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 vehículo de autobalanceado dos ruedas tipo SegWay

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Sánchez Martínez, Adrián

Tutor/a: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

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 VEHÍCULO AUTOBALANCEADO DE DOS RUEDAS TIPO SEGWAY

Trabajo de Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR: Sánchez Martínez, Adrián

Tutor: Casanova Calvo, Vicente Fermín

CURSO ACADÉMICO: 2023/2024

Agradecimientos

A mis padres, a mi hermano Hugo y a mis abuelos por su amor incondicional, su constante apoyo y por brindarme la oportunidad de perseguir todos mis sueños. Gracias por estar siempre.

A mis amigos Carlos, David, Alejandro, Kevin, Oscar y Sergio, cuya amistad ha sido un pilar fundamental durante toda mi vida. Los momentos compartidos, las risas y las conversaciones han sido inolvidables

A mis amigos de la universidad, quienes han sido mis compañeros de viaje durante todo este proceso y han estado a mi lado en cada paso del camino. Especialmente a mis amigos Marta, Pablo y Esme, porque me habéis ayudado a divertirme cuando lo he necesitado, pero también habéis sabido estar en todos los momentos difíciles de esta etapa.

A Melissa, por tu constante apoyo y por estar siempre presente cuando la necesitaba. Tu amistad ha sido un pilar muy importante en mi vida.

Finalmente, quiero agradecer a mi tutor, Vicente Casanova, por su orientación experta, su paciencia y su dedicación. Sin su ayuda, este logro no habría sido posible.

Sin vuestra ayuda, nada de esto habría sido posible.

Gracias a todos.

Resumen

El presente Trabajo de final de Grado se basa en la simulación e implementación real mediante el microcontrolador Arduino de un prototipo de robot autobalanceado tipo *Segway*.

El vehículo consigue mantenerse en pie mediante la acción de los dos motores situados en sus ruedas, resolviendo de esta manera uno de los problemas clásicos de la teoría de control, el péndulo invertido. Para ello consta de un sensor IMU que proporciona su ángulo de inclinación respecto a la vertical, de un sistema de control que se encarga de establecer unas determinadas acciones para que el robot se mantenga de pie y de unos actuadores, en este caso los motores, que llevan a cabo dichas acciones.

Se comienza el proyecto creando un diseño en 3D del vehículo mediante el programa *Siemens NX*. Posteriormente se realiza una simulación del movimiento del vehículo mediante la herramienta *Simscape Multibody* del *software Matlab/Simulink*. En la última parte del trabajo se realizará la implementación del prototipo real, empleando el microcontrolador Arduino para el control automático de la estabilidad y el control manual de la trayectoria.

Palabras clave: Segway, Simescape Multibody, Arduino.

Abstract

The present Bachelor's Thesis is based on the simulation and real implementation through the Arduino microcontroller of a Segway-type self-balancing robot prototype.

The vehicle manages to stay upright by the action of the two motors located on its wheels, thus solving one of the classic problems of control theory, the inverted pendulum. For this purpose, it consists of an IMU sensor that provides its inclination angle with respect to the vertical, a control system that establishes certain actions to keep the robot upright, and actuators, in this case the motors, which carry out these actions.

The project begins by creating a 3D design of the vehicle using the *Siemens NX* program. Subsequently, a simulation of the vehicle's movement is performed using the *Simscape Multibody* tool of the *Matlab/Simulink* software. In the final part of the work, the implementation of the real prototype will be carried out, using the Arduino microcontroller for automatic stability control and manual trajectory control.

Paraules clau: Segway, Simescape Multibody, Arduino.

Resum

El present Treball de Fi de Grau es basa en la simulació i implementació real mitjançant el microcontrolador Arduino d'un prototip de robot autoequilibrat tipus *Segway*.

El vehicle aconsegueix mantindre's en peu mitjançant l'acció dels dos motors situats en les seues rodes, resolvent d'aquesta manera un dels problemes clàssics de la teoria de control, el pèndol invertit. Per a això consta d'un sensor IMU que proporciona el seu angle d'inclinació respecte a la vertical, d'un sistema de control que s'encarrega d'establir unes determinades accions perquè el robot es mantinga en peu i d'uns actuadors, en aquest cas els motors, que duen a terme aquestes accions.

Es comença el projecte creant un disseny en 3D del vehicle mitjançant el programa *Siemens NX*. Posteriorment es realitza una simulació del moviment del vehicle mitjançant l'eina *Simscape Multibody* del programa *Matlab/Simulink*. En l'última part del treball es realitzarà la implementació del prototip real, emprant el microcontrolador Arduino per al control automàtic de l'estabilitat i el control manual de la trajectòria.

Keywords: Segway, Simescape Multibody, Arduino.

Índice general

Agradecimientos . . . III
Resumen . . . VII
Abstract . . . IX
Resum . . . XI
Índice de figuras . . . XVII
Índice de tablas . . . XVIII
Lista de acrónimos . . . XIX

1 Memoria **1**
1.1 **Introduccion** . . . 1
1.1.1 Objeto . . . 2
1.1.2 Fases del proyecto . . . 2
1.1.3 Estructura del documento . . . 4
1.2 **Antecedentes** . . . 5
1.2.1 Marco teórico del proyecto . . . 5
1.2.2 Historia y evolución de los robots autobalanceados . . . 8
1.2.3 Principio de funcionamiento de un vehículo autobalanceado . . . 12
1.2.4 Modelo matemático del robot . . . 17
Modelo del péndulo . . . 18
Modelo de la rueda . . . 20
Modelo del motor DC . . . 21
Modelo no lineal . . . 22
Modelo linealizado . . . 23
1.3 **Requerimientos del proyecto y factores a considerar** . . . 24
1.3.1 Especificaciones del proyecto . . . 24
Requerimientos de la simulación . . . 24
Requerimientos de la implementación . . . 25
Limitaciones del Proyecto . . . 25
1.3.2 Normativa aplicable al proyecto . . . 26
1.4 **Planteamiento de alternativas y solución adoptada** . . . 27
1.4.1 Alternativas de diseño . . . 27
1.4.2 Alternativas de control . . . 28
1.4.3 Alternativas de los componentes electrónicos . . . 29
Sensor IMU . . . 29

Motores . . . 30
Módulo control motores (*driver*) . . . 31
Microcontrolador . . . 32
Baterías . . . 33
Módulo Bluetooth . . . 34
1.5 **Simulación del movimiento del robot** . . . 35
1.5.1 Diseño 3D del modelo . . . 35
1.5.2 Control de estabilidad . . . 36
1.5.3 Control manual del robot . . . 43
1.5.4 Control automático de la velocidad lineal y angular . . . 48
1.5.5 Control de trayectorias . . . 55
1.6 **Implementación real del robot** . . . 62
1.6.1 Montaje y funcionamiento de los componentes . . . 63
Sensor GY-521 . . . 63
Motores CC y driver L298N . . . 65
Módulo bluetooth HC-06 . . . 65
Otros materiales utilizados para el montaje . . . 66
1.6.2 Programación del robot . . . 67
Control de la estabilidad . . . 68
Control manual . . . 72
1.7 **Análisis de resultados** . . . 75
1.7.1 Resultados y pruebas de la simulación . . . 75
1.7.2 Resultados de la implementación real . . . 78
1.8 **Conclusiones y mejoras futuras** . . . 81

2 Planos **83**

3 Pliego de condiciones **89**
3.1 Objeto . . . 90
3.2 Condiciones de los materiales . . . 91
3.2.1 Bandejas de la estructura . . . 91
3.2.2 Tornillería y elementos de fijación . . . 91
3.2.3 Baterías . . . 91
3.2.4 Componentes electrónicos . . . 92
3.3 Condiciones de ejecución . . . 93
3.3.1 Estructura del robot . . . 93
3.3.2 Circuito electrónico . . . 93
3.3.3 Programación del robot . . . 94
3.4 Prueba de servicio . . . 95

4 Presupuesto **96**
4.1 Cuadro de precios unitarios . . . 97
4.2 Cuadro de precios descompuestos . . . 99
4.3 Cuadro de mediciones y ejecución del material . . . 101

