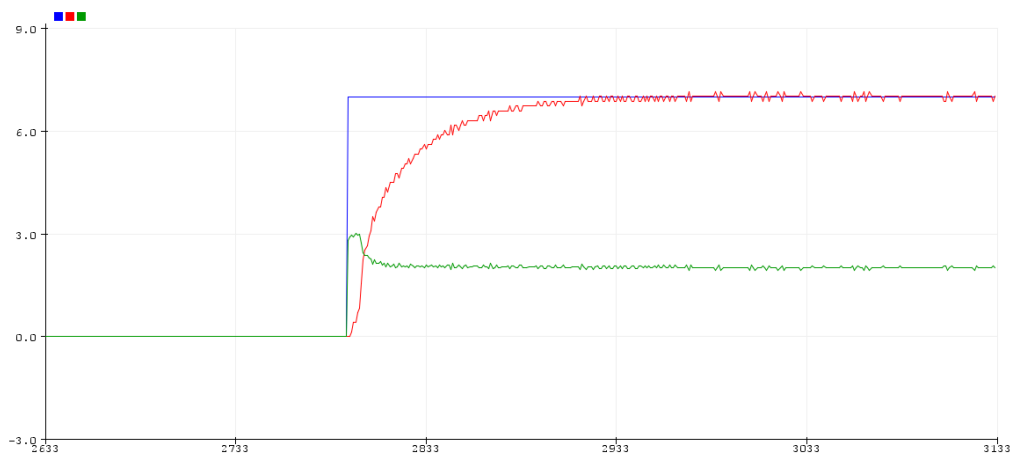


Gráfico 23. Respuesta de la velocidad de los motores frente a un cambio en escalón de la referencia.

7.2. Control remoto

Ejecutando el programa de control remoto implementado en Arduino junto con la aplicación Android diseñada en *Bluetooth Electronics* a tal efecto, se logra un control manual sobre el robot actuando sobre la velocidad de cada una de las ruedas por separado.

Por lo que respecta al funcionamiento de la aplicación, cabe decir que al pulsar las flechas superiores la velocidad aumenta de 1 rad/s en 1 rad/s y al pulsar las inferiores disminuye también de 1 rad/s en 1 rad/s. Ante estas variaciones, ambas ruedas responden acelerando o decelerando hasta alcanzar la referencia. Las velocidades de referencia se muestran en la aplicación y se actualizan cada vez que se pulsa una flecha a la par que las ruedas varían su velocidad real para alcanzarla. Además, al llegar a los valores límite, el máximo y mínimo nunca se sobrepasan, pues si se pulsa la flecha superior estando en el valor máximo o la inferior estando en el mínimo, se ignora la orden y no varía la velocidad de referencia mostrada en la aplicación ni tampoco la velocidad de la rueda correspondiente. Por otra parte, e independientemente de cuál sea la velocidad de las ruedas, al pulsar STOP ambas ruedas se detienen y las velocidades de referencia en los displays pasan a ser nulas.

Así pues, el funcionamiento de la aplicación y del programa del microcontrolador es el correcto.

Ahora resta analizar el comportamiento del robot ante las órdenes de control recibidas vía Bluetooth desde la aplicación.

La cinemática del robot se ha explicado en el apartado 3.5. A partir de las ecuaciones obtenidas puede concluirse que guardan ciertas similitudes con las que regulan la cinemática de un robot diferencial de dos ruedas. Al fin y al cabo, este robot se basa en un vehículo de dos ruedas, pero con la particularidad de estar en el interior de una esfera.

En ambos casos, la velocidad lineal del robot viene dada por la media de la velocidad de sus ruedas multiplicada por una constante. La diferencia entre los dos reside en que en el vehículo diferencial esa constante es el radio de la rueda y en este caso es el producto del radio de la rueda y el radio de la esfera dividido entre la distancia vertical entre el punto de contacto de la rueda con la esfera y el centro de esta. Por su parte, la velocidad angular de giro depende de la diferencia entre las velocidades de las ruedas y multiplicada también por un valor constante. En el vehículo diferencial es la distancia horizontal entre las ruedas, y en este caso depende de una serie de parámetros como son la inercia de la esfera, la inercia del vehículo, el radio de las ruedas o también la distancia horizontal entre ellas.

Así pues, salvo por las constantes que relacionan el movimiento del robot con el de sus ruedas, el comportamiento ante el giro de las ruedas es similar.

Del mismo modo que ocurre con los robots diferenciales, al accionar ambas ruedas con la misma velocidad el robot se mueve en línea recta hacia delante, o hacia atrás si se invierte el sentido de giro de ambas ruedas.

Si la velocidad de las ruedas es diferente se produce un giro hacia el lado cuya rueda tiene velocidad menor. Cuanto mayor sea la diferencia entre ambas ruedas, menor es el radio de giro. Además, si las dos ruedas giran en el mismo sentido, el centro del arco descrito se halla más allá de la proyección de la esfera sobre la superficie. Por el contrario, si el sentido de giro de ambas ruedas es distinto, el centro del arco se encuentra dentro de la proyección de la esfera sobre la superficie, dando lugar a un giro más cerrado. El caso extremo se

produce cuando ambas ruedas giran con igual velocidad, pero en sentidos opuestos. Entonces, el robot gira sobre sí mismo sin desplazarse.

Todo este comportamiento es similar al de un robot diferencial tradicional.

Es necesario hacer también mención del pulsador de paro, que al parar ambas ruedas tal y como ya se ha comentado, detiene al robot.

Durante las pruebas realizadas con el control remoto se observó un cierto cabeceo del vehículo interno que afectaba negativamente al movimiento del robot. Esta situación se daba ante velocidades elevadas y cambios bruscos en la velocidad de referencia. Para solucionarlo, se optó por alterar el regulador PI de los motores, pues el tiempo de establecimiento era más pequeño y daba lugar a aceleraciones más bruscas que tenían como consecuencia este cabeceo. Tras reajustar los parámetros del PI a los indicados en el apartado 7.1, se consiguió aumentar la estabilidad del vehículo de forma significativa mejorando el movimiento del robot. Si bien es cierto que ante cambios bruscos de velocidad sigue existiendo un ligero cabeceo, es mucho menor y no afecta al correcto desplazamiento del robot.

Existe una situación en la que se sigue produciendo un cabeceo acusado. Al pulsar el botón de paro el robot se detiene inmediatamente, por lo que, si yendo a velocidades elevadas se activa STOP, se produce un cierto cabeceo debido a la inercia del sistema, que perdura durante unos segundos. Aquí es preciso decir que este pulsador se ha implementado para situaciones de emergencia que requieran detener rápidamente al robot, por lo que, en circunstancias normales, el robot se detendrá haciendo uso de las flechas, reduciendo la velocidad de forma más suave y evitando por tanto el cabeceo debido al frenado brusco.

Por último, es importante destacar, que el led 13 en ningún momento se enciende, sino que permanece apagado, lo cual indica que no existe retraso sobre el periodo de muestreo.

7.3. Seguimiento de trayectorias

Tal y como se ha comentado en los apartados 6.5 y 6.6, se han definido tres trayectorias distintas para el robot, a saber, una circular, otra con la forma del símbolo del infinito y una tercera con forma cuadrada.

Representando en MATLAB las trayectorias definidas se obtienen el Gráfico 24, el Gráfico 25 y el Gráfico 26. Nótese que son las mismas trayectorias utilizadas en la simulación, pero escaladas para un espacio menor.

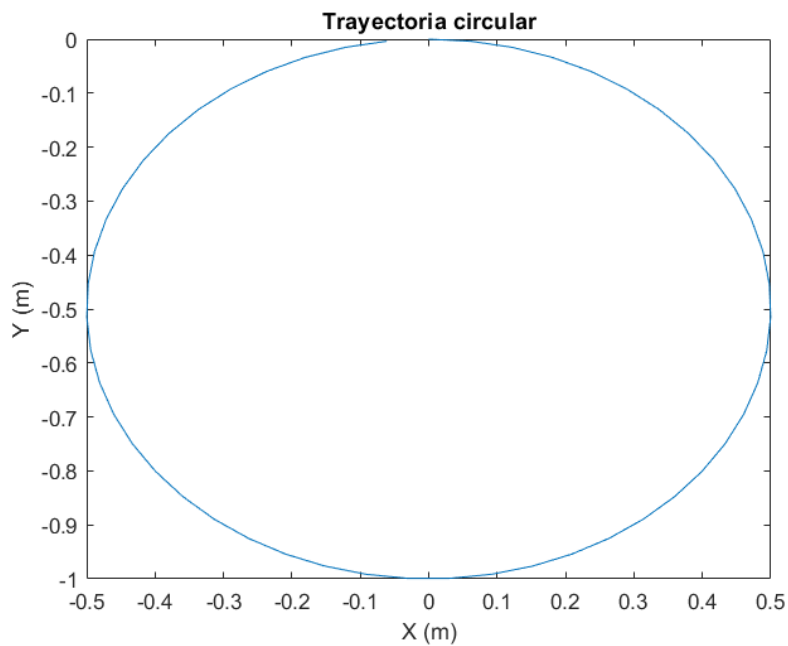


Gráfico 24. Trayectoria circular predefinida para el seguimiento de trayectorias.

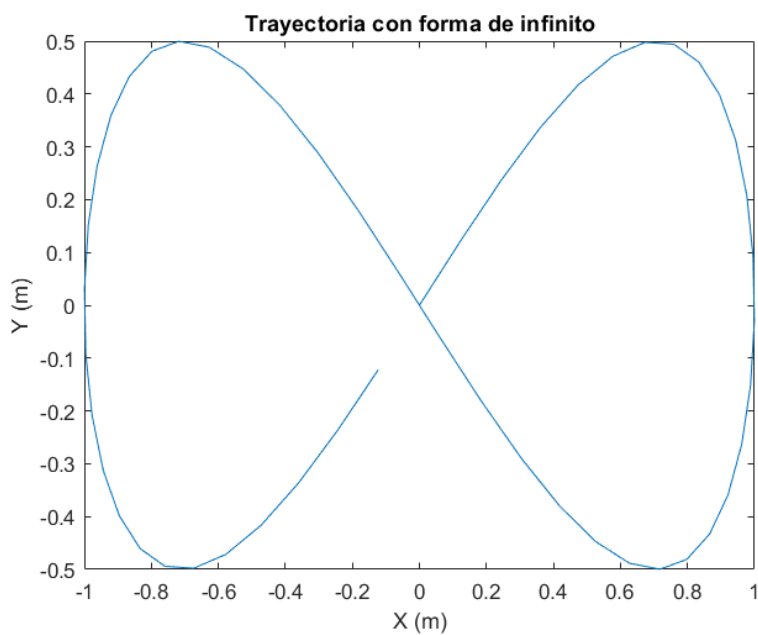


Gráfico 25. Trayectoria con la forma del símbolo del infinito predefinida para el seguimiento de trayectorias.

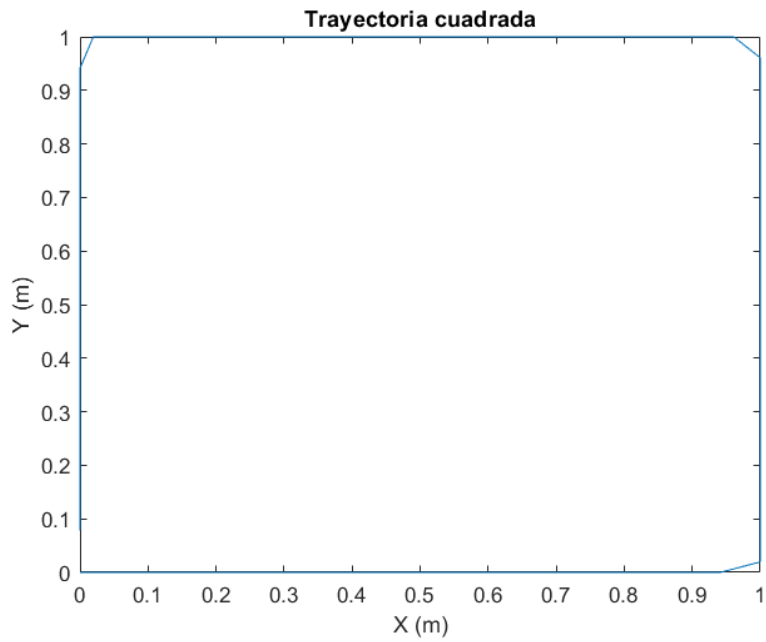


Gráfico 26. Trayectoria cuadrada predefinida para el seguimiento de trayectorias.

Ejecutando el programa del seguimiento de trayectorias utilizando además la aplicación diseñada en *Bluetooth Electronics* se han seguido las tres trayectorias obteniendo, en el gráfico de la propia aplicación los resultados que se observan en la Figura 125, la Figura 126 y la Figura 127.



Figura 125. Seguimiento de la trayectoria circular en la aplicación Android.