4.4 Resumen general del presupuesto 102

Anexo A: Especificaciones técnicas de los componentes **103**

Anexo B: Código Arduino para la programación del robot **105**

Anexo C: Objetivos de Desarrollo Sostenible **111**

Referencias **114**

Índice de figuras

1.1 Logos de programas utilizados. 3
1.2 Esquema inicial de las fases del proyecto 3
1.3 Representación de robots en la literatura y el cine del siglo XX. 6
1.4 Elementos de un sistema mecatrónico. 6
1.5 Ejemplos de tipos de robots. 7
1.6 Giros y direcciones según velocidad ruedas en plataforma diferencial. . . . 9
1.7 Robot EMIEW 3 desarrollado por *Hitachi*. 10
1.8 Vehículos autoequilibrados para el transporte. 11
1.9 Balanbot Arduino *Self Balancing Robot*. 12
1.10 Diagrama del cuerpo libre del péndulo simple. 13
1.11 Diagrama del cuerpo libre del péndulo invertido. 14
1.12 Diagrama de bloques para el control del péndulo invertido. 16
1.13 Diagrama del cuerpo libre de la barra del péndulo invertido. 18
1.14 Diagrama del cuerpo libre de la rueda del péndulo invertido. 20
1.15 Circuito eléctrico equivalente de un motor DC. 21
1.16 Módulo GY-521 Acelerómetro y Giroscopio MPU-6050. 30
1.17 Motorreductor 30:1 *Metal Gearmotor* 37Dx52L mm 12V. 31
1.18 *Driver* puente H modelo L298N. 32
1.19 Microcontrolador Arduino Due. 33
1.20 Batería de litio 7,4 V 6000 mAh. 34
1.21 Módulo Bluetooth HC-06. 34
1.22 Fases de la simulación. 35
1.23 Modelo 3D del robot en Siemens NX. 36
1.24 Ventana bloque *File Solid* de *Matlab*. 37
1.25 Ventana del bloque *Spatial Contact Forces* de *Simescape Multibody*. 40
1.26 Diagrama de bloques para el modelado de la estabilidad del robot. 40
1.27 Modelo general para el control de la estabilidad del robot. 41
1.28 Regulador PD para la estabilidad del robot. 42
1.29 Gráfica del ángulo de inclinación respecto al tiempo. 42
1.30 Simulación de la estabilidad del robot. 43
1.31 Funciones del mando para el control manual. 43
1.32 Grados de libertad en las articulaciones para el control manual. 44
1.33 Diagrama de bloques para el modelado del control manual del robot. . . . 45

1.34 Implementación del bloque para el control del *gamepad*. 45
1.35 Modelo general para el control manual del robot. 46
1.36 Gráfica del ángulo de inclinación respecto al tiempo en el control manual. 47
1.37 Gráficas de la posición xy y velocidad angular de las ruedas. 47
1.38 Grados de libertad en las articulaciones para el control automático. 48
1.39 Cálculo de la velocidad relativa al robot. 48
1.40 Cálculo de la velocidad relativa al robot. 49
1.41 Diagrama de bloques para el modelado del control automático del robot. . 49
1.42 Controladores PI y PD para el control de la velocidad angular y lineal. . . 50
1.43 Modelo general para el control automático de la velocidad y el giro. . . . 51
1.44 Valores introducidos en la referencia de los controladores. 52
1.45 Gráficas de la posición xy del robot. 53
1.46 Gráfica del ángulo de inclinación respecto al tiempo en el control automático. 53
1.47 Gráficas de velocidad lineal y angular junto a la referencia introducida. . . 54
1.48 Gráfica de la velocidad angular de las ruedas en el control automático. . . 54
1.49 Diagrama de bloques para el modelado de las trayectorias. 55
1.50 Modelo general para el control de trayectorias del robot. 56
1.51 Subsistema para el cálculo de la posición de destino. 57
1.52 Subsistema para el cálculo de la trayectoria. 57
1.53 Subsistema para normalizar el valor del ángulo de giro. 58
1.54 Vista cenital del plano xy para la visualización de la trayectoria realizada 59
1.55 Gráfica de la posición xy del robot respecto al tiempo. 60
1.56 Gráfica del control de la velocidad angular y lineal. 60
1.57 Visualización y simulación de la trayectoria implementada. 61
1.58 Algoritmo de las Curvas de Lissajous. 61
1.59 Modelo 3D del robot diseñado en *Inventor* con los componentes electrónicos. 62
1.60 Esquema montaje y alimentaciones componentes. 63
1.61 Ejes sensor MPU-6050. 64
1.62 Conexiones sensor MPU-6050. 64
1.63 Conexiones de los motores y el driver al microcontrolador. 65
1.64 Conexiones del módulo bluetooth HC-06 al microcontrolador. 66
1.65 Materiales para el montaje de las ruedas y los motores. 66
1.66 Diagrama de flujo para la programación del robot. 67
1.67 Gráfica para visualizar el ángulo de inclinación y la estabilidad del robot. 72
1.68 Interfaz aplicación *Bluetooth Electronics* para el control manual. 73
1.69 Pruebas realizadas para la estabilidad y el control manual. 76
1.70 Pruebas realizadas para los diferentes tipos de controles. 77
1.71 Modelo final del vehículo autobalanceado tipo *Segway*. 79
1.72 Trayectoria en forma de *zigzag* realizada para el control manual. 80
1.73 Objetivos de Desarrollo Sostenible (ONU). 82

1 Objetivos de Desarrollo Sostenible (ONU).[1] 111

Índice de tablas

1.1 Diferencia entre los vehículos *Segway* y *Hoverboard*. 11
1.2 Parámetros que intervienen en el modelo matemático del robot. 17
1.3 Normativa aplicable al proyecto. 26
1.4 Ventajas y desventajas del uso del metacrilato. 27
1.5 Ventajas y desventajas del control PID. 28

4.1 Cuadro de materiales. 97
4.2 Cuadro de maquinaria. 98
4.3 Cuadro de mano de obra. 98
4.4 Partida 1 de precios descompuestos. 99
4.5 Partida 2 de precios descompuestos. 99
4.6 Partida 3 de precios descompuestos. 100
4.7 Cuadro de mediciones. 101
4.8 Cuadro resumen general presupuesto. 102

A.1 Especificaciones técnicas del sensor MPU6050. 103
A.2 Especificaciones técnicas del motor 37Dx52L de Pololu. 103
A.3 Especificaciones del controlador L298N. 104
A.4 Especificaciones técnicas del microcontrolador Arduino Due. 104

C.1 Impacto de los ODS en el proyecto 113

Lista de acrónimos

TFG	Trabajo Final de Grado
PID	Regulador Proporcional-Integral-Derivativo
PD	Regulador Proporcional-Derivativo
PI	Regulador Proporcional-Integral
IMU	*Inertial Measurement Unit* (Unidad de Medición Inercial)
GDL	Grados de libertad
FEM	Fuerza electromotriz
CC(DC)	Corriente Continua (*Direct Current*)
UI	*User Interface* (Interfaz de Usuario)
ODS	Objetivos de Desarrollo Sostenible

Capítulo 1

Memoria

1.1. Introduccion

El presente Trabajo de Final de Grado, en el ámbito de la Ingeniería Electrónica Industrial y Automática, representa el resultado de integrar y aplicar los conocimientos adquiridos durante la formación universitaria. A continuación, se presenta el objetivo y las fases del proyecto, así como la estructura de la memoria.

1.1.1. Objeto

El objetivo de este proyecto es el diseño y la implementación de un prototipo de vehículo autobalanceado tipo *Segway* basado en el sistema de control del péndulo invertido. El robot utilizará sensores, tales como giroscopios y acelerómetros para mantener la estabilidad vertical mediante la acción coordinada de dos motores situados en las ruedas.

1.1.2. Fases del proyecto

Para poder modelar y simular el comportamiento dinámico del vehículo se utilizará *Matlab* (ver Figura 1.1(a))[1] , y concretamente herramientas específicas del programa como *Simescape Multibody* y *Simulink*. Para ello, será necesario crear un modelo detallado del vehículo en un *software* de diseño CAD 3D (*Siemens NX*) que incorporará características físicas y dinámicas de los componentes. Para comprobar el funcionamiento de las diferentes simulaciones se realizarán 3 pruebas:

1. **Control manual del vehículo**: Se controlará el sistema mediante un *joystick* que permitirá al usuario realizar trayectorias manuales. En esta primera fase, no se realizará un control de la velocidad lineal y angular, por lo tanto, la velocidad será constante en todo momento.

2. **Control automático de la velocidad y giro**: Se podrá controlar el sistema tanto con un *joystick* como por una secuencia definida de valores de velocidad lineal y angular. La diferencia respecto del manejo en modo manual radica en que la velocidad ya no será constante y aparecerán reguladores PD y PI para poder realizar el control automático de las velocidades.

3. **Control de trayectorias**: En esta última fase, haciendo uso de las curvas de Lissajous, el robot deberá seguir una secuencia marcada en el suelo, con un control muy preciso de la velocidad y el giro. Al modificar ciertos parámetros en las ecuaciones senoidales que definen las curvas de la trayectoria, el robot podrá adaptarse para ejecutar una amplia variedad de secuencias, sin que su control se vea afectado.

A lo largo de las simulaciones, se recogerán algunos datos y/o parámetros sobre la estabilidad del vehículo, la precisión de las trayectorias y la respuesta del sistema ante diferentes condiciones de control. Estos parámetros se irán analizando en cada una se las fases para ajustar ganancias de los controladores de la forma más adecuada y así poder optimizar al máximo el rendimiento del sistema.

Finalmente, se realizará la **implementación real** del robot empleando el microcontrolador Arduino para realizar un control automático de la estabilidad y un control manual de la trayectoria. El prototipo físico incluirá algunos de los componentes que se

[1]Fuente: `https://es.wikipedia.org/wiki/Archivo:Matlab_Logo.png`

seleccionarán posteriormente y se programará el microcontrolador mediante la plataforma de libre acceso Arduino IDE (ver Figura 1.1(b))[2] para poder realizar una correcta estabilización del prototipo en tiempo real.

Durante esta fase, se prestará especial atención a las conexiones y a la integración del *hardware* y el *software*. Por otro lado, también se realizarán ajustes detallados en los algoritmos de control basados en las observaciones y en las simulaciones. Estos ajustes incluirán una calibración inicial de los sensores y motores, además de la optimización y depuración del código fuente, para adaptar al prototipo a diferentes escenarios de prueba.

(a) Logo Matlab (b) Logo Arduino

Figura 1.1: Logos de programas utilizados.

En la Figura 1.2 se muestra un esquema inicial de las fases del proyecto, que se detallarán a lo largo del trabajo.

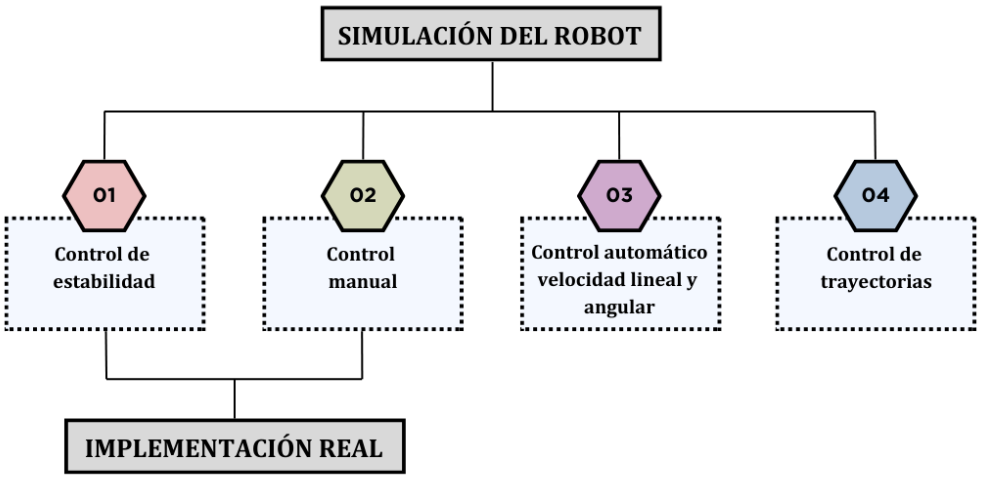


Figura 1.2: Esquema inicial de las fases del proyecto

[2]Fuente: `https://es.wikipedia.org/wiki/Archivo:Arduino_Logo.svg`

1.1.3. Estructura del documento

El presente documento estará estructurado en diversas secciones que se detallarán a continuación:

- **Sección 1: Introducción**. Preámbulo inicial que expone los objetivos principales que pretende abordar este proyecto, además de las fases que se seguirán durante la ejecución.
- **Sección 2: Antecedentes**. En esta sección se presentará al principio el marco teórico que engloba el proyecto, además de la historia de los vehículos autobalanceados, para posteriormente realizar un estudio donde se comprobará si el modelo es estable o no. Finalmente se realizará el estudio del modelo matemático del robot, incluyendo el sistema de la barra del péndulo, la rueda y el motor.
- **Sección 3: Requerimientos del proyecto y factores a considerar**. Se presentarán los objetivos y requerimientos que tendrá que cumplir el proyecto y la normativa aplicable a este.
- **Sección 4: Planteamiento de alternativas y solución adoptada**. Se presentarán alternativas al proyecto en cuanto al montaje del robot (elección de los materiales correctos), al método de control y a los componentes, especificando las razones de la elección de cada uno.
- **Sección 5: Simulación del movimiento del robot**. Se llevarán a cabo diferentes tipos de simulaciones, desde el control manual por parte del usuario, pasando por el control automático de la velocidad lineal y angular, hasta un seguimiento de trayectorias complejas.
- **Sección 6: Implementación real del robot**. Se realizará el montaje del robot y la conexión de los componentes, para posteriormente programar el microcontrolador y así lograr la estabilidad y el control manual de trayectorias.
- **Sección 7: Análisis de resultados**. Se compararán los resultados obtenidos y se realizarán pruebas para comprobar el correcto funcionamiento de las simulaciones y de la implementación real.
- **Sección 8: Conclusiones y mejoras futuras**. Explicación final de lo que ha englobado el proyecto, posibles mejoras futuras y Objetivos de Desarrollo Sostenible del trabajo.

1.2. Antecedentes

Para comprender la base y la relevancia de este proyecto, es esencial explorar los fundamentos teóricos y el desarrollo histórico dentro de las áreas que abarca este TFG. Esta sección proporciona una visión general de la evolución de los vehículos autobalanceados y su principio de funcionamiento, así como un análisis del modelo matemático en el que estará basado el robot.

1.2.1. Marco teórico del proyecto

Este proyecto se sitúa dentro del ámbito de la robótica y la mecatrónica.

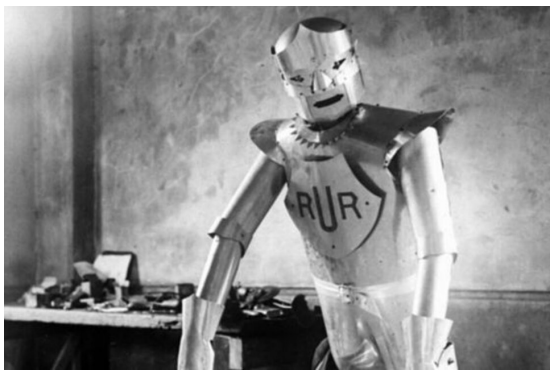
Por ende, obtener una definición universal para el término Robot puede llegar a ser compleja. Diversas organizaciones y asociaciones internacionales ofrecen definiciones diferentes, aunque próximas entre sí. La A3 Association for Advancing Automation (1979) define el término **Robot** como un "manipulador funcional reprogramable, capaz de mover material, piezas, herramientas o dispositivos especializados mediante movimientos variables programados, con el fin de realizar tareas diversas".

La robótica tiene un trasfondo interesante que se remonta al siglo XX, donde la literatura y el cine jugaron un papel crucial en su popularización. La palabra "robot" proviene del término checo *robota*, que significa "trabajo forzado" o "esclavitud". En 1920, el escritor Karel Capek publicó la novela *Rossum's Universal Robots*, una pieza de ciencia ficción que exploraba la idea de seres artificiales diseñados para realizar trabajos humanos (ver Fig. 1.3(a))[3]. Posteriormente, en 1926, en la película *Metrópolis* se presenta a un robot con aspecto antropomórfico (ver Fig. 1.3(b))[4]. Finalmente, en 1950, el famoso autor Isaac Asimov dio un gran impulso al campo de la robótica con su libro *Yo, Robot*, donde estableció las Tres Leyes de la Robótica.