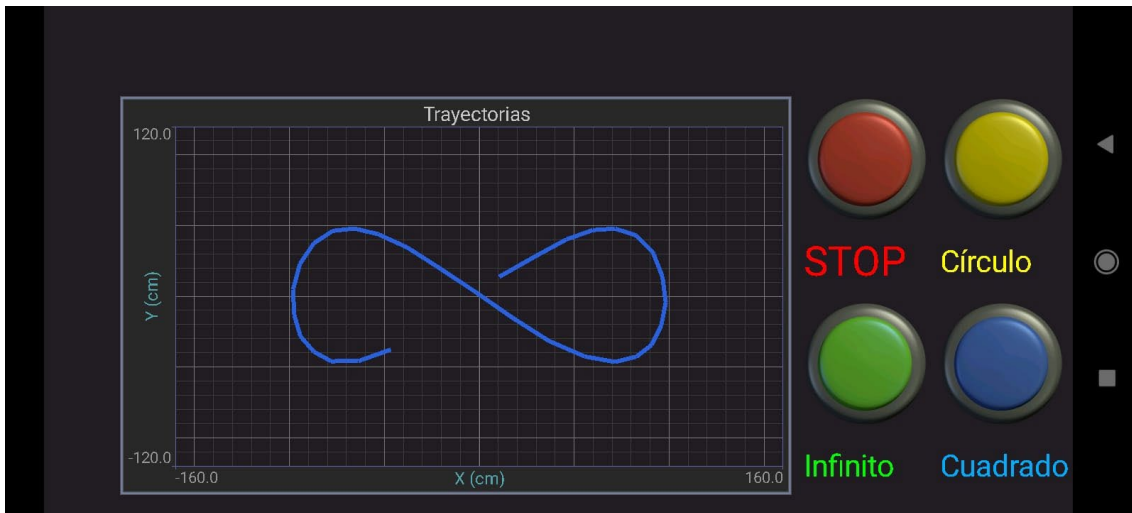


Figura 126. Seguimiento de la trayectoria con la forma del símbolo del infinito en la aplicación Android.

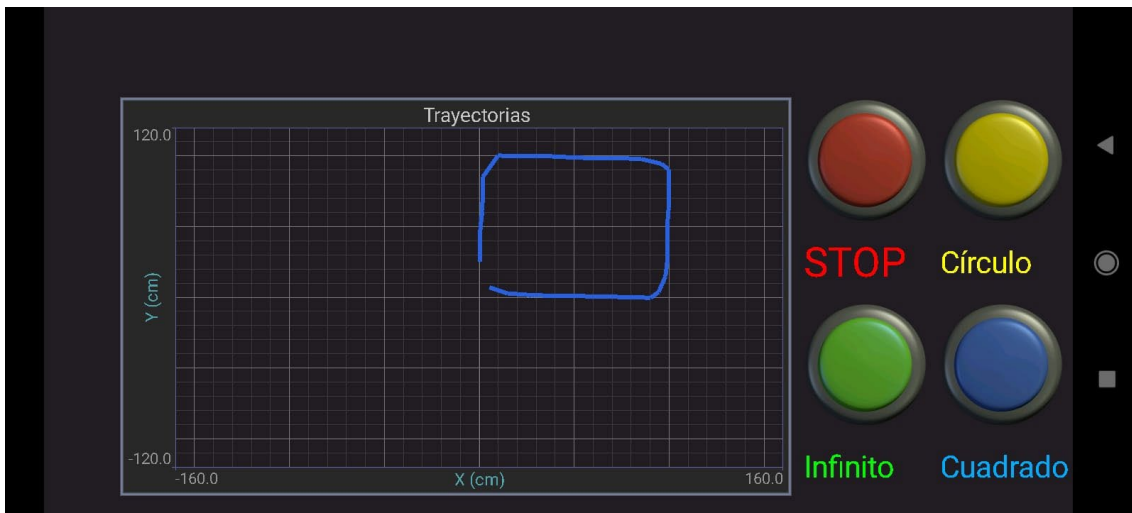


Figura 127. Seguimiento de la trayectoria cuadrada en la aplicación Android.

A la vista de los resultados obtenidos y mostrados en la Figura 125, la Figura 126 y la Figura 127, cabe decir que el seguimiento de trayectorias por parte del robot funciona correctamente. Si bien es necesario decir que únicamente se mide la orientación, mientras que la posición se estima. Por tanto, los resultados muestran un correcto seguimiento de la orientación, pero no necesariamente de la posición.

No obstante, en las pruebas realizadas se ha observado que el posicionamiento es también correcto. Aunque en algunas ocasiones existen diferencias de unos centímetros entre la posición deseada y la posición real, no es un error significativo. Este error es debido a que en el modelo no se tiene en cuenta el deslizamiento de la esfera sobre el suelo, cosa que, en determinadas superficies, como el pavimento interior de un edificio, es común. Pero, en cualquier caso, y a pesar de cierta desviación en suelos resbaladizos se puede asegurar un seguimiento de trayectorias aceptable.

Nuevamente, y al igual que en caso anterior se observa un ligero cabeceo del robot ante cambios bruscos de la velocidad y orientación. Sin embargo, este cabeceo acaba suavizándose al cabo de un tiempo y no supone ningún impedimento para seguir las trayectorias.

Por otra parte, el botón de paro funciona correctamente, deteniendo la ejecución de la trayectoria y parando al robot. Tras pulsar STOP, se puede empezar nuevamente una trayectoria desde el inicio accionando el pulsador correspondiente.

Por último, es importante destacar, que el led 13 en ningún momento se enciende, sino que permanece apagado, lo cual indica que no existe retraso sobre el periodo de muestreo.

8. Relación con los Objetivos de Desarrollo Sostenible

Sostenible

Los Objetivos de Desarrollo Sostenible (ODS) de la Organización de las Naciones Unidas (ONU) buscan abordar una serie de desafíos globales. Fueron fijados el 25 de septiembre de 2015 por los líderes mundiales con el fin de alcanzar un conjunto de metas con el propósito de erradicar la pobreza, proteger al planeta y garantiza la prosperidad para todos [18]. En la Figura 128 se pueden observar todos y cada uno de estos objetivos.



Figura 128. Objetivos de Desarrollo Sostenible [18].

Un robot móvil esférico como el prototipo objeto de estudio en este proyecto puede contribuir a varios de estos objetivos, como son los siguientes:

ODS 4: Educación de calidad

Puede ser empleado como una herramienta interactiva en las aulas para enseñar robótica, programación y otras ciencias, fomentando el interés en el campo de la ciencia y de la ingeniería entre los estudiantes.

ODS 7: Energía asequible y no contaminante

Un diseño esférico puede ser optimizado para minimizar el consumo de energía, contribuyendo al desarrollo de tecnologías más eficientes y sostenibles. Asimismo, se puede implementar de tal forma que únicamente use fuentes de energía limpias.

ODS 9: Industria, Innovación e Infraestructura

El desarrollo de robots móviles esféricos promueve la innovación tecnológica y además puede ser usado para inspeccionar infraestructuras, mejorar la seguridad y eficiencia en la industria.

ODS 11: Ciudades y comunidades sostenibles

Un robot móvil de este tipo puede ser utilizado para el monitoreo de la calidad del aire, detección de fugas de agua o gas, y otras tareas relacionadas, contribuyendo a la sostenibilidad de las ciudades.

ODS 12: Producción y consumo responsables

En la fabricación y diseño de estos robots se pueden implementar principios de economía circular, asegurando que los materiales sean reciclables y que el proceso de producción genere menos residuos.

ODS 13: Acción por el clima

Una esfera robótica puede ser empleada en la recolección de datos ambientales en áreas remotas o peligrosas, ayudando a monitorear los efectos del cambio climático y a tomar decisiones para reducir su impacto.

ODS 15: Vida de ecosistemas terrestres

Un robot esférico puede ser utilizado para monitorear la biodiversidad en áreas protegidas, recolectando datos sobre flora y fauna sin intervenir de manera invasiva en los hábitats naturales de las especies.

9. Conclusiones y trabajo futuro

En este apartado se exponen las conclusiones a las que se ha llegado tras la realización del presente proyecto, así como algunas líneas de trabajo futuro.

En primer lugar, se ha realizado un estudio de las distintas metodologías existentes para inducir el movimiento de una esfera, seleccionándose la del vehículo interno por su mayor sencillez en cuanto a su diseño y modelo matemático, su alta maniobrabilidad y su capacidad para superar pendientes y esquivar obstáculos. Partiendo del tipo de mecanismo seleccionado se ha llevado a cabo un diseño propio de un robot móvil esférico combinando algunas piezas disponibles en el mercado con otras diseñadas en SolidWorks, obteniendo un ensamblaje completo para una esfera robótica basada en un vehículo interno.

En segundo lugar, mediante el entorno de simulación que ofrece *Simscape Multibody* se ha implementado una simulación del robot importando las piezas seleccionadas y diseñadas y teniendo en cuenta sus masas e inercias. Esta simulación ha servido para estudiar el comportamiento del robot y su respuesta frente al movimiento de sus ruedas, con lo que se ha podido comprobar que el modelo cinemático estudiado en el marco teórico se corresponde con el movimiento obtenido en la simulación.

La simulación se ha utilizado también para verificar el correcto funcionamiento de los algoritmos desarrollados tanto para el control remoto, a partir del establecimiento de las velocidades de referencia de ambas ruedas por el usuario, como para el seguimiento de trayectorias. En los dos casos, se ha obtenido un resultado satisfactorio en la simulación. Al simular el control remoto, se observa como el robot se desplaza en línea recta hacia delante cuando las velocidades de las ruedas son iguales y positivas, hacia atrás si son iguales y negativas, y cómo gira si las velocidades son distintas. Si las velocidades son iguales y de signo opuesto gira sobre sí mismo, permaneciendo su centro fijo en un mismo punto. Todo esto confirma la veracidad del modelo cinemático del robot.

Por otro lado, al simular el seguimiento de trayectoria con el algoritmo *Follow the carrot* implementado se aprecia como el robot sigue el camino marcado con un error mínimo.

Así pues, todos los resultados obtenidos en la simulación son los esperados y confirman el correcto diseño del robot y el planteamiento de los algoritmos de control y de trayectorias.

Realizado el diseño y verificado el funcionamiento del sistema mediante la ya mencionada simulación, se ha procedido al montaje del robot. Aquí cabe destacar algunos aspectos importantes que han dado lugar a problemas durante el proceso.

Primeramente, uno de los grandes desafíos ha sido lograr una esfera que pudiera montarse y desmontarse abriéndose por la mitad sin dejar ningún elemento en su interior ni en su exterior que pudiera entorpecer el movimiento del vehículo en el interior de la esfera o de la propia esfera sobre la superficie. La solución más viable, por la que se ha optado finalmente, es la de dos semiesferas, cuyos bordes encajen, siendo estos bordes más finos que el resto de la esfera y estando uno a la altura de la superficie exterior y otro de la interior.

Otro problema ha sido fijar las ruedas a los ejes de los motores. Esto se ha logrado taladrando los agujeros de las ruedas, que tenían menor diámetro que el eje del motor y

colocando entre ambos un fragmento de un tubo de metal del diámetro del eje del motor y con su misma forma, que presenta un lado plano.

También es importante resaltar la labor de las ruedas esféricas, que mantienen fijo al robot en la esfera evitando que se mueva y choque contra la pared interna de la esfera. Los pequeños muelles que llevan los soportes de las ruedas esféricas en la parte trasera logran garantizar el contacto con la esfera y hacen que introducir y sacar el vehículo de la bola resulte sencillo. Si las ruedas esféricas estuviesen fijas en la posición justa para tocar la esfera por dentro, introducir el vehículo en la esfera resultaría más difícil y, además, cualquier pequeño error en el cálculo o movimiento de alguna de ellas, podría provocar que el vehículo quedará suelto o impedir el contacto de alguna de las ruedas motrices con la esfera.

Con todo, se ha conseguido montar correctamente el robot, ensamblando todas y cada una de sus piezas mecánicas, así como sus componentes electrónicos.

Una vez montado el robot, se ha implementado, un programa para su control remoto, utilizando una aplicación móvil. Los resultados obtenidos muestran un funcionamiento similar al logrado en la simulación, tal y como cabía esperar. No obstante, existía un cabeceo ante cambios de velocidad, que se ha corregido variando el regulador PI aumentando el tiempo de respuesta y logrando aceleraciones más suaves.

Esto prueba la importancia de un buen control. En este caso, no es preciso un control excesivamente rápido, esto es, no es crítico que el motor alcance la velocidad deseada en cuestión de pocos milisegundos, pero sí que el vehículo sea estable. Por tanto, se ha primado este requisito sobre la rapidez del control.

Por último, se ha desarrollado también otro programa para el seguimiento de trayectorias predeterminadas, haciendo uso también de una aplicación móvil. Para ello ha sido necesario utilizar un giroscopio. Es cierto que se dispone de un modelo matemático con el que, a partir de las velocidades de las ruedas y de los parámetros físicos del robot, se podría estimar la orientación y posición del robot. Sin embargo, la dificultad de determinar con precisión algunos de los parámetros que intervienen en la estimación de la orientación hace que sea mucho más sencillo el uso de un giroscopio que permita leer la velocidad de giro para, a partir de ella, determinar la orientación. La velocidad lineal y la posición son más sencillas de estimar, mientras que su medida mediante el acelerómetro, integrado en el mismo módulo que el giroscopio, no es precisa debido al ligero cabeceo que persiste en el vehículo, lo que provoca que las medidas en la aceleración en los ejes X e Y sean erróneas. Por estos motivos, se ha optado por estimar la velocidad lineal, y a partir de ella la posición del robot, pero no la orientación, que se ha medido mediante un sensor.