La evolución de los robots en el ámbito de la industria ha sido creciente a lo largo de los últimos años. En poco más de 30 años, los desarrollos realizados sobre este campo han permitido que la robótica adquiera mucha fuerza en todas las áreas productivas.

En el desarrollo de la robótica industrial, hay 5 momentos relevantes: (Garibay Pascual, Jonathan Ruiz, 2006)

- En 1950 se desarrolla el primer manipulador.
- En 1958 se realizan los primeros proyectos y prototipos de robots, gracias a la fundación de *Unimation*.
- En 1970 Las universidades de *Standford* y el *MIT* se proponen controlar un robot mediante un computador. Se implementan los primeros microprocesadores.
- En 1980 aparece una fuerte investigación en el campo de la robótica inteligente.



(a) Robot de la novela *Rossum's Universal Robots*

(b) Robot de la película *Metrópolis*

Figura 1.3: Representación de robots en la literatura y el cine del siglo XX.

La introducción de los microprocesadores en la década de 1970 permitió avances significativos en la tecnología. La integración de la electrónica y la mecánica ha sido crucial para el desarrollo de los robots modernos. Este progreso ha sido tan notable que, para describir esta sinergia tecnológica, en 1969 un ingeniero japonés acuñó el término "**mecatrónica**", como combinación de "meca" de la palabra mecanismos y "trónica" de la palabra "electrónica". Actualmente, este término se usa para describir una filosofía en la Tecnología de la Ingeniería en la cual hay una integración coordinada y concurrente del desarrollo de la ingeniería mecánica con la electrónica y el control inteligente por computadora (ver Fig. 1.4)[5] (Bolton, 2017)

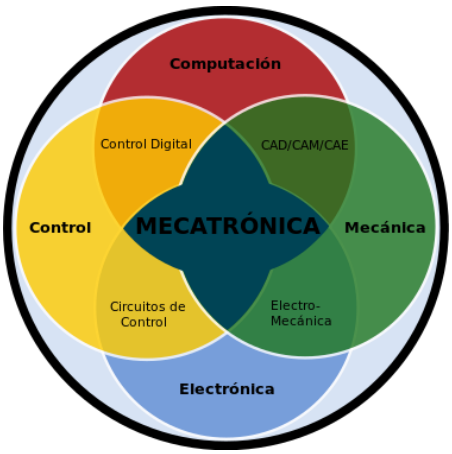
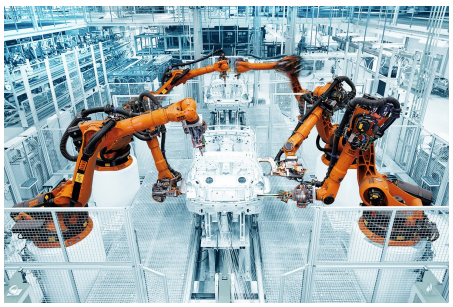


Figura 1.4: Elementos de un sistema mecatrónico.

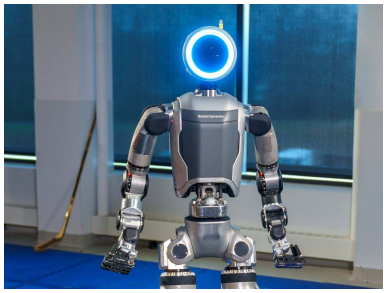
[3]Fuente: `https://engelsbergideas.com/notebook/karel-capeks-robots-took-on-a-life-of-their-own/`
[4]Fuente: `https://lineassobrearte.com/2015/03/29/metropolis-de-fritz-lang-1926/`
[5]Fuente: `https://www.profesionalreview.com/2022/12/10/mecatronica/`

Por ende, resulta complicado realizar una clasificación general de los tipos de robots que existen hoy en día debido a la diversidad y complejidad de sus aplicaciones, así como su continúa evolución. Los robots se han adaptado para trabajar y funcionar en diversos sectores y entornos de trabajo, lo que hace que cualquier intento de categorización se vea limitado por la rápida innovación y la creciente aparición de nuevas variantes y configuraciones. De forma general y basándose en su morfología, se puede realizar una clasificación general de los robots existentes en la actualidad (Garibay Pascual, Jonathan Ruiz, 2006).

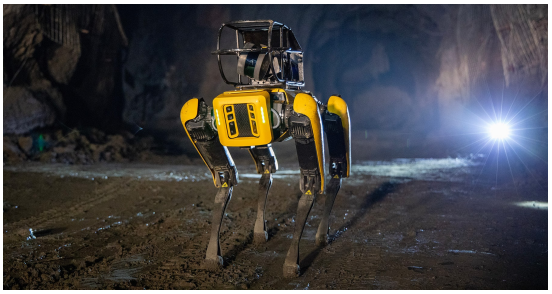


(a) Célula robotizada de Robots industriales articulados KUKA

(b) Robot para combatir el COVID-19 de Weston Robot



(c) Robot humanoide Atlas de Boston Dynamics



(d) Perro robot dinámico de Boston Dynamics

Figura 1.5: Ejemplos de tipos de robots.

- **Robot industrial o manipulador**: Son sistemas robóticos destinados a realizar tareas de manipulación física. Es decir, están destinados a realizar de forma automatizada diversos procesos de manipulación y/o fabricación de objetos. En el extremo, suelen llevar una pinza u efector para realizar estas tareas. (ver Fig. 1.5(a))[6]
- **Robots móviles**: Son robots capaces de desplazarse automáticamente debido a la acción de ruedas, patas o cualquier otro mecanismo que les permita desplazarse por el entorno. Reciben la información del exterior mediante los sensores que llevan incorporados. (ver Fig. 1.5(b))[7]

[6]Fuente: `https://eurecatacademy.org/cursos/programacion-de-robots-kuka/`
[7]Fuente: `https://es.mathworks.com/solutions/robotics/mobile-robots.html`

- **Robots androides o humanoides**: Estos robots están diseñados para imitar la apariencia y el comportamiento humano en la medida de lo posible. Tienen una forma antropomórfica, con características similares a las de los humanos, como extremidades, manos, cabeza y torso. Actualmente, es un campo con una creciente investigación debido a que estos robots, además de diseñarse para parecerse físicamente a un ser humano, están siendo desarrollados para replicar las capacidades cognitivas humanas. (ver Fig. 1.5(c))[8]
- **Robots zoomórficos**: Estos robots están inspirados en la anatomía y el comportamiento de animales. Pueden tener forma de animales reales o ficticios y están diseñados para imitar su movimiento y comportamiento. (ver Fig. 1.5(d))[9]

El campo de la robótica ha experimentado un desarrollo exponencial y diversificado en los últimos años, abarcando desde los robots robustos del siglo XX hasta los robots con aspecto antropomórfico y zoomórfico que se están llevando a cabo hoy en día. La integración de la electrónica, la mecánica y la inteligencia artificial está permitiendo el diseño de máquinas complejas capaces de interactuar con el sistema que les rodea, además de poder adaptarse a situaciones complejas. Todos estos avances, además de demostrar la importancia de este sector en la sociedad, también destacan la continua investigación y desarrollo en este sector, para enfrentar así los desafíos futuros y mejorar la calidad de vida de las personas. (Samaniego, 2023)

1.2.2. Historia y evolución de los robots autobalanceados

Un robot autobalanceado es todo aquel capaz de equilibrarse en posición vertical sobre sus dos ruedas motorizadas. Este tipo de robots utilizan sensores tales como giróscopos o acelerómetros para detectar el ángulo de inclinación y realizar las correcciones necesarias para que se cumpla la condición de estabilidad. La dos ruedas junto con sus motores, están situadas debajo de la base y permiten que el chasis del robot pueda mantenerse en posición vertical.

Dos conceptos muy importantes a tener en cuenta son el **centro de masa** y el **centro de gravedad** del robot. Las ruedas de un robot autobalanceado deben estar cerca del centro de masa para mantener bajo el centro de gravedad. Esto es importante para el control de estabilidad del robot. Sin embargo, para lograr un movimiento suave y una estabilidad dinámica, es necesario compensar el centro de masa colocando los componentes en la parte superior. Si todos los componentes están en la parte inferior, la estabilidad será fácil de mantener pero el movimiento del robot será más complejo.

[8]Fuente: `https://hipertextual.com/2024/04/atlas-robot-humanoide-ia-boston-dynamics`
[9]Fuente: `https://bostondynamics.com/products/spot/`

El vehículo utiliza un sistema de tracción diferencial, que se basa en dos ruedas motrices independientes. Esto, posibilita determinados movimientos: (ver Fig. 1.6)[10]

- Mantener una trayectoria rectilínea cuando la velocidad y dirección de ambas ruedas son iguales.
- Realizar giros sobre sí mismo cuando las ruedas giran en direcciones opuestas.
- Tomar curvas ajustando la velocidad de cada rueda en función del tipo de giro deseado.
- Adaptarse a diferentes terrenos y situaciones debido a que los motores de las ruedas son independientes entre sí.

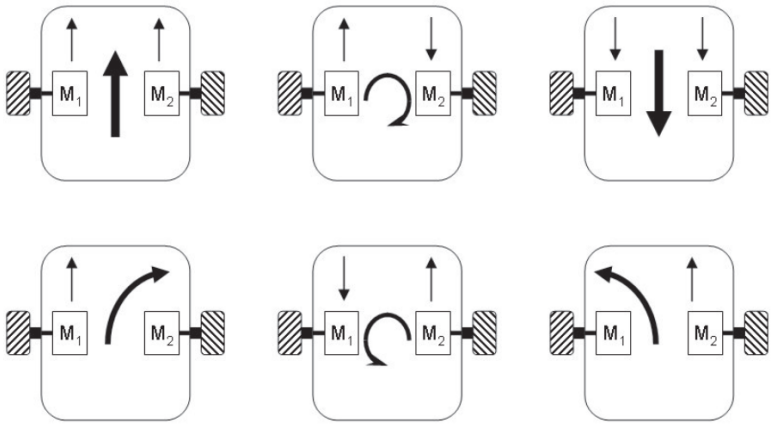


Figura 1.6: Giros y direcciones según velocidad ruedas en plataforma diferencial.

En la actualidad, existen diversas aplicaciones de este tipo de robots:

1. **Robots simbióticos humanos: EMIEW**. (ver Fig. 1.7)[11]. Son modelos de robots desarrollados por la marca Hitachi durante el programa "*Project for the Practical Application of Next-Generation Robots—Prototype Development Support*".

   Estos robots fueron diseñados para compartir espacio de forma segura con personas y ofrecer diversos servicios. Su objetivo principal era moverse lo más rápido posible por entornos irregulares, a una velocidad máxima de 6 km/h. Los principales desafíos a superar incluían garantizar la estabilidad constante del robot y permitir una rápida respuesta a los cambios en el entorno. Mediante el análisis exhaustivo de los movimientos de diversos animales, se desarrolló un mecanismo de tracción de sus ruedas. Además, se implementaron tecnologías avanzadas como el reconocimiento por voz y la navegación autónoma.

[10]Fuente: `https://images.app.goo.gl/rJ9K8xJpLWCTYyzd8`
[11]Fuente: `https://www.latercera.com/noticia/emiew-3-robot-capaz-interactuar-las-personas/`

Figura 1.7: Robot EMIEW 3 desarrollado por *Hitachi*.

2. **Robots para el transporte**.

   - **Segway**: (ver Fig. 1.8(a))[12] Es una de las aplicaciones más exitosas dentro de este tipo de vehículos, creado por Dean Kamen en el año 2001 bajo SEGWAY®. Es un vehículo de transporte unipersonal de dos ruedas basado en una combinación de sensores que detectan el grado de inclinación del usuario y en función de esto, regulan la velocidad de las ruedas para poder mantener el equilibrio. Su diseño incluye un manillar con el que se puede controlar la dirección y la velocidad.

   - **Hoverboard**: (ver Fig. 1.8(b))[13] Otro caso de éxito de este tipo de vehículos, el cual posee una batería recargable portátil y consta de dos ruedas unidas por dos plataformas, que cuentan con un mecanismo de equilibrio interno. Para ello, está dotado de sensores y un sistema giroscópico integrado en estas plataformas. Este sistema, permite al usuario controlar el movimiento del *Hoverboard* mediante la inclinación del cuerpo, lo que proporciona un desplazamiento ágil y dinámico, aunque algo inestable debido a la ausencia de manillar.

[12]Fuente: `https://deporteurbano.es/producto/ninebot-segway-ninebot-s-350-wh-black/`
[13]Fuente: `https://commons.wikimedia.org/wiki/File:Hoverboard_1.jpg`

(a) *Segway Ninebot Self Balancing Transporter*

(b) *Hoverboard Scooter* LBS15

Figura 1.8: Vehículos autoequilibrados para el transporte.

Característica	**Segway**	**Hoverboard**
Diseño	Está dotado de un manillar y una plataforma para los pies	Es una plataforma plana que no está dotada de manillar
Control	Manillar para controlar la dirección y la velocidad	El equilibrio se controla mediante el grado de inclinación que adquiere el cuerpo
Estabilidad	Tiene una mayor estabilidad ya que está dotado de manillar y mejor diseño	Es menos estable debido a la ausencia de manillar
Tamaño	Grande y pesado	Compacto y ligero
Uso	Generalmente para recorridos largos	Desplazamientos cortos y entretenimiento

Tabla 1.1: Diferencia entre los vehículos *Segway* y *Hoverboard*.

3. **Robots con fines didácticos y entretenimiento: Balanbot**. (ver Fig. 1.9)[14]. Es un kit de robot autoequilibrado basado en la plataforma de libre acceso Arduino. Este vehículo utiliza fuertes estructuras acrílicas y mantiene el equilibrio atendiendo a su centro de gravedad. Fue creado por un grupo de ingenieros (Steve Chang, Bruce Chen y Ryan Quin) los cuales demostraron que, utilizando una IMU MP6050, que combina acelerómetro y giróscopo, un controlador de motores L298P y una placa de desarrollo Arduino junto a dos motores, se podía desarrollar un robot que mantiene la estabilidad vertical y en el que se pueden desarrollar múltiples aplicaciones con fines educativos. Una ventaja de este robot, es, que debido a su estructura modular, permite a los usuarios intercambiar los componentes por otros distintos para poder experimentar con otros modelos.

Figura 1.9: Balanbot Arduino *Self Balancing Robot.*

1.2.3. Principio de funcionamiento de un vehículo autobalanceado

El principio de funcionamiento de un robot autobalanceado está basado en el mecanismo del **péndulo invertido**, debido a la necesidad de mantener el equilibrio cuando está en movimiento. Este mecanismo es uno de los sistemas no-lineales más utilizados hoy en día para la investigación de diferentes aspectos en los sistemas de control. Posee una dinámica no lineal que permite equilibrar un objeto en una posición inestable, y mediante diferentes reguladores y estrategias de control, mantener ese objeto en equilibrio vertical. Esta analogía se aplica a los vehículos autobalanceados debido a: (Bohórquez, 2003)

- **Equilibrio dinámico y corrección del ángulo de inclinación**: El robot, al igual que el péndulo debe ajustar debidamente su posición vertical con el objetivo de no caer incluso con perturbaciones, En todo momento, se mide la posición angular con respecto a la vertical y se ajusta continuamente para eliminar el error de posición al máximo.

[14]Fuente: `https://dwmzone.com/en/robot-kits/197-balanbot-self-balancing-robot-kit.html`

- **Control**: El sistema de control en ambos casos es muy parecido ya que se busca ejercer una acción de control sobre los motores para corregir cualquier desviación del equilibrio vertical. En el caso del péndulo invertido, la acción de control se aplica en la base móvil para contrarrestar esa inclinación, al igual que en un robot autobalanceado.
- **Estabilidad en movimiento**: En ambos casos, se juega con la aceleración angular de las ruedas para acelerar o desacelerar en función del grado de inclinación que adquiera el robot, asegurando siempre que el centro de gravedad se sitúe por encima de las ruedas.

Para entender el principio de funcionamiento del péndulo invertido, es esencial conocer el comportamiento dinámico del **péndulo simple**. Para ello, en la (Fig. 1.10)[15] se puede apreciar una representación del diagrama del cuerpo libre, es decir, de las fuerzas que actúan sobre esta configuración. (Vitaliti, 2023)

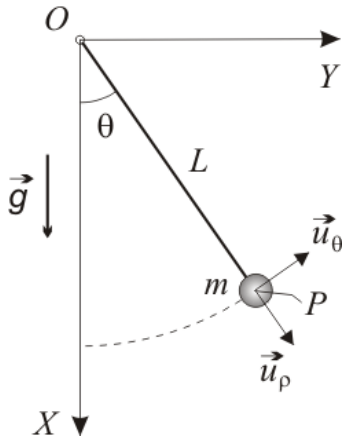


Figura 1.10: Diagrama del cuerpo libre del péndulo simple.