Con todo, las pruebas realizadas muestran un adecuado seguimiento de las trayectorias, de una forma similar a la simulación. Si bien es cierto que, en algunos terrenos resbaladizos como pueden ser las losas de un interior, existe algo de deslizamiento, que no se tiene en cuenta en el modelo. Por ello, el seguimiento del ángulo es correcto, pues existe un sensor que lo mide, pero se pueden producir pequeñas desviaciones en las distancias recorridas. A pesar de ello, el error no es significativo.

Por último, se proponen algunas posibles mejoras al prototipo implementado que pueden ser objeto de trabajos futuros.

Primeramente, y, aunque como ya se ha comentado, se ha logrado reducir de manera significativa el cabeceo del robot, aún persiste un cierto cabeceo, si bien mucho menor que el inicial. Una posible línea de trabajo podría consistir en tratar de eliminarlo. Esto podría lograrse aumentando el peso del vehículo y bajando su centro de gravedad, suavizando aún más la respuesta de los motores o incluso se podría implementar, haciendo uso del acelerómetro, un control de los ángulos de cabeceo.

Otra mejora podría ser la incorporación de un módulo GPS al robot, con lo que ya no sería preciso estimar la posición, sino que se podría medir. Con ello se lograría un control más preciso y se podrían eliminar los errores en la posición incluso en terrenos con poco rozamiento.

Finalmente, se podrían incorporar al robot sensores ambientales o cámaras que permitan realizar un análisis de las condiciones del entorno, lo que aumentaría las funcionalidades del robot. Incluso se podrían desarrollar aplicaciones Bluetooth que, además de controlar el robot reciban la información recogida por los sensores y cámaras incorporadas.

En definitiva, a pesar de las dificultades encontradas y de todo el posible trabajo futuro por hacer, se puede afirmar que se han logrado todos los objetivos planteados para este proyecto al haberse diseñado implementado y programado un prototipo de robot móvil esférico.

10. Bibliografía

En este apartado se indican las fuentes consultadas en la fase de investigación del proyecto.

- [1] F. Rubio, F. Valero, y C. Llopis-Albert, «A review of mobile robots: Concepts, methods, theoretical framework, and applications», *Int. J. Adv. Robot. Syst.*, vol. 16, n.º 2, p. 172988141983959, mar. 2019, doi: 10.1177/1729881419839596.
- [2] D. Rotondo, V. Puig, F. Nejjari, y J. Romera, «A Fault-Hiding Approach for the Switching Quasi-LPV Fault Tolerant Control of a Four-Wheeled Omnidirectional Mobile Robot», *IEEE Trans. Ind. Electron.*, pp. 1-1, 2014, doi: 10.1109/TIE.2014.2367002.
- [3] T. Kislassi y D. Zarrouk, «A Minimally Actuated Reconfigurable Continuous Track Robot», *IEEE Robot. Autom. Lett.*, pp. 1-1, 2019, doi: 10.1109/LRA.2019.2959237.
- [4] P. Liljeback, O. Stavadahl, K. Y. Pettersen, y J. T. Gravdahl, «Mamba - A waterproof snake robot with tactile sensing», en *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA: IEEE, sep. 2014, pp. 294-301. doi: 10.1109/IROS.2014.6942575.
- [5] J. Alves y J. Dias, «Design and control of a spherical mobile robot», vol. 217, 2003.
- [6] V. A. Crossley, «A Literature Review on the Design of Spherical Rolling Robots».
- [7] A. Halme, J. Suomela, T. Schönberg, y Y. Wang, «A SPHERICAL MOBILE MICRO-ROBOT FOR SCIENTIFIC APPLICATIONS».
- [8] A. Bicchi, A. Balluchi, D. Prattichizzo, y A. Gorelli, «Introducing the “SPHERICLE”: an experimental testbed for research and teaching in nonholonomy», en *Proceedings of International Conference on Robotics and Automation*, Albuquerque, NM, USA: IEEE, 1997, pp. 2620-2625. doi: 10.1109/ROBOT.1997.619356.
- [9] S. Bhattacharya y S. K. Agrawal, «Design, experiments and motion planning of a spherical rolling robot», en *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, USA: IEEE, 2000, pp. 1207-1212. doi: 10.1109/ROBOT.2000.844763.
- [10] R. Mukherjee, M. A. Minor, y J. T. Pukrushpan, «Simple motion planning strategies for spherobot: a spherical mobile robot», en *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, Phoenix, AZ, USA: IEEE, 1999, pp. 2132-2137. doi: 10.1109/CDC.1999.831235.
- [11] Amir Homayoun Javadi A y P. Mojabi, «Introducing August: a novel strategy for an omnidirectional spherical rolling robot», en *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA: IEEE, 2002, pp. 3527-3533. doi: 10.1109/ROBOT.2002.1014256.
- [12] J. Lux, «Alternative Way of Shifting Mass to Move a Spherical Robot», NPO-30491, jun. 2005. Accedido: 28 de abril de 2024. [En línea]. Disponible en: <https://ntrs.nasa.gov/citations/20110014914>

- [13] T. Ohsawa, «Geometric Kinematic Control of a Spherical Rolling Robot», *J. Nonlinear Sci.*, vol. 30, n.º 1, pp. 67-91, feb. 2020, doi: 10.1007/s00332-019-09568-x.
- [14] Y. Oka y K. Suzuki, Eds., *Nuclear Reactor Kinetics and Plant Control*. en *An Advanced Course in Nuclear Engineering*. Tokyo: Springer Japan, 2013. doi: 10.1007/978-4-431-54195-0.
- [15] S. A. Wahyu y D. P. Riky, «DC motor simulation transfer function estimation: case study Proteus Ver. 7», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 674, n.º 1, p. 012040, nov. 2019, doi: 10.1088/1757-899X/674/1/012040.
- [16] R. Bhadani, «Path Planning of Unmanned System using Carrot-chasing Algorithm». arXiv, 24 de diciembre de 2020. Accedido: 18 de mayo de 2024. [En línea]. Disponible en: <http://arxiv.org/abs/2012.13227>
- [17] MathWorks, «<https://es.mathworks.com/products/simscape-multibody.html>», Simscape Multibody. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: <https://es.mathworks.com/products/simscape-multibody.html>
- [18] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», *Desarrollo Sostenible*. Accedido: 13 de junio de 2024. [En línea]. Disponible en: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

ANEXO I:

PROGRAMA DEL CONTROL REMOTO EN ARDUINO

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

```

#define RT 13 // led 13

#define pinENB_d 6 // velocidad motor derecho
#define pinIN4_d 4 // pines sentido de giro motor derecho
#define pinIN3_d 5

#define pinENA_i 7 // velocidad motor izquierdo
#define pinIN2_i 8 // pines sentido de giro motor izquierdo
#define pinIN1_i 9

#define pinENC_chA_d 3 // pines encoder motor derecho
#define pinENC_chB_d 2

#define pinENC_chA_i 18 // pines encoder motor izquierdo
#define pinENC_chB_i 19

#define MINU -10 // acción de control mínima
#define MAXU 10 // acción de control máxima
#define MINW -12 // velocidad mínima
#define MAXW 12 // velocidad máxima

long t1=0;
long t2=0;
int T=10 ; // periodo de muestreo (ms)
float Tm=T/1000.0; // periodo de muestreo (s)
float C2RS=(2*3.14/4480.0)/Tm; // factor de conversión de cuentas a rad/s

float acc_d=0.0,acc_i=0.0; // acciones de control
float vel_d=0.0,vel_i=0.0; // velocidades de los motores
float pos_d=0.0,pos_i=0.0; // posiciones de los motores
long quad_d=2241,quad_i=2241 ; // cuentas de los encoder
long enc_ant_d=0,enc_ant_i=0; // cuenta anterior de los encoder
long enc_act_d=0,enc_act_i=0; // cuenta actual de los encoder
int enc_dif_d=0,enc_dif_i=0; // diferencia de la cuenta de los encoder

float ref_d=0.0,ref_i=0.0; // referencias de velocidad

// error, Kp del regulador, integral del error,
// integral del error anterior, Ki del regulador RUEDA DERECHA
float error_d,kp_d=0.5,integral_d,integral_anterior_d=0,ki_d=1.5;
// error, Kp del regulador, integral del error,
// integral del error anterior, Ki del regulador RUEDA IZQUIERDA
float error_i,kp_i=0.5,integral_i,integral_anterior_i=0,ki_i=1.5;

void setup()
{
    Serial.begin(9600); // baud rate comunicación serie
    Serial.setTimeout(50); // timeout

    pinMode(RT,OUTPUT) ; // led 13 como salida
    digitalWrite(RT,LOW); // led 13 apagado

    // pines de los encoder como entradas
    pinMode(pinENC_chA_d,INPUT);
    pinMode(pinENC_chB_d,INPUT);
    pinMode(pinENC_chA_i,INPUT);
    pinMode(pinENC_chB_i,INPUT);

    // Interrupciones encoder motor derecho
    attachInterrupt(digitalPinToInterrupt(pinENC_chA_d),encoder_chA_d,CHANGE);
    attachInterrupt(digitalPinToInterrupt(pinENC_chB_d),encoder_chB_d,CHANGE);

    // Interrupciones encoder motor izquierdo
    attachInterrupt(digitalPinToInterrupt(pinENC_chA_i),encoder_chA_i,CHANGE);

```

```

attachInterrupt(digitalPinToInterrupt(pinENC_chB_i),encoder_chB_i,CHANGE);

// Salidas digitales y PWM para control de motores
pinMode(pinIN4_d, OUTPUT);
pinMode(pinIN3_d, OUTPUT);
pinMode(pinENB_d, OUTPUT);
pinMode(pinIN1_i, OUTPUT);
pinMode(pinIN2_i, OUTPUT);
pinMode(pinENA_i, OUTPUT);
}

void loop()
{
  // lectura serie
  char lectura;
  if(Serial.available()>0){ // si se ha recibido algún dato
    lectura=Serial.read(); // lee un caracter
    if(lectura=='L'){ // si se lee L
      ref_i++; // incremta velocidad de la rueda izquierda
      ref_i=limites_ref(ref_i); // función límites_ref
      // envía *I, el valor de la referencia y *
      Serial.print("*I");Serial.print(ref_i);Serial.println("");
    }else if(lectura=='l'){ // si se lee l
      ref_i--; // decremta velocidad de la rueda izquierda
      ref_i=limites_ref(ref_i); // función límites_ref
      // envía *I, el valor de la referencia y *
      Serial.print("*I");Serial.print(ref_i);Serial.println("");
    }else if(lectura=='R'){ // si se lee R
      ref_d++; // incremta velocidad de la rueda derecha
      ref_d=limites_ref(ref_d); // función límites_ref
      // envía *D, el valor de la referencia y *
      Serial.print("*D");Serial.print(ref_d);Serial.println("");
    }else if(lectura=='r'){ // si se lee r
      ref_d--; // decremta velocidad de la rueda derecha
      ref_d=limites_ref(ref_d); // función límites_ref
      // envía *D, el valor de la referencia y *
      Serial.print("*D");Serial.print(ref_d);Serial.println("");
    }else if(lectura=='S'){ // si se lee S
      ref_d=0; // referencia a 0
      // envía *D, el valor de la referencia y *
      Serial.print("*D");Serial.print(ref_d);Serial.println("");
      ref_i=0; // referencia a 0
      // envía *I, el valor de la referencia y *
      Serial.print("*I");Serial.print(ref_i);Serial.println("");
    }
  }
}

enc_act_d=quad_d; // valor actual encoder rueda derecha
// diferencia entre el valor actual y el anterior encoder rueda derecha
enc_dif_d=enc_act_d-enc_ant_d;
// cálculo de la velocidad de la rueda derecha en rad/s
vel_d=(enc_act_d-enc_ant_d)*C2RS;
// actualización del valor del encoder actual rueda derecha
enc_ant_d=enc_act_d;

enc_act_i=quad_i; // valor actual encoder rueda izquierda
// diferencia entre el valor actual y el anterior encoder rueda izquierda
enc_dif_i=enc_act_i-enc_ant_i;
// cálculo de la velocidad de la rueda izquierda en rad/s
vel_i=(enc_act_i-enc_ant_i)*C2RS;
// actualización del valor del encoder actual rueda izquierda
enc_ant_i=enc_act_i;

//Señal de error rueda derecha
error_d=ref_d-vel_d;

//Integral del error rueda derecha
integral_d=error_d*Tm+integral_anterior_d;