Para analizar esta situación genérica y obtener su ecuación del movimiento se tomará L con masa despreciable y se tendrá el cuenta que m es una masa aislada o puntual en el extremo libre.

En esta configuración, la suma de los momentos respecto al punto O de las fuerzas exteriores será igual al momento de inercia equivalente del péndulo multiplicado por su aceleración angular.

$$\sum_{\text{ext}}^{0} M = J \cdot \ddot{\theta} \tag{1.1}$$

[15] Fuente: `http://tesla.us.es/wiki/index.php/`

Partiendo de la Ecuación 1.1 y considerando que el momento efectivo causado por la fuerza peso es el producto de esta fuerza por la distancia mínima al eje de rotación, obtenemos la Ecuación del movimiento del péndulo simple (1.2)(1.3). En esta ecuación, el ángulo θ aumenta en sentido antihorario debido a que los momentos se producen en sentido horario. Además, el segundo término de (1.2) está determinado por el momento de inercia equivalente del péndulo respecto al centro de momentos. Este momento de inercia equivalente se calcula mediante el teorema de Steiner, considerando el momento de inercia de la masa puntual referido por la distancia.

$$-m \cdot g \cdot l \cdot sin(\theta) = m \cdot l^2 \cdot \ddot{\theta} \tag{1.2}$$

Simplificando se obtiene la ecuación del movimiento del péndulo simple en forma de movimiento oscilatorio armónico o movimiento armónico simple:

$$\ddot{\theta} + \frac{g}{l}\theta = 0 \tag{1.3}$$

Entonces, la solución de (1.3) vendrá dada como ecuación armónica:

$$\theta(t) = A \cdot \cos\left(\sqrt{\frac{g}{l}} \cdot t\right) + B \cdot \sin\left(\sqrt{\frac{g}{l}} \cdot t\right) \quad \forall t \geq 0 \tag{1.4}$$

Se observa que la influencia de la gravedad g hace que el sistema trate de mantener siempre el equilibrio. Es un **sistema estable** sin amortiguamiento.

Con respecto al péndulo simple, el **péndulo invertido** (ver Fig. 1.11)[16] posee el mismo momento de inercia con respecto al eje de rotación, aunque el momento generado por la fuerza peso en este caso es positivo (la influencia del campo gravitatorio tiende a desestabilizar el péndulo)

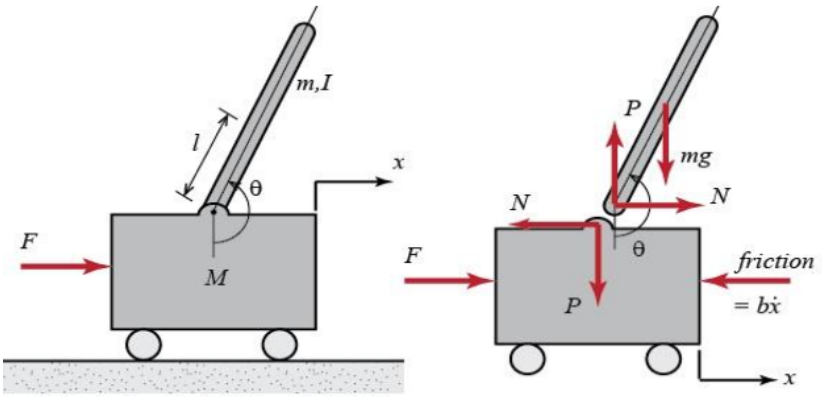


Figura 1.11: Diagrama del cuerpo libre del péndulo invertido.

[16]Fuente: `https://medium.com/@controleavancadoemultivariavel`

En analogía al modelo del péndulo simple:

$$\sum_{\text{ext}}^{0} M = J \cdot \ddot{\theta} \tag{1.5}$$

$$m \cdot g \cdot l \cdot sin(\theta) = m \cdot l^2 \cdot \ddot{\theta} \tag{1.6}$$

Simplificando (1.6):

$$\ddot{\theta} - \frac{g}{l}\theta = 0 \tag{1.7}$$

En este caso, la solución es un movimiento armónico inestable, ya que en vez de funciones trigonométricas, tiene como soluciones funciones hiperbólicas. Es un sistema inestable ya que una vez apartado de su posición de equilibrio, no es capaz de volver a este por sí mismo.

$$\theta(t) = C \cdot \cosh\left(\sqrt{\frac{g}{l}} \cdot t\right) + D \cdot \sinh\left(\sqrt{\frac{g}{l}} \cdot t\right) \quad \forall t \geq 0 \tag{1.8}$$

Para estabilizar el sistema, se podría suponer que hay una fuerza perturbadora $\vec{F}$, que sería la encargada de romper el equilibrio cuando el sistema se encuentra en condiciones iniciales nulas, es decir, $\theta = 0$. Otra suposición que se puede realizar para completar la ecuación diferencial del movimiento del péndulo invertido es que el eje de rotación tiene un rozamiento que generará un momento opuesto proporcional a la velocidad angular.

Entonces, la suma de momentos de las $\vec{F}_{\text{ext}}$ bajo esta nueva condición sería:

$$F(t) \cdot l \cdot cos(\theta) + m \cdot g \cdot l \cdot sin(\theta) - B \cdot \dot{\theta} = m \cdot l^2 \cdot \ddot{\theta} \tag{1.9}$$

Donde se suman el momento que ejerce la fuerza en función de la distancia, el momento ejercido po el peso (el que había anteriormente) y el momento de fricción (el momento es proporcional a la primera potencia de la velocidad angular); igualando todo lo anterior al momento de inercia equivalente multiplicado por la aceleración angular, siendo B la constante de proporcionalidad o de rozamiento viscoso.

En el caso de estar ante oscilaciones muy pequeñas, se puede suponer que $\cos(\theta) \approx 0$ y $\sin(\theta) \approx 0$, de lo que se obtiene la ecuación completa del péndulo invertido:

$$F(t) \cdot l = m \cdot l^2 \cdot \ddot{\theta} + B \cdot \dot{\theta} - m \cdot g \cdot l \cdot \theta \tag{1.10}$$

Se plantea el diagrama de bloques (ver Fig. 1.12) con previa transformada de Laplace:

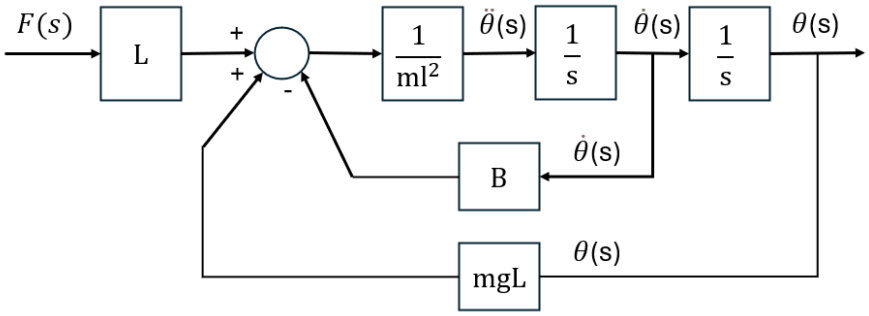


Figura 1.12: Diagrama de bloques para el control del péndulo invertido.

Donde la entrada sería una fuerza perturbadora $F(s)$ y la salida el ángulo de estabilidad $\theta(s)$. El momento ejercido por el peso es una realimentación positiva, por lo que es un **sistema inestable** (es decir, es un sistema que no será capaz de restablecer el equilibrio por sí mismo), ya que estas realimentaciones tienden a desestabilizar el sistema. Por lo tanto, la función de transferencia del péndulo invertido sin compensar sería:

$$\frac{\theta(s)}{F(s)} = \frac{l}{ml^2s^2 + Bs - mgl} \tag{1.11}$$

Cuando el péndulo está cerca de la posición vertical, su comportamiento puede aproximarse a un sistema lineal. Sin embargo, a medida que el péndulo se desvía más de esta posición, la relación entre las entradas y las salidas del sistema se vuelve **no lineal**. Por ejemplo, la fuerza necesaria para estabilizar el péndulo aumenta de manera no lineal a medida que el ángulo de inclinación aumenta. Además, las oscilaciones del péndulo introducen comportamientos no lineales. Por lo tanto, las ecuaciones que describen el modelo también lo serán, y requerirán métodos adecuados para su análisis y control.

1.2.4. Modelo matemático del robot

A continuación, se realizará un análisis del modelo matemático del robot en cuestión. Este análisis, estará dividido en tres partes: la primera para el péndulo invertido, la segunda para las ruedas y la tercera para el sistema de los motores eléctricos. Tanto el modelo del péndulo como el de la rueda, tienen 3 ecuaciones cada uno, dos para las direcciones de los ejes x e y y la otra, para la rotación. En la tabla 1.2 se muestran los parámetros que se han utilizado en el modelo matemático de los tres componentes principales del robot (Sundin, 2012)

Parámetros	**Significado**	**Unidades**
Péndulo		
g	Aceleración de la gravedad	m/s^2
I_p	Momento de inercia del péndulo	$kg \cdot m^2$
m_p	Masa del péndulo	kg
L_p	Distancia entre los centros de masa de la rueda y el péndulo	m
ψ_p	Ángulo del péndulo respecto a la vertical	rad
$\dot{\psi}_p$	Velocidad angular del péndulo	rad/s
x_p	Posición en el eje x del péndulo	m
y_p	Posición en el eje y del péndulo	m
N_x	Fuerza entre el péndulo y la rueda en la dirección del eje x	N
N_y	Fuerza entre el péndulo y la rueda en la dirección del eje y	N
Rueda		
I_r	Momento de inercia de la rueda	$kg \cdot m^2$
m_r	Masa de la rueda	kg
r	Radio de las ruedas	m
θ_r	Ángulo de la rueda respecto a la vertical	rad
$\dot{\theta}_r$	Velocidad angular de la rueda	rad/s
x_r	Posición en el eje x de la rueda	m
y_r	Posición en el eje y de la rueda	m
N_r	Fuerza normal ejercida por el suelo hacia la rueda	N
F_r	Fuerza de rozamiento entre el suelo y la rueda	N
Motor DC		
R_a	Valor de la resistencia nominal	Ω
k_t	Constante de par del motor DC	Nm/A
k_f	Constante de la FEM inversa del motor DC	$Vsec/rad$
T_m	Par de los motores	Nm
U_m	Voltaje de entrada a los motores	V
n_m	Relación de engranajes	-

Tabla 1.2: Parámetros que intervienen en el modelo matemático del robot.

Modelo del péndulo

En la (Figura 1.13)[17] se muestra el diagrama del cuerpo libre de la barra del péndulo invertido:

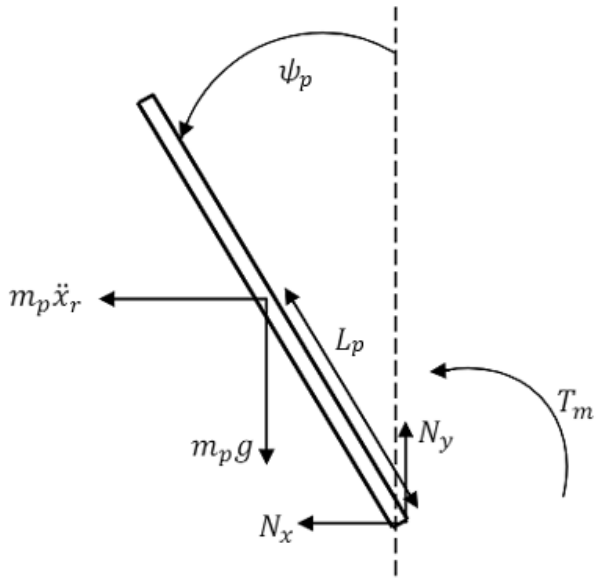


Figura 1.13: Diagrama del cuerpo libre de la barra del péndulo invertido.

A continuación se muestran las tres ecuaciones que modelan el comportamiento de las fuerzas del péndulo siguiendo la **Segunda Ley de Newton**, también conocida como el principio fundamental de la dinámica. La Ecuación (1.12) es la ecuación del momento generado por el péndulo en sentido antihorario alrededor de su centro de masa, considerando el momento de torsión aplicado, la fuerza gravitacional y la aceleración angular del péndulo. Por otro lado, la Ecuación (1.13) representa el sumatorio de las fuerzas que actúan en el eje x y la Ecuación (1.14) representa el sumatorio de las fuerzas que actúan en el eje y, mostrando cómo las fuerzas horizontales y verticales se equilibran para mantener la estabilidad del péndulo en ambas direcciones.

$$\sum \text{ Momentos } = I_p\ddot{\psi} \tag{1.12}$$

$$\sum F_{xpend} = 0 \tag{1.13}$$

$$\sum F_{ypend} = 0 \tag{1.14}$$

[17] Fuente: `https://odr.chalmers.se/server/api/core/bitstreams/3e7cc529-0f70-4785-b99d-13db701904fb/content`

Desarrollando las 3 ecuaciones anteriores respectivamente, se obtiene (Sundin, 2012):

$$T_m + m_p g L_p \sin\psi_p + m_p \ddot{x}_r L_p \cos\psi_p = I_p \ddot{\psi}_p \tag{1.15}$$

$$-N_x - m_p \ddot{x}_r = m_p \ddot{x}_p \tag{1.16}$$

$$N_y - m_p g = m_p \ddot{y}_p \tag{1.17}$$

La aceleraciones $\ddot{x}_p$ y $\ddot{y}_p$ no son lineales, por lo tanto, deben transformarse del sistema de coordenadas xy a un sistema de coordenadas rotacional, para poder trabajar con ellas. Para ello, se han expresado x_p y y_p de la siguiente forma:

$$x_p = -L_p \sin\psi_p \tag{1.18}$$

$$y_p = L_p \cos\psi_p \tag{1.19}$$

Para obtener la expresión de las velocidades en el sistema de referencia rotacional, se derivan (1.18) y (1.19):

$$\dot{x}_p = -L_p \dot{\psi}_p \cos\psi_p \tag{1.20}$$

$$\dot{y}_p = -L_p \dot{\psi}_p \sin\psi_p \tag{1.21}$$

Para obtener la expresión de las aceleraciones en el sistema de referencia rotacional, se derivan (1.20) y (1.21):

$$\ddot{x}_p = -L_p \ddot{\psi}_p \cos\psi_p + L_p \dot{\psi}_p^{\,2} \sin\psi_p \tag{1.22}$$

$$\ddot{y}_p = -L_p \ddot{\psi}_p \sin\psi_p - L_p \dot{\psi}_p^{\,2} \cos\psi_p \tag{1.23}$$

Para encontrar el momento de inercia del péndulo, según el teorema de Steiner, el cuerpo se considera como un prisma de masa distribuida uniformemente:

$$I_p = \frac{1}{12} m_p (W_p)^2 + \frac{1}{3} m_p (H_p)^2 \tag{1.24}$$

donde m_p es la masa del péndulo, W_p es el ancho y H_p es la altura del sistema analizado.

Modelo de la rueda

En la (Figura 1.14)[18] se muestra el diagrama del cuerpo libre de la rueda:

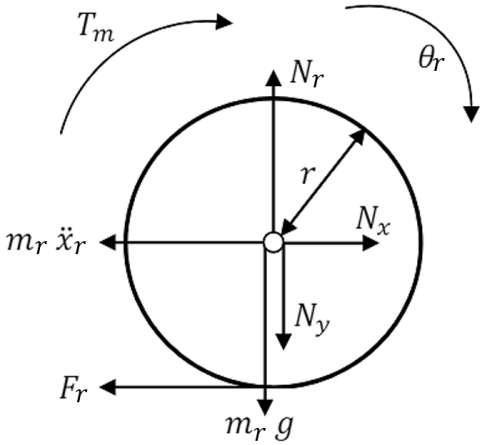


Figura 1.14: Diagrama del cuerpo libre de la rueda del péndulo invertido.