```

```

//Acción de control rueda derecha
acc_d=kp_d*error_d+ki_d*integral_d;

// Actualización del error acumulado anterior rueda derecha
integral_anterior_d=integral_d;

//Señal de error rueda izquierda
error_i=ref_i-vel_i;

//Integral del error rueda izquierda
integral_i=error_i*Tm+integral_anterior_i;

//Acción de control rueda izquierda
acc_i=kp_i*error_i+ki_i*integral_i;

// Actualización del error acumulado anterior rueda izquierda
integral_anterior_i=integral_i;

// límites acción de control (rueda derecha)
if (acc_d>=MAXU){
    acc_d=MAXU;
}

if (acc_d<=MINU){
    acc_d=MINU;
}

if (acc_d>=0) { // acción de control positiva (rueda derecha)
    digitalWrite(pinIN4_d, HIGH); // sentido de giro positivo (rueda derecha)
    digitalWrite(pinIN3_d, LOW);
} else { // caso contrario
    digitalWrite(pinIN4_d, LOW); //sentido de giro negativo (rueda derecha)
    digitalWrite(pinIN3_d, HIGH);
}

// límites acción de control (rueda izquierda)
if (acc_i>=MAXU){
    acc_i=MAXU;
}
if (acc_i<=MINU){
    acc_i=MINU;
}

if (acc_i>=0) { // acción de control positiva (rueda izquierda)
    digitalWrite(pinIN1_i, HIGH); // sentido de giro positivo (rueda
izquierda)
    digitalWrite(pinIN2_i, LOW);
} else { // caso contrario
    digitalWrite(pinIN1_i, LOW); //sentido de giro negativo (rueda izquierda)
    digitalWrite(pinIN2_i, HIGH);
}

// si la acción de control es nula, la velocidad se fuerza a cero
if(ref_i==0) acc_i=0;
if(ref_d==0) acc_d=0;

// envía la acción de control a los pines ENB y ENA del puente en H
analogWrite(pinENB_d, (byte) (abs(acc_d)*255.0/MAXU)) ;
analogWrite(pinENA_i, (byte) (abs(acc_i)*255.0/MAXU)) ;

t2=millis(); // mide el tiempo
//Si hay retardo se enciende el led
if (t2-t1>T) {digitalWrite(RT,HIGH) ;}
else {digitalWrite(RT,LOW) ;}
//Espera al siguiente periodo de muestreo
while (t2-t1<T){
    t2=millis();
}

```



```

    }
    t1=millis(); // actualiza t1
}

// función límites_ref
float limites_ref(float ref){
    if(ref>MAXW){
        ref=MAXW;
    }else if(ref<MINW){
        ref=MINW;
    }
    return ref;
}

// funciones para leer los encoder (por interrupción)
void encoder_chA_d (){
    if(digitalRead(pinENC_chA_d)==digitalRead(pinENC_chB_d)){
        quad_d++;
    }else{
        quad_d--;
    }
}
void encoder_chB_d (){
    if(digitalRead(pinENC_chA_d)==digitalRead(pinENC_chB_d)){
        quad_d--;
    }else{
        quad_d++;
    }
}
void encoder_chA_i (){
    if(digitalRead(pinENC_chA_i)==digitalRead(pinENC_chB_i)){
        quad_i++;}
    else{
        quad_i--;
    }
}
void encoder_chB_i (){
    if(digitalRead(pinENC_chA_i)==digitalRead(pinENC_chB_i)){
        quad_i--;
    }else{
        quad_i++;
    }
}
}

```



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

ANEXO II:

PROGRAMA DEL SEGUIMIENTO DE TRAYECTORIAS EN ARDUINO

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

```

#define RT 13 // led 13

#define pinENB_d 6 // velocidad motor derecho
#define pinIN4_d 4 // pines sentido de giro motor derecho
#define pinIN3_d 5

#define pinENA_i 7 // velocidad motor izquierdo
#define pinIN2_i 8 // pines sentido de giro motor izquierdo
#define pinIN1_i 9

#define pinENC_chA_d 3 // pines encoder motor derecho
#define pinENC_chB_d 2

#define pinENC_chA_i 18 // pines encoder motor izquierdo
#define pinENC_chB_i 19

#define MINU -10 // acción de control mínima
#define MAXU 10 // acción de control máxima
#define MINW -5 // velocidad mínima
#define MAXW 5 // velocidad máxima

#define radio_rueda 0.045
#define radio_esfera 0.15
#define h 0.116

#define MPU9250_ADDRESS 0x68
#define MAG_ADDRESS 0x0C
#define GYRO_FULL_SCALE_250_DPS 0x00
#define GYRO_FULL_SCALE_500_DPS 0x08
#define GYRO_FULL_SCALE_1000_DPS 0x10
#define GYRO_FULL_SCALE_2000_DPS 0x18
#define ACC_FULL_SCALE_2_G 0x00
#define ACC_FULL_SCALE_4_G 0x08
#define ACC_FULL_SCALE_8_G 0x10
#define ACC_FULL_SCALE_16_G 0x18

#include "Wire.h"

long t1=0;
long t2=0;
int T=20 ; // periodo de muestreo (ms)
float Tm=T/1000.0; // periodo de muestreo (s)
float C2RS=(2*3.14/4480.0)/Tm; // factor de conversión de cuentas a rad/s

float acc_d=0.0,acc_i=0.0; // acciones de control
float vel_d=0.0,vel_i=0.0; // velocidades de los motores
float pos_d=0.0,pos_i=0.0; // posiciones de los motores
long quad_d=2241,quad_i=2241 ; // cuentas de los encoder
long enc_ant_d=0,enc_ant_i=0; //cuenta anterior de los encoder
long enc_act_d=0,enc_act_i=0; // cuenta actual de los encoder
int enc_dif_d=0,enc_dif_i=0; // diferencia de la cuenta de los encoder

float ref_d=0.0,ref_i=0.0; // referencias de velocidad

// error, Kp del regulador, integral del error,
// intergral del error anterior, Ki del regulador RUEDA DERECHA
float error_d,kp_d=0.5,integral_d,integral_anterior_d=0,ki_d=1.5;
// error, Kp del regulador, integral del error,
// intergral del error anterior, Ki del regulador RUEDA IZQUIERDA
float error_i,kp_i=0.5,integral_i,integral_anterior_i=0,ki_i=1.5;

int const np=33; // número de puntos de la trayectoria

float X[np]; // vector para los valores x de la trayectoria
float Y[np]; // vector para los valores y de la trayectoria

float x=0.0, y=0.0, theta=0.0,theta_ref; // variables para la posición x e y y
para el ángulo y el ángulo de referencia

```

```

float kp_theta=2; // constante para el regulador del ángulo
bool paro=true; // variable booleana para detener la trayectoria

int cont=1; //contador

void setup()
{
    Serial.begin(9600); // baud rate comunicación serie
    Serial.setTimeout(50); // timeout

    Wire.begin();
    // Configurar acelerometro
    I2CwriteByte(MPU9250_ADDRESS, 28, ACC_FULL_SCALE_16_G);
    // Configurar giroscopio
    I2CwriteByte(MPU9250_ADDRESS, 27, GYRO_FULL_SCALE_2000_DPS);
    // Configurar magnetometro
    I2CwriteByte(MPU9250_ADDRESS, 0x37, 0x02);
    I2CwriteByte(MAG_ADDRESS, 0x0A, 0x01);

    pinMode(RT,OUTPUT) ; // led 13 como salida
    digitalWrite(RT,LOW); // led 13 apagado

    // pines de los encoder como entradas
    pinMode(pinENC_chA_d,INPUT);
    pinMode(pinENC_chB_d,INPUT);
    pinMode(pinENC_chA_i,INPUT);
    pinMode(pinENC_chB_i,INPUT);

    // Interrupciones encoder motor derecho
    attachInterrupt(digitalPinToInterrupt(pinENC_chA_d),encoder_chA_d,CHANGE);
    attachInterrupt(digitalPinToInterrupt(pinENC_chB_d),encoder_chB_d,CHANGE);

    // Interrupciones encoder motor izquierdo
    attachInterrupt(digitalPinToInterrupt(pinENC_chA_i),encoder_chA_i,CHANGE);
    attachInterrupt(digitalPinToInterrupt(pinENC_chB_i),encoder_chB_i,CHANGE);

    // Salidas digitales y PWM para control de motores
    pinMode(pinIN4_d, OUTPUT);
    pinMode(pinIN3_d, OUTPUT);
    pinMode(pinENB_d, OUTPUT);
    pinMode(pinIN1_i, OUTPUT);
    pinMode(pinIN2_i, OUTPUT);
    pinMode(pinENA_i, OUTPUT);

    analogWrite(pinENB_d,0) ;
    analogWrite(pinENA_i,0) ;

}

} // fin setup

void loop()
{
    // lectura serie
    char lectura;
    if(Serial.available()>0){ // si se ha recibido algún dato y está en paro
        lectura=Serial.read(); // lee un caracter
        if(lectura=='S'){
            paro=true;
        }else if((lectura=='O')&&(paro)){ // si se lee 0

            cont=1;
            Serial.println("HC*");
            paro=false;
            theta=0.0;
            int i=0;
            while(i<np){

```

```

        // trayectoria circular
        X[i]=0.5*sin(2*PI*i/(float)np);
        Y[i]=0.5*cos(2*PI*i/(float)np)-0.5;

        i++;
    }// fin while

}else if ((lectura=='I')&&(paro)){ // si se lee I

    cont=1;
    Serial.println("HC*");
    paro=false;
    theta=0.0;
    int i=0;
    while(i<np){

        // trayectoria infinito
        X[i]=1*sin(2*PI*i/(float)np);
        Y[i]=0.5*sin(4*PI*i/(float)np);

        i++;
    }// fin while

}else if((lectura=='C')&&(paro)){ // si se lee C

    cont=1;
    Serial.println("HC*");
    paro=false;
    theta=0.0;
    int i=0;
    while(i<np){

        // trayectoria cuadrada
        if (i < np/4){

            X[i]=0+(1-0)*i/((float)np/4);
            Y[i]=0;

        }else if (i < np/2){

            X[i]=1;
            Y[i]=(1-0)*(i-(np/4))/((float)np/4);

        }else if (i < np*3/4){

            X[i]=1-(1-0)*(i-2*(np/4))/((float)np/4);
            Y[i]=1;

        }else{

            X[i]=0;
            Y[i]=1-(1-0)*(i-3*(np/4))/((float)np/4);

        }// fin if cuadrado

        i++;
    }// fin while

}

}

// lectura de la velocidad angular en z
uint8_t Buf[14];
I2Cread(MPU9250_ADDRESS, 0x3B, 14, Buf);

int16_t gz = Buf[12] << 8 | Buf[13];

```

```

// paso a grados por segundo
float gzf=(float)gz*2000.0/32768.0;

// cálculo de theta (no se tienen en cuenta las pequeñas velocidades ->
error del sensor)
if(gzf>=1.5||gzf<=-1.5)
    theta=theta+gzf*T/1000.0;

// paso de theta a radianes
float theta_rad=theta*PI/180.0;

// distancia al punto objetivo
float distancia=sqrt((X[cont]-x)*(X[cont]-x)+(Y[cont]-y)*(Y[cont]-y));

// ángulo entre el punto en que está el robot y el punto objetivo
theta_ref=atan2((Y[cont]-y),(X[cont]-x));

// error en theta
float error_theta=theta_ref-theta_rad;

// límites de theta entre -pi y pi
if(error_theta>PI)
    error_theta=error_theta-2*PI;
else if(error_theta<-PI)
    error_theta=error_theta+2*PI;

// acción de control para corregir el ángulo
float acc_theta=kp_theta*error_theta;

// si no está paro a true envia acciones de control
if(!paro){
    ref_d=0.15/0.045+10*acc_theta;
    ref_d=limites_ref(ref_d);
    ref_i=0.15/0.045-10*acc_theta;
    ref_i=limites_ref(ref_i);
}else{ //si está paro a true se detiene
    ref_d=0;
    ref_i=0;
}

if(distancia<0.1){ // distancia menor de 10 cm
    cont++; // pasa al siguiente punto de la trayectoria
    // envía trama de datos para el gráfico de la aplicación
    Serial.print("HX");Serial.print(x*100);Serial.print("Y");
    Serial.print(y*100);Serial.print("");
}

if(cont>=np){ // si ha llegado al punto final
    paro=true; // paro a true
}

// cálculo de la posición en x e y
x=x+(vel_i+vel_d)*radio_rueda*radio_esfera/(2*h)*cos(theta*3.1416/180)*Tm;
y=y+(vel_i+vel_d)*radio_rueda*radio_esfera/(2*h)*sin(theta*3.1416/180)*Tm;

enc_act_d=quad_d; // valor actual encoder rueda derecha
// diferencia entre el valor actual y el anterior encoder rueda derecha
enc_dif_d=enc_act_d-enc_ant_d;
// cálculo de la velocidad de la rueda derecha en rad/s
vel_d=(enc_act_d-enc_ant_d)*C2RS;
// actualización del valor del encoder actual rueda derecha
enc_ant_d=enc_act_d;

enc_act_i=quad_i; // valor actual encoder rueda izquierda
// diferencia entre el valor actual y el anterior encoder rueda izquierda
enc_dif_i=enc_act_i-enc_ant_i;
// cálculo de la velocidad de la rueda izquierda en rad/s
vel_i=(enc_act_i-enc_ant_i)*C2RS;

```

```

// actualización del valor del encoder actual rueda izquierda
enc_ant_i=enc_act_i;

//Señal de error rueda derecha
error_d=ref_d-vel_d;

//Integral del error rueda derecha
integral_d=error_d*Tm+integral_anterior_d;

//Acción de control rueda derecha
acc_d=kp_d*error_d+ki_d*integral_d;

// Actualización del error acumulado anterior rueda derecha
integral_anterior_d=integral_d;

//Señal de error rueda izquierda
error_i=ref_i-vel_i;

//Integral del error rueda izquierda
integral_i=error_i*Tm+integral_anterior_i;

//Acción de control rueda izquierda
acc_i=kp_i*error_i+ki_i*integral_i;

// Actualización del error acumulado anterior rueda izquierda
integral_anterior_i=integral_i;

// límites acción de control (rueda derecha)
if (acc_d>=MAXU){
    acc_d=MAXU;
}

if (acc_d<=MINU){
    acc_d=MINU;
}

if (acc_d>=0) { // acción de control positiva (rueda derecha)
    digitalWrite(pinIN4_d, HIGH); // sentido de giro positivo (rueda derecha)
    digitalWrite(pinIN3_d, LOW);
} else { // caso contrario
    digitalWrite(pinIN4_d, LOW); //sentido de giro negativo (rueda derecha)
    digitalWrite(pinIN3_d, HIGH);
}

// límites acción de control (rueda izquierda)
if (acc_i>=MAXU){
    acc_i=MAXU;
}
if (acc_i<=MINU){
    acc_i=MINU;
}

if (acc_i>=0) { // acción de control positiva (rueda izquierda)
    digitalWrite(pinIN1_i, HIGH); // sentido de giro positivo (rueda
izquierda)
    digitalWrite(pinIN2_i, LOW);
} else { // caso contrario
    digitalWrite(pinIN1_i, LOW); //sentido de giro negativo (rueda izquierda)
    digitalWrite(pinIN2_i, HIGH);
}

// si la acción de control es nula, la velocidad se fuerza a cero
if(ref_i==0) acc_i=0;
if(ref_d==0) acc_d=0;

// envía la acción de control a los pines ENB y ENA del puente en H
analogWrite(pinENB_d, (byte) (abs(acc_d)*255.0/MAXU)) ;
analogWrite(pinENA_i, (byte) (abs(acc_i)*255.0/MAXU)) ;

```

```

t2=millis(); // mide el tiempo
//Si hay retardo se enciende el led
if (t2-t1>T) {digitalWrite(RT,HIGH) ;}
else {digitalWrite(RT,LOW) ;}
//Espera al siguiente periodo de muestreo
while (t2-t1<T){
  t2=millis();
}
t1=millis(); // actualiza t1

}

// función límites_ref
float limites_ref(float ref){
  if(ref>MAXW){
    ref=MAXW;
  }else if(ref<MINW){
    ref=MINW;
  }
  return ref;
}

// funciones para leer los encoder (por interrupción)
void encoder_chA_d (){
  if(digitalRead(pinENC_chA_d)==digitalRead(pinENC_chB_d)){
    quad_d++;
  }else{
    quad_d--;
  }
}
void encoder_chB_d (){
  if(digitalRead(pinENC_chA_d)==digitalRead(pinENC_chB_d)){
    quad_d--;
  }else{
    quad_d++;
  }
}
void encoder_chA_i (){
  if(digitalRead(pinENC_chA_i)==digitalRead(pinENC_chB_i)){
    quad_i++;}
  else{
    quad_i--;
  }
}
void encoder_chB_i (){
  if(digitalRead(pinENC_chA_i)==digitalRead(pinENC_chB_i)){
    quad_i--;
  }else{
    quad_i++;
  }
}

//Función auxiliar lectura
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data)
{
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.endTransmission();

  Wire.requestFrom(Address, Nbytes);
  uint8_t index = 0;
  while (Wire.available())
    Data[index++] = Wire.read();
}

// Función auxiliar de escritura

```



```
void I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data)
{
    Wire.beginTransaction(Address);
    Wire.write(Register);
    Wire.write(Data);
    Wire.endTransmission();
}
```



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

Documento II:

PLANOS

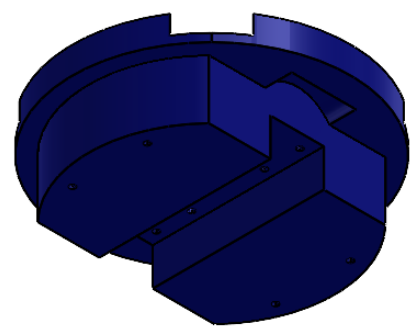
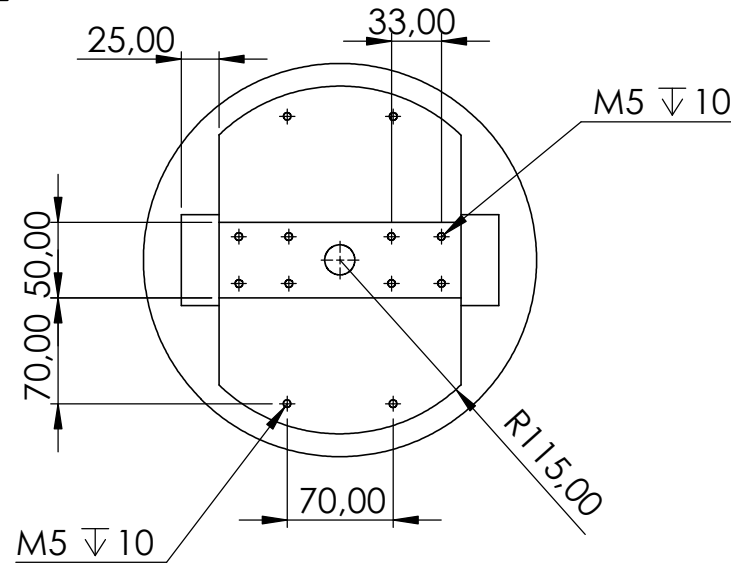
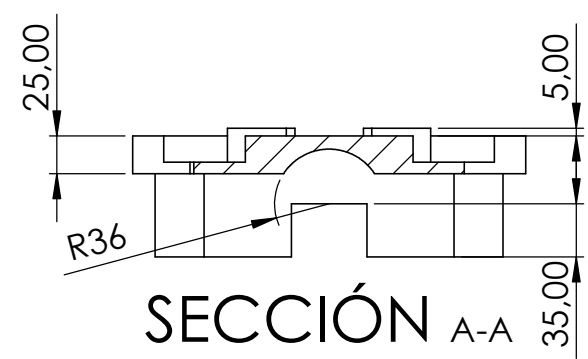
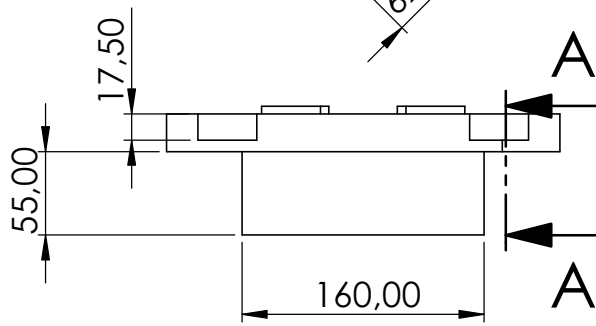
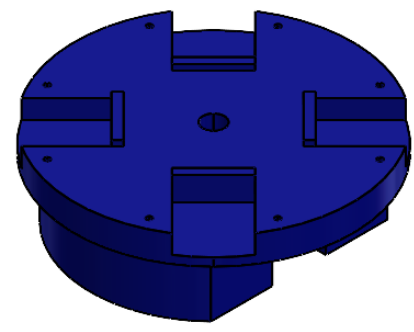
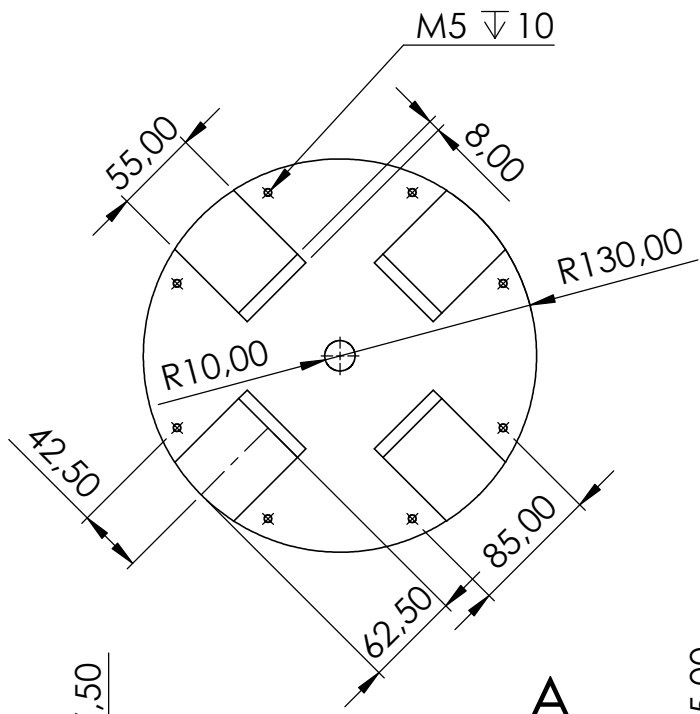
AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Índice

Plano 01: Chasis del vehículo	133
Plano 02: Parte inferior del vehículo	134
Plano 03: Soporte para rueda esférica	135
Plano 04: Soporte para motor	136
Plano 05: Sujeción para soporte de rueda esférica	137
Plano 06: Despiece del ensamblaje	138
Plano 07: Despiece del conjunto motor, rueda y soporte	139



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



FECHA: 14/06/2024

TÍTULO:

Chasis del
Vehículo

PROYECTO:

Diseño, implementación y control
de un prototipo de SpheroBot

AUTOR:

Luis Martínez Martínez

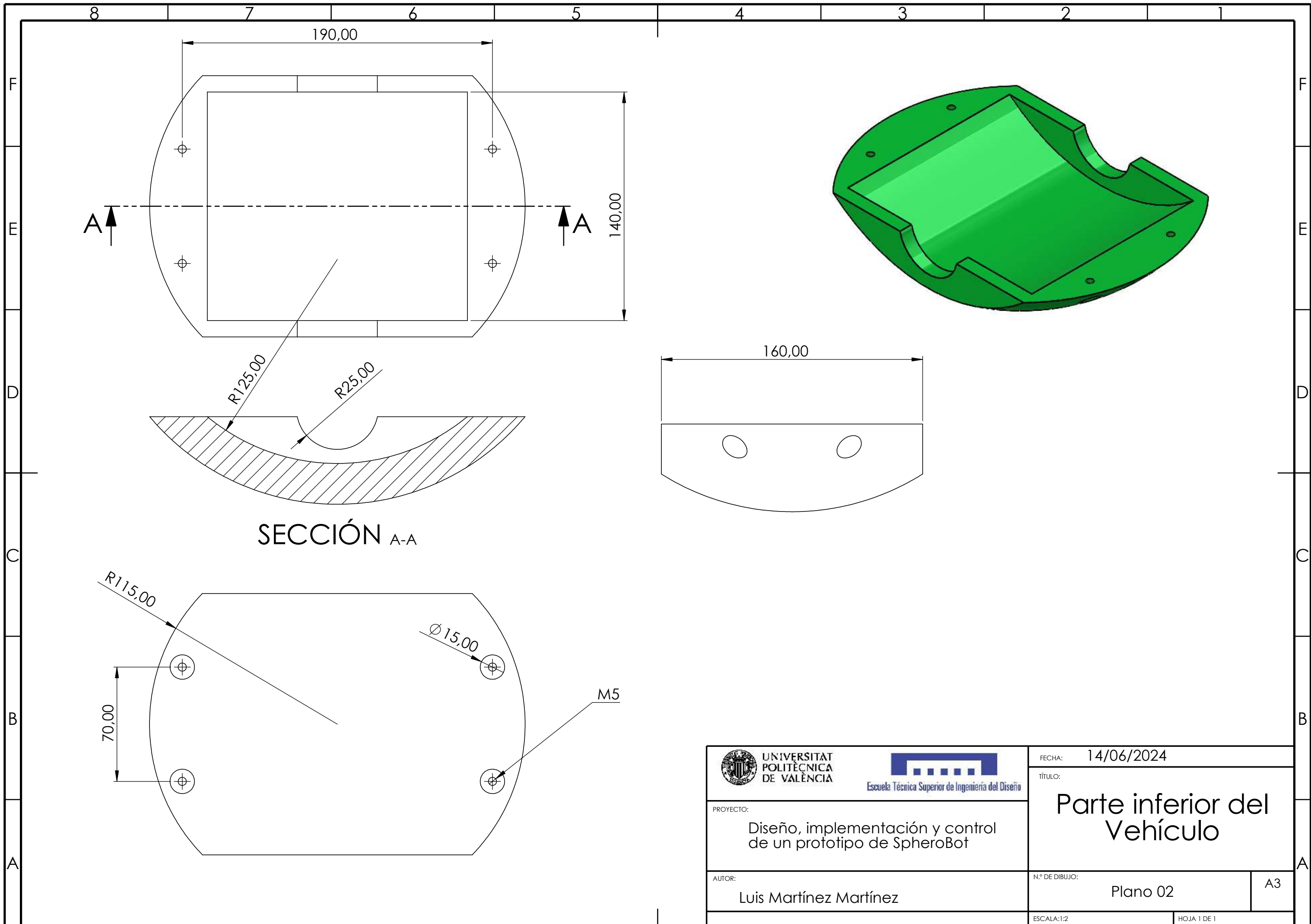
N.º DE DIBUJO:

Plano 01



A4

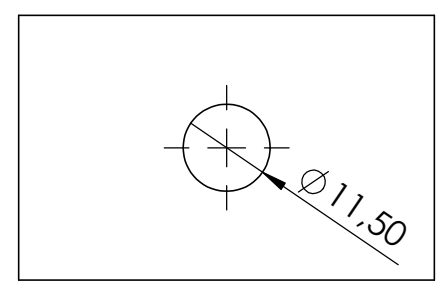
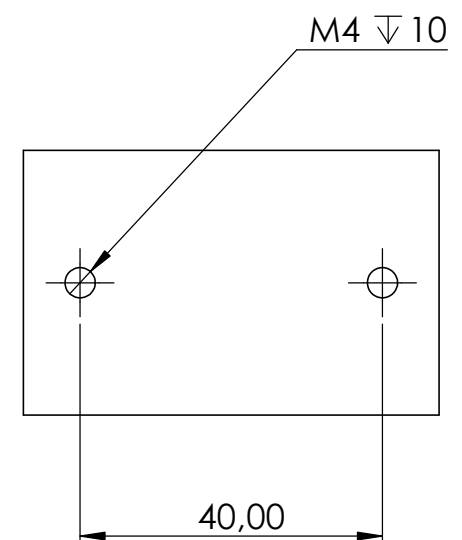
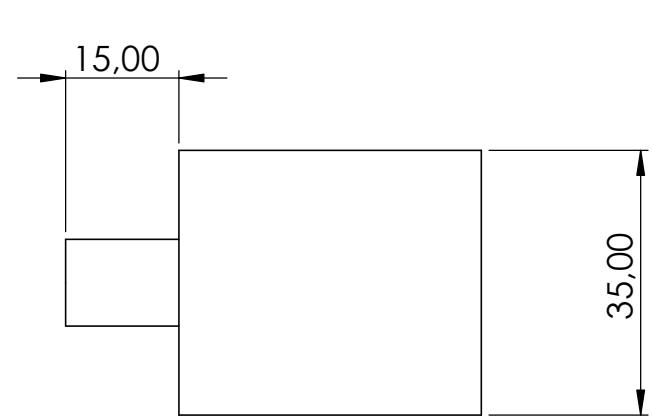
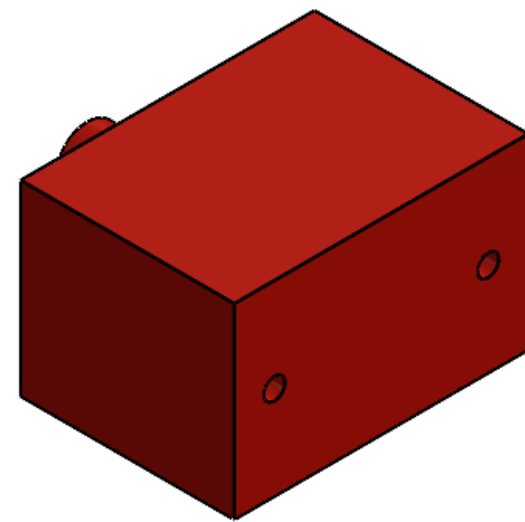
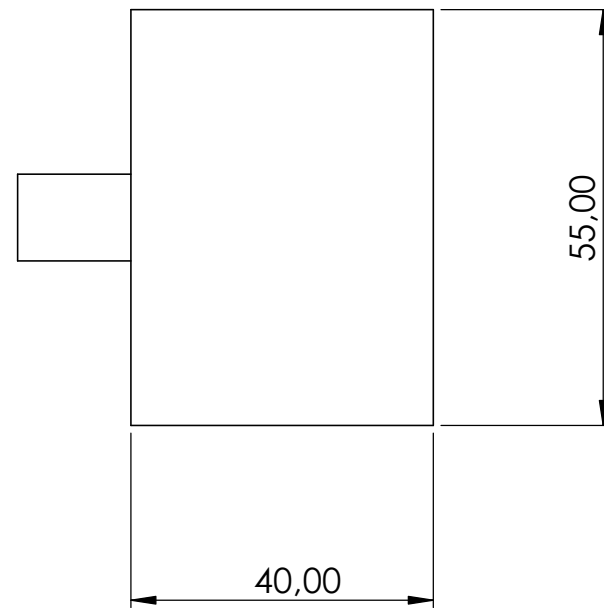
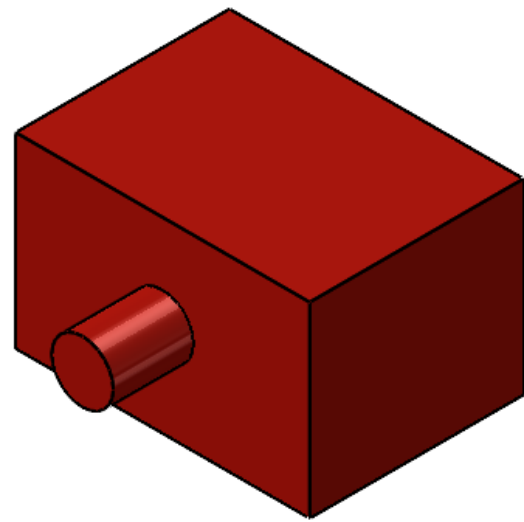
ESCALA:1:5

HOJA 1 DE 1

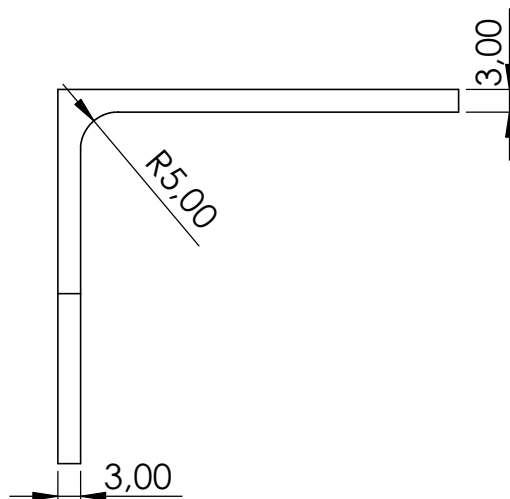
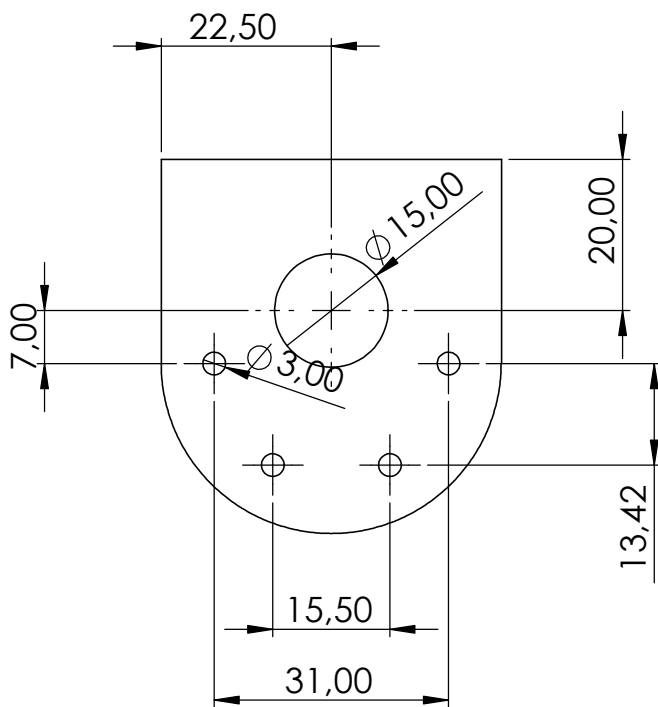
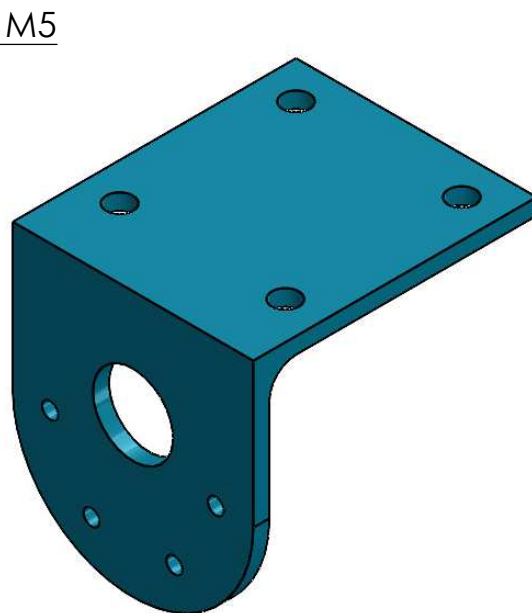
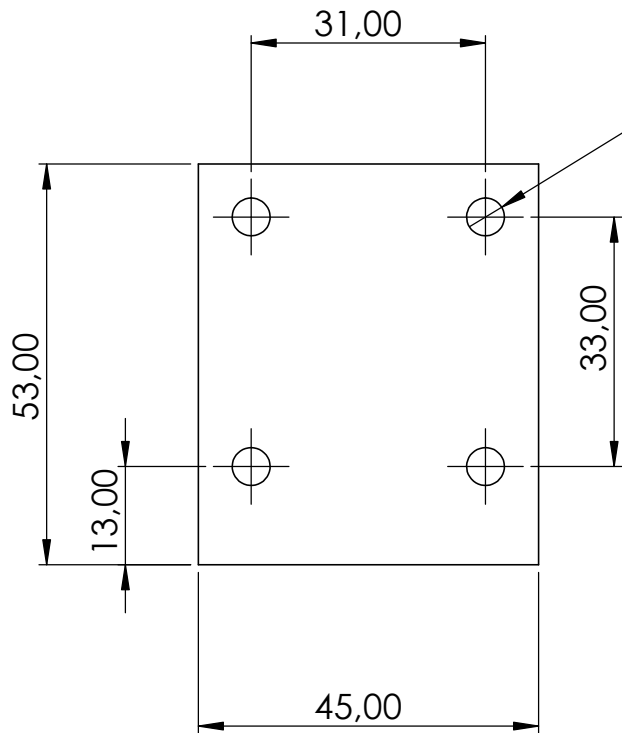


SECCIÓN A-A

 UNIVERSITAT POLITÈCNICA DE VALÈNCIA  Escuela Técnica Superior de Ingeniería del Diseño	FECHA: 14/06/2024	
	TÍTULO: Parte inferior del Vehículo	
PROYECTO: Diseño, implementación y control de un prototipo de SpheroBot	N.º DE DIBUJO: Plano 02	
AUTOR: Luis Martínez Martínez	A3	
ESCALA: 1:2	HOJA 1 DE 1	



 UNIVERSITAT POLITÈCNICA DE VALÈNCIA <i>Escuela Técnica Superior de Ingeniería del Diseño</i>	FECHA: 14/06/2024	
	TÍTULO: Soporte para rueda esférica	
PROYECTO: Diseño, implementación y control de un prototipo de SpheroBot	N.º DE DIBUJO: Plano 03	
AUTOR: Luis Martínez Martínez	ESCALA: 1:1	A3
		HOJA 1 DE 1



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

FECHA: 14/06/2024

TÍTULO:

Soporte para
motor

PROYECTO:

Diseño, implementación y control
de un prototipo de SpheroBot

AUTOR:

Luis Martínez Martínez

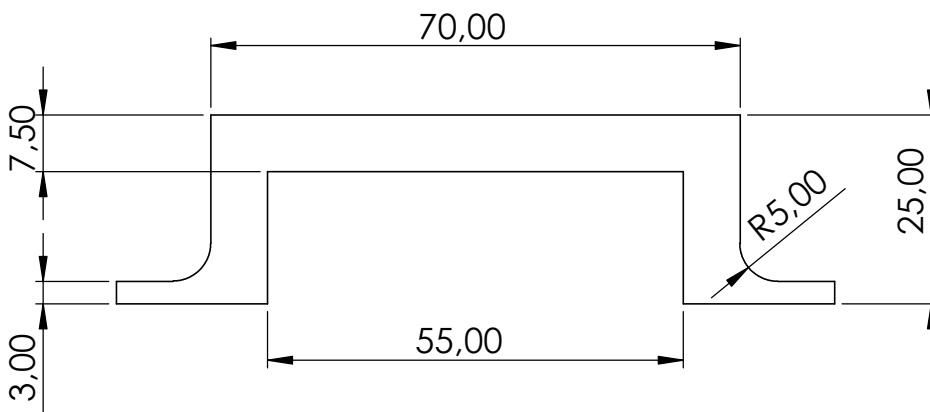
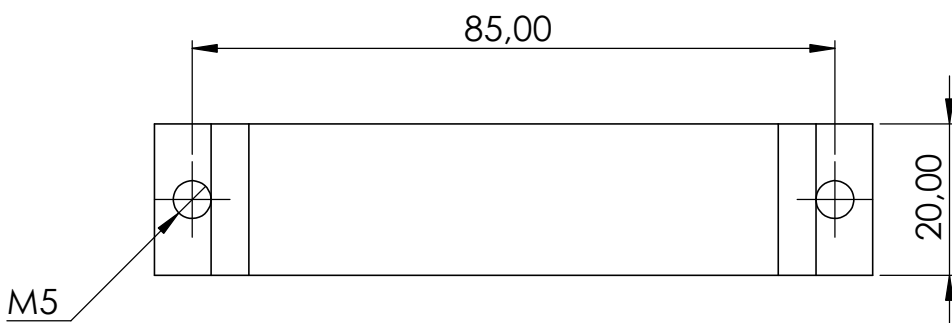
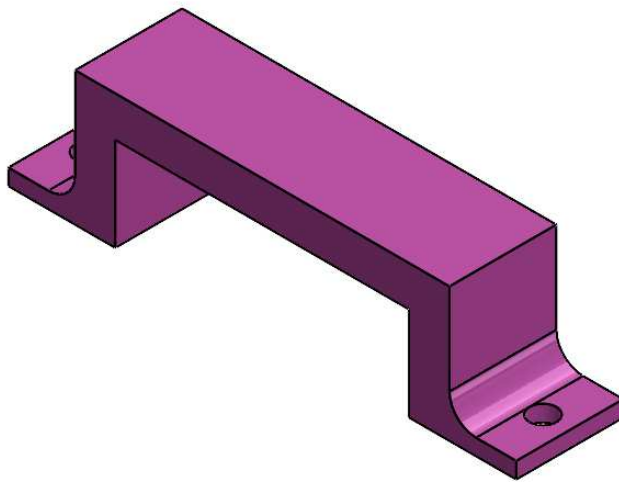
N.º DE DIBUJO:

Plano 04

A4

ESCALA:1:1

HOJA 1 DE 1



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escuela Técnica Superior de Ingeniería del Diseño

FECHA: 14/06/2024

TÍTULO:
**Sujeción para
soporte de rueda
esférica**

PROYECTO:
Diseño, implementación y control
de un prototipo de SpheroBot

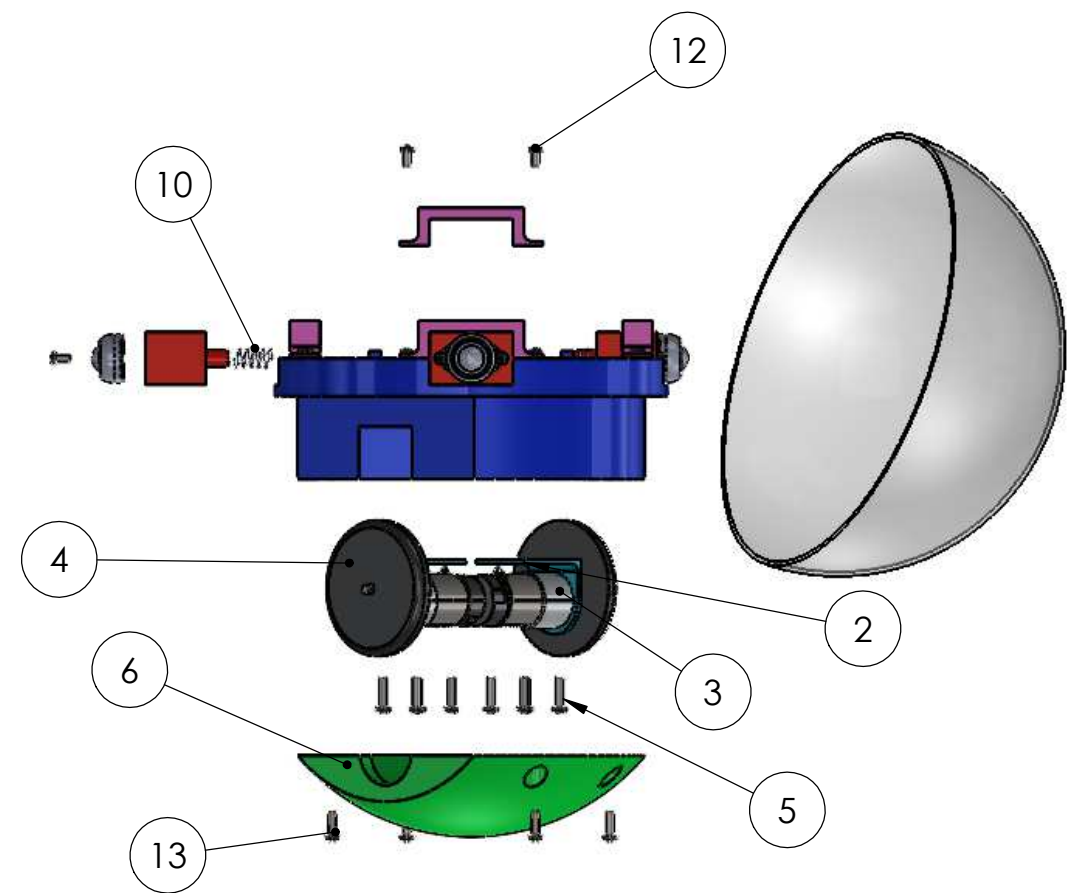
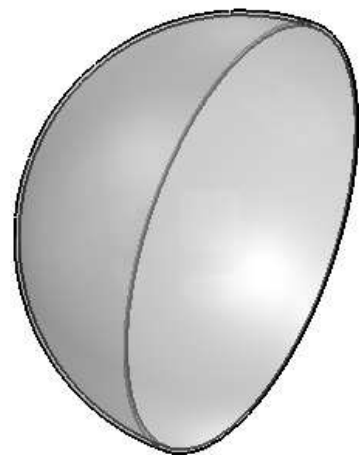
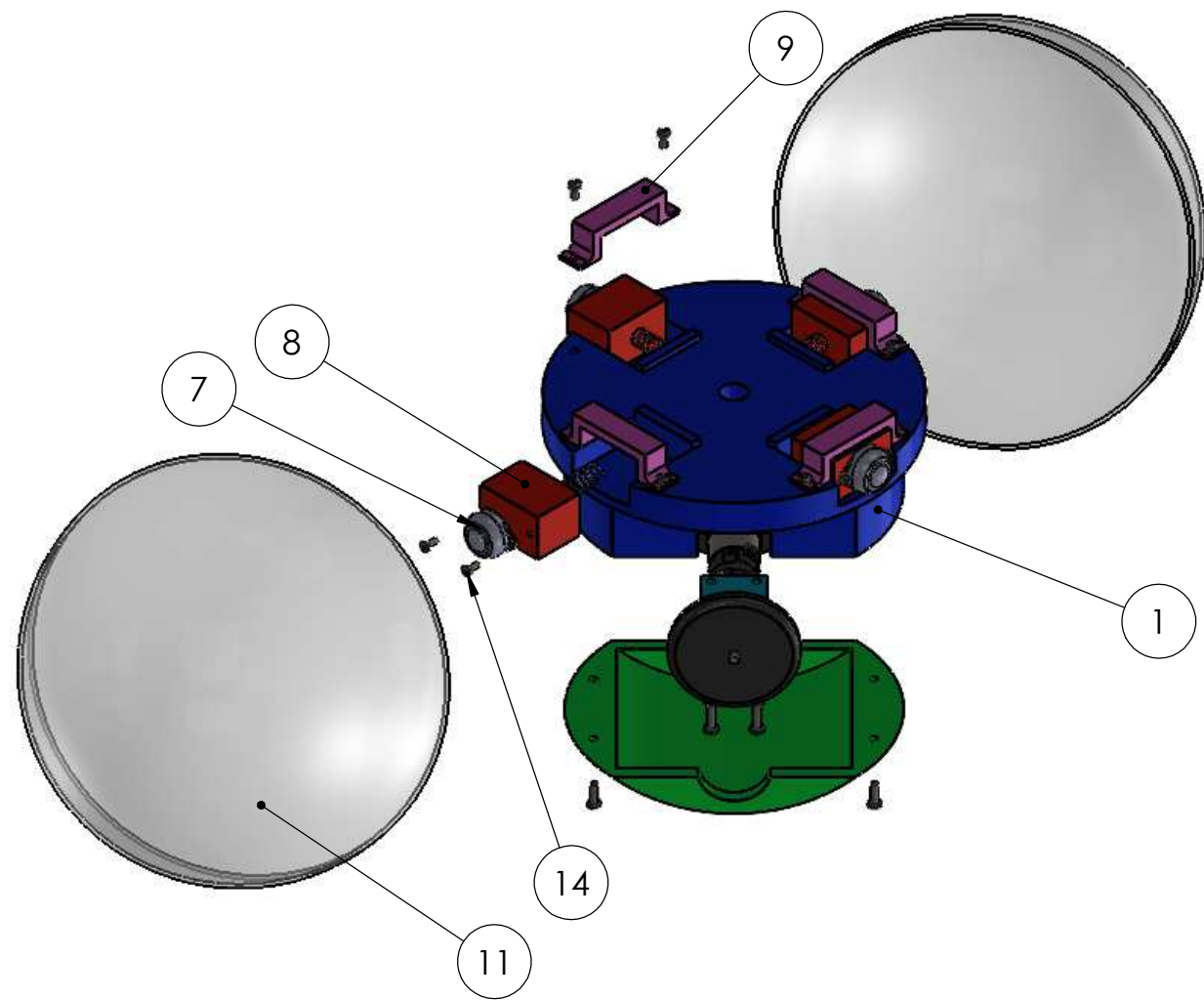
AUTOR:
Luis Martínez Martínez

N.º DE DIBUJO:
Plano 05

A4

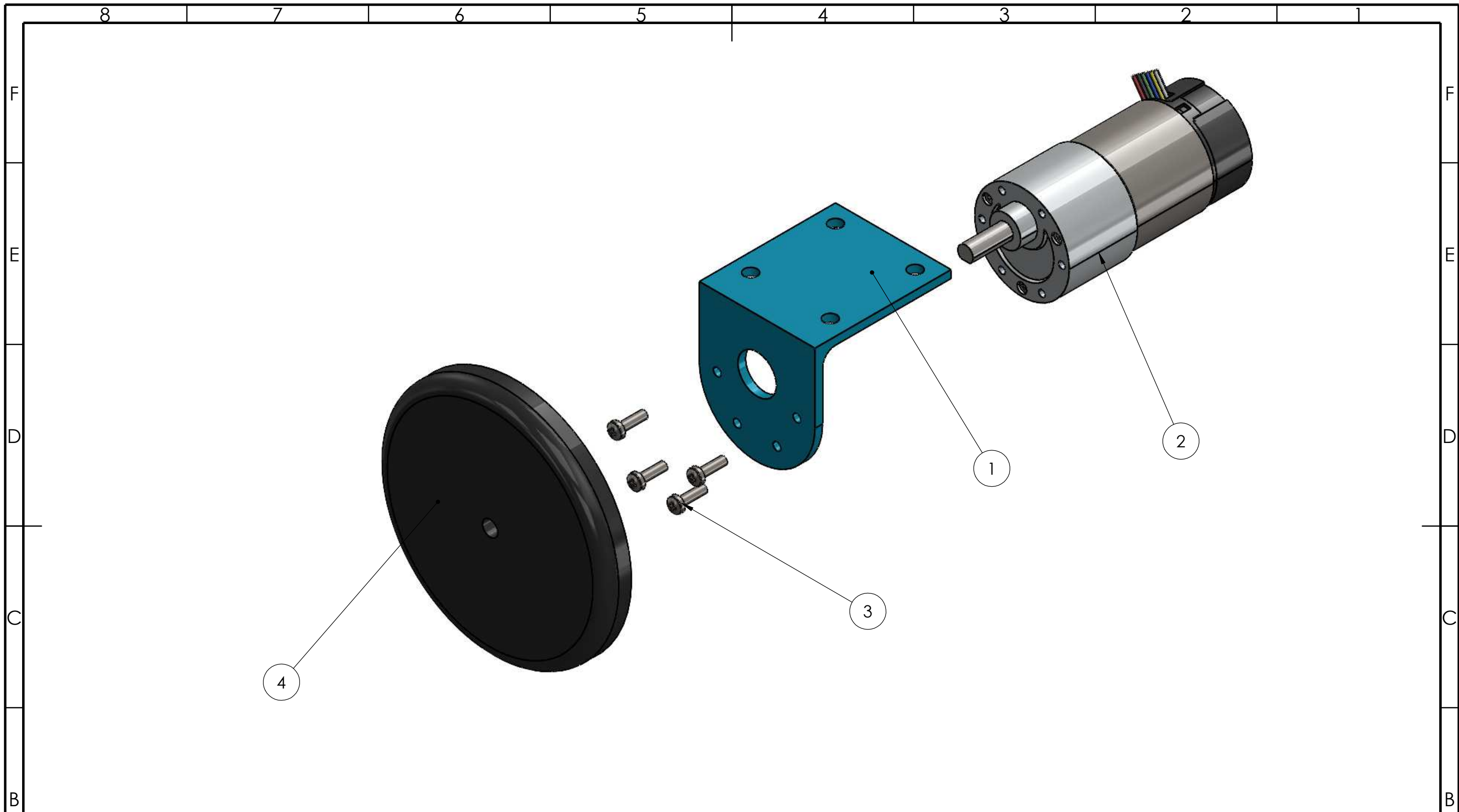
ESCALA:1:1

HOJA 1 DE 1



1	Chasis del vehículo
2	Soporte para motor
3	Motor Pololu 37Dx70L mm
4	Rueda
5	Tornillo M5x25 mm
6	Parte inferior del vehículo
7	Rueda esférica
8	Soporte para rueda esférica
9	Sujeción para soporte para rueda esférica
10	Muelle
11	Semiesfera
12	Tornillo M5x15 mm
13	Tornillo M5x25 mm
14	Tornillo M4x15 mm

 UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño	FECHA: 14/06/2024	
	TÍTULO: Despiece del ensamblaje	
PROYECTO: Diseño, implementación y control de un prototipo de SpheroBot	N.º DE DIBUJO: Plano 06	
AUTOR: Luis Martínez Martínez	A3	
ESCALA: 1:5	HOJA 1 DE 1	



1	Soporte para motor
2	Motor Pololu 37Dx70L mm
3	Tornillo M3x15 mm
4	Rueda

 UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño	FECHA: 14/06/2024
	TÍTULO: Despiece del conjunto motor, rueda y soporte
PROYECTO: Diseño, implementación y control de un prototipo de SpheroBot	N.º DE DIBUJO: Plano 07
AUTOR: Luis Martínez Martínez	A3
ESCALA: 1:1	HOJA 1 DE 1



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

DOCUMENTO III:

PLIEGO DE CONDICIONES

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Índice

1. Objeto	142
2. Condiciones generales	142
3. Especificaciones técnicas	142
3.1. Piezas mecánicas.....	142
3.1.1. Materiales del prototipo	142
3.1.2. Muelles	142
3.1.3. Esfera.....	142
3.1.4. Ruedas.....	142
3.2. Componentes electrónicos	143
3.2.1. Motores.....	143
3.2.2. Microcontrolador	143
3.2.3. Sensor para medir orientación	143
3.2.4. Puente en H.....	143
3.2.5. Módulo Bluetooth	143
3.2.6. Batería	143
3.3. Ensamblaje	143
3.3.1. Ruedas.....	143
3.3.2. Batería	143
3.3.3. Soportes de las ruedas esféricas.....	144
3.3.4. Tornillos	144
3.4. Carga de la batería	144
3.5. Control	144
4. Pruebas de funcionamiento	144

1. Objeto

El presente pliego de condiciones se refiere al montaje, conexionado y puesta en funcionamiento del prototipo de *SpheroBot* diseñado en este proyecto. Este documento se centra en las especificaciones técnicas de los materiales y componentes, en el montaje, en el control y en la carga de la batería.

2. Condiciones generales

La normativa legal por la cual se rige este trabajo es la Normativa de trabajos de fin de grado y trabajos de fin de máster de la Universitat Politècnica de València, aprobada por el Consejo de Gobierno el 21 de julio de 2022.

3. Especificaciones técnicas

3.1. Piezas mecánicas

3.1.1. Materiales del prototipo

Las piezas diseñadas en SolidWorks e impresas mediante impresora 3D, es decir, el chasis del vehículo, la parte inferior de este, los soportes para los motores, los soportes para las ruedas esféricas y los puentes que sujetan estas últimas al chasis, se han de imprimir utilizando PLA (ácido poliláctico). Este material es ligero y resistente, y además se puede lijar y taladrar con facilidad, lo que es de utilidad a la hora de realizar el ensamblaje.

3.1.2. Muelles

Para garantizar el contacto de las ruedas esféricas con la esfera que envuelve el vehículo se usan muelles de diámetro interno de 5 mm. No se deben usar muelles con elevadas constantes elásticas, pues no interesa que ejerzan una fuerza excesiva sobre la esfera, ya que podrían llegar a abrirla.

3.1.3. Esfera

La esfera utilizada es de metacrilato transparente y debe ser divisible en dos semiesferas que encajen entre sí de forma tal que las superficies interna y externa de la esfera queden totalmente lisas. Esto último es de vital importancia para el correcto funcionamiento del robot. En cuanto a la posibilidad de abrir y cerrar la esfera con facilidad también es de gran relevancia, pues es necesario acceder al interior de la esfera para la recarga de la batería y la programación del microcontrolador.

3.1.4. Ruedas

Las ruedas utilizadas son de plástico, pero es necesario que incorporen un recubrimiento de goma en la parte externa, que entra en contacto con la esfera, para evitar que deslicen.

3.2. Componentes electrónicos

3.2.1. Motores

Los motores que se usan para este proyecto son de 12 V de corriente continua modelo 37D de la marca Pololu con encoder. Existen diferentes opciones con distinta relación de reducción. En este caso se usan motores con reducción 50:1.

3.2.2. Microcontrolador

El microcontrolador utilizado debe tener al menos cuatro pines de interrupciones para realizar la lectura de los dos encoders. Por este motivo se ha utilizado la placa Arduino Mega. Además, debe tener la memoria y la velocidad suficientes para la ejecución del programa y comunicación I2C para el sensor y USART para el módulo Bluetooth, así como salidas PWM para el control de los motores. Todas estas características las ha de cumplir el microcontrolador empleado.

3.2.3. Sensor para medir orientación

Es necesario un módulo con un giroscopio para poder medir la orientación en el plano XY . El módulo MPU-9250 dispone de un giroscopio, además de un acelerómetro y un magnetómetro.

3.2.4. Puente en H

Para el control de los motores se requiere un puente en H, que posea una entrada de 12 V, que es la tensión de los motores utilizados. Con este fin se opta por el módulo L298N. Es imprescindible recordar retirar los jumpers de los pines ENA y ENB, ya que, de no hacerlo, los motores girarán a su velocidad máxima en todo momento, imposibilitado el control de su velocidad. Otras variantes de este módulo como el L298P también serían válidas.

3.2.5. Módulo Bluetooth

En el desarrollo de este proyecto se ha utilizado el módulo Bluetooth HC-06. Podrían usarse otros módulos similares como el HC-05. Es preciso tener en cuenta que el programa funciona a 9600 baudios, y debe coincidir con la velocidad a la que trabaje este módulo.

3.2.6. Batería

Hace falta una batería de 12 V para la alimentación de los motores y del resto de la electrónica. La batería debe ser recargable mediante su conexión a la red eléctrica.

3.3. Ensamblaje

3.3.1. Ruedas

Es fundamental que las ruedas queden fijas a los ejes de los motores. Por ello se ha de utilizar una pieza metálica con la forma del eje que se insertará en el centro de la rueda. También puede pegarse la rueda al eje mediante pegamento.

3.3.2. Batería

La batería se ubicará en la parte inferior del chasis, bajo los motores y sobre la pieza diseñada para cubrir la parte inferior del vehículo. El cable deberá atravesar el chasis por el conducto creado a tal efecto para alimentar el puente en H y la placa, que se sitúan sobre

el chasis. Con esto se busca, no solo ahorrar espacio sino también mantener bajo el centro de gravedad del robot para darle mayor estabilidad.

3.3.3. Soportes de las ruedas esféricas

Para garantizar el contacto entre las ruedas esféricas y la esfera es preciso, además del muelle, que las piezas que soportan estas ruedas puedan deslizarse sobre el chasis hacia delante y hacia atrás, por lo que, en caso de que no deslicen, se deben lijar hasta que lo hagan con suavidad.

3.3.4. Tornillos

Para el ensamblaje de las piezas se requieren de diferentes tornillos, que se detallan a continuación.

- Ocho tornillos M5x15 mm para unir los puentes que sujetan las piezas que llevan las ruedas esféricas al chasis.
- Ocho tornillos M4x15 mm para unir las ruedas esféricas a las piezas que las sujetan.
- Doce tornillos M5x25 mm, de los cuales ocho se utilizan para unir los soportes de los motores al chasis y cuatro para unir la parte inferior del vehículo al resto al chasis.
- Ocho tornillos M3x15 mm para sujetar los motores a sus soportes.
- Tornillos M3x15mm para sujetar al chasis los componentes electrónicos.

Salvo los tornillos que sujetan los motores a sus soportes, es necesario que sean acabados en punta aguda, para que queden bien fijos a las piezas.

3.4. Carga de la batería

Para la carga de la batería es preciso abrir la esfera, extraer el vehículo y desatornillar la su pieza interior para poder acceder a ella. Con el botón de la batería en ON se conecta a la red. La luz del enchufe permanece encendida en color rojo mientras está en carga y cambia a verde cuando la carga se ha completado. El tiempo estimado de carga completa es de 8 horas aproximadamente.

3.5. Control

Para el control remoto del robot se ha de tener en cuenta el alcance del módulo Bluetooth, que es de 10 m según el fabricante. Por tanto, el usuario debe permanecer a una distancia menor del robot en todo momento para no perder la conexión.

Por otra parte, se recomienda detener al prototipo reduciendo la velocidad mediante las flechas hasta llegar a 0. El botón de paro se debe limitar a situaciones de emergencia, pues provoca un cabeceo en el robot que perdura unos segundos.

4. Pruebas de funcionamiento

Tras el montaje del robot se deben realizar diversas pruebas para verificar su correcto funcionamiento. Para ello, es preciso instalar la aplicación Android para el control del robot. Primeramente, se debe probar el control remoto, para verificar el giro de los motores, su correcto sentido de giro y el funcionamiento del control, que ha de ser capaz de alcanzar

la referencia indicada. Esta primera prueba se hará con el vehículo fuera de la esfera y asegurándose de que sus las ruedas no tocan ninguna superficie

Después se colocará el vehículo en la esfera y se dejará sobre el suelo para probar su control remoto. Se deben probar todos los movimientos posibles, es decir, movimiento en línea recta hacia delante y hacia atrás, giros a la derecha y a la izquierda, hacia delante y hacia atrás, con ambas ruedas girando en el mismo sentido y en sentido opuesto, y giro del robot sobre sí mismo. Estas pruebas deben llevarse a cabo además con distintas velocidades. También debe verificarse el botón de paro, comprobando que se detiene el robot de inmediato.

Una vez verificado el control de las ruedas vía Bluetooth, se probará el seguimiento de trayectorias, comprobando que la orientación y las distancias recorridas por el robot son en todo momento las marcadas por la trayectoria definida. Se ha de comprobar el funcionamiento de las tres trayectorias disponibles en la aplicación, del botón de paro y del gráfico que representa el camino seguido.



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño, implementación y control de un prototipo de SpheroBot

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

DOCUMENTO IV:

PRESUPUESTO

AUTOR/A: Martínez Martínez, Luis

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Índice

1. Costes de materiales	148
2. Costes de personal	148
3. Costes de equipos y licencias.....	149
4. Presupuesto total del Proyecto	149
5. Presupuesto de un prototipo de SpheroBot	149

A continuación, se detalla el presupuesto de este proyecto.

1. Costes de materiales

En primer lugar, en la Tabla 1 se indican los costes de los materiales empleados para la construcción del prototipo (IVA no incluido).

Materiales			
Descripción	uds	Coste unitario (€/ud)	Coste total
Esfera de plástico divisible en dos semiesferas	1	23,02 €	23,02 €
Rueda loca esférica	4	1,09 €	4,38 €
Rueda de 90 mm de diámetro	2	9,36 €	18,72 €
Muelles	4	0,99 €	3,95 €
Tornillería	1	1,78 €	1,78 €
Motor Pololu 37D	2	47,30 €	94,59 €
Módulo Bluetooth HC-06	1	8,68 €	8,68 €
Arduino MEGA	1	17,37 €	17,37 €
Batería recargable de 12 V DC	1	20,53 €	20,53 €
Módulo de puente en H L298N	1	3,15 €	3,15 €
Conector DC hembra	1	0,51 €	0,51 €
Módulo MPU-9250	1	6,19 €	6,19 €
Cables	1	3,95 €	3,95 €
Rollo de 1 Kg de PLA para la impresión de las piezas	1	14,59 €	14,59 €
Total			221,43 €

Tabla 1. Costes de materiales.

2. Costes de personal

La Tabla 2 muestra los costes de personal asociados a las diferentes fases del trabajo.

Personal			
Descripción	horas	Coste unitario (€/h)	Coste total
Documentación	20	20,00 €	400,00 €
Diseño en SolidWorks	25	25,00 €	625,00 €
Programación en Matlab/Simulink	60	25,00 €	1.500,00 €
Montaje	10	15,00 €	150,00 €
Programación en Arduino	60	30,00 €	1.800,00 €
Redacción del proyecto	100	25,00 €	2.500,00 €
Pruebas y análisis de resultados	25	20,00 €	500,00 €
Total			7.475,00 €

Tabla 2. Costes de personal.

3. Costes de equipos y licencias

En la Tabla 3 figuran los costes de las licencias y equipos necesarios para este proyecto.

Equipos y licencias			
Descripción	uds	Coste unitario (€/ud)	Coste total
Licencia de Matlab y Simulink para estudiantes	1	69,00 €	69,00 €
Simscape Multibody	1	7,00 €	7,00 €
Licencia de SolidWorks para estudiantes	1	45,00 €	45,00 €
Licencia de Microsoft Office 365	1	69,00 €	69,00 €
Ordenador portátil Lenovo Intel Core i7 con 8 GB de RAM	1	768,86 €	768,86 €
Total			958,86 €

Tabla 3. Costes de equipos y licencias.

4. Presupuesto total del Proyecto

La Tabla 4 contiene el presupuesto total del proyecto, incluyendo, además del material y montaje del prototipo, los costes de equipos y licencias, junto con los trabajos de documentación, diseño, simulación, programación y verificación.

Presupuesto Total del Proyecto	
Materiales	215,24 €
Personal	7.475,00 €
Equipos y licencias	958,86 €
Total sin IVA	8.649,10 €
IVA (21%)	1.816,31 €
Total (IVA incluido)	10.465,41 €

Tabla 4. Presupuesto total del proyecto.

5. Presupuesto de un prototipo de SpheroBot

Por último, la Tabla 5 muestra el coste de un prototipo de *SpheroBot*, considerando únicamente los materiales y el montaje, excluyendo equipos, licencias y los trabajos de documentación, diseño, simulación, programación y verificación.

Presupuesto Prototipo	
Materiales	215,24 €
Montaje	150,00 €
Total sin IVA	365,24 €
IVA (21%)	76,70 €
Total (IVA incluido)	441,93 €

Tabla 5. Coste de un prototipo de SpheroBot.