Al igual que en el modelo dinámico del péndulo, se muestran las tres ecuaciones que modelan el comportamiento de las fuerzas de la rueda del péndulo siguiendo la Segunda Ley de Newton. La ecuación (1.25) es la ecuación de momento alrededor del centro de masa de la rueda. La ecuación (1.26) representa las fuerzas actuando en la dirección del eje$x$ y la ecuación (1.27) las fuerzas actuando en la dirección del eje y, mostrando como las fuerzas horizontales y verticales se equilibran para mantener la estabilidad de la rueda en ambas direcciones.

$$\sum \text{Momentos } = I_r\ddot{\theta} \tag{1.25}$$

$$\sum F_{xrueda} = 0 \tag{1.26}$$

$$\sum F_{yrueda} = 0 \tag{1.27}$$

Desarrollando las 3 ecuaciones anteriores respectivamente, se obtiene (Sundin, 2012):

$$T_m + F_r = I_r\ddot{\theta} \tag{1.28}$$

$$N_x - m_r\ddot{x_r} - F_r = 0 \tag{1.29}$$

$$N_r - N_y - m_r g = m_r\ddot{y_r} \tag{1.30}$$

[18]Fuente: `https://odr.chalmers.se/server/api/core/bitstreams/3e7cc529-0f70-4785-b99d-13db701904fb/content`

La transformación del sistema de coordenadas xy al sistema de coordenadas rotacional de las aceleraciones $\ddot{x}_p$ y $\ddot{y}_p$ sería:

$$\ddot{x_r} = r\ddot{\theta_r} \tag{1.31}$$

$$\ddot{y}_r = 0 \tag{1.32}$$

Y de la misma forma que con el péndulo, utilizando el Teorema de Steiner para calcular el momento de inercia de un círculo, se obtiene que la inercia será:

$$I_r = \frac{2r\sin^2\alpha}{3\alpha} \tag{1.33}$$

Donde r es el radio de la rueda y α es el ángulo que para un círculo completo sería 2π radianes.

Modelo del motor DC

La última parte para completar el modelo matemático del robot sería el análisis de los motores de corriente continua. Para ello, se necesitará obtener relaciones entre el voltaje de entrada al motor, la señal de control y el par suministrado por los motores. En la (Figura 1.15)[19] se muestra el circuito equivalente de un motor DC. Aplicando las leyes de Kirchoff y realizando un análisis de la malla, se obtiene la Ecuación (1.34):

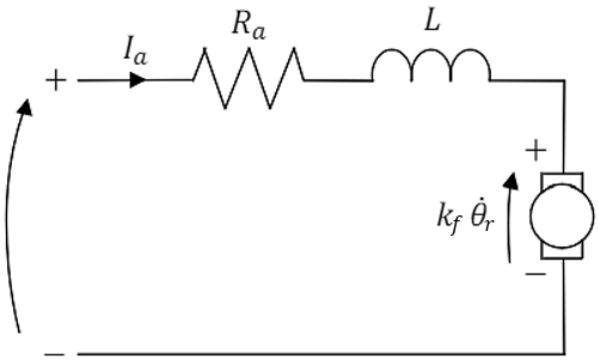


Figura 1.15: Circuito eléctrico equivalente de un motor DC.

$$U_m = R_a I_a + k_f\dot{\theta_r} + L_m\frac{di}{dt} \tag{1.34}$$

[19]Fuente: `https://odr.chalmers.se/server/api/core/bitstreams/3e7cc529-0f70-4785-b99d-13db701904fb/content`

Al ser la constante de tiempo eléctrica mucho menor que la inductancia, esta puede despreciarse, lo que da como resultado la Ecuación (1.35):

$$U_m = R_a I_a + k_f \dot{\theta_r} \tag{1.35}$$

El par del eje debe superar la inercia del motor, así como el amortiguamiento viscoso y la fricción, que son difíciles de estimar y generalmente muy pequeños. Por lo tanto, se desprecian todos ellos, debido a su insignificancia. De esta manera, el par del eje se puede describir como:

$$T_m = n_m k_t I_a \tag{1.36}$$

Posteriormente, sabiendo que T_r corresponde al par nominal del motor y $I_{a,r}$ es la corriente nominal inducida por el motor, estimamos los valores de la constante del par k_t y de la constante de la fuerza contraelectromotriz del motor k_f. Los dos primeros valores se pueden obtener fácilmente del *datasheet* o ficha técnica de los motores que se vayan a utilizar.

$$k_t = \frac{T_r}{I_{a,r}} \tag{1.37}$$

$$k_e = \frac{(V_r - I_{a,r} R_a) \cdot 60}{2\pi n_r} \tag{1.38}$$

Finalmente, el par del motor queda definido como:

$$T_m = \frac{n_m k_t U_m}{R_a} - \frac{n_m k_f k_t \dot{\theta_r}}{R_a} \tag{1.39}$$

Modelo no lineal

Una vez obtenidos los modelos matemáticos de los tres componentes principales del robot, se realiza el modelo no lineal para obtener así la aceleración angular del péndulo $\ddot{\psi}$ y la aceleración angular de las ruedas $\ddot{\theta}$.

$$\ddot{\psi_p} = \frac{I_r(gL_pR_a \sin\psi_p + 2n_m k_t(U_m - k_f\dot{\theta_r})) + r(L_p r R_a m_p \sin\psi_p(m_p(g - L_p\dot{\psi_p}^2\cos\psi_p) + gm_r)}{R_a(I_p(I_r + r^2(m_p + m_r)) - (L_p)^2 r^2 (m_p)^2 \cos^2\psi_p)} + \frac{2n_m k_t(U_m - k_f\dot{\theta_r})(m_p(L_p\cos\psi_p + r) + rm_r))}{R_a(I_p(I_r + r^2(m_p + m_r)) - (L_p)^2 r^2 (m_p)^2 \cos^2\psi_p)} \tag{1.40}$$

$$\ddot{\theta_r} = \frac{L_p r m_p \cos\psi_p(gL_pR_a \sin\psi_p + 2n_m k_t(U_m - \dot{\theta_r})) + I_p(2n_m k_t(U_m - \dot{\theta_r}) - L_p r R_a m_p \dot{\psi_p}^2 \sin\psi - p)}{R_a(I_p(I_r + r^2(m_p + m_r)) - (L_p)^2 r^2 (m_p)^2 \cos^2\psi_p)} \tag{1.41}$$

Modelo linealizado

Para poder linealizar correctamente el modelo analizado, se definen primero las variables de estado:

$$x = \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.42}$$

Esto proporciona el modelo de espacio de estados linealizado y las ecuaciones que van a definir el modelo completo del robot:

$$\dot{x} = Ax + Bu \tag{1.43}$$

$$\dot{y} = Cx + Du \tag{1.44}$$

donde:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gL_p^2 r m_p^2}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} & \frac{-2n_m I_p k_f k_t - 2L_p n_m r k_f k_t m_p}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} & 0 \\ 0 & \frac{gL_p I_r m_p R_a + L_p r^2 m_p(g m_p + g m_r) Ra}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} & \frac{-2n_m I_r k_f k_t - 2n_m r k_f k_t((L_p+r)m_p + r m_r)}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{2n_m I_p k_t + 2L_p n_m r k_t m_p}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} \\ \frac{2n_m I_r k_t + 2n_m r k_t((L_p+r)m_p + r m_r)}{(-L_p^2 r^2 m_p^2 + I_p(I_r + r^2(m_p+m_r)))R_a} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1.45}$$

En este modelo, se pueden observar todas las variables que intervienen en el modelado dinámico de un robot autobalanceado, de acuerdo con la Tabla 1.2.

1.3. Requerimientos del proyecto y factores a considerar

1.3.1. Especificaciones del proyecto

En este proyecto, inspirado en vehículos como el *Segway*, se propone crear un vehículo autobalanceado que sea capaz de mantener su estabilidad vertical de forma autónoma, combinando sistemas mecánicos con ingeniería de control. Para llevarlo a cabo correctamente, es necesario plantear previamente una serie de requisitos y especificaciones que el modelo final deberá cumplir, todo ello detallado para guiar de una forma más sencilla el proceso de diseño, implementación y validación del vehículo simulado y real.

Requerimientos de la simulación

- El diseño 3D del vehículo se realizará utilizando el programa *Siemens NX*, incorporando, de forma aproximada, características físicas y dinámicas de los componentes, para facilitar su posterior fabricación y ensamblaje.

- Se implementará un diseño modular para permitir futuras modificaciones del proyecto y para facilitar la exportación de las piezas a la plataforma de simulación del vehículo.

- La simulación del modelo del vehículo se realizará mediante el uso del programa *Matlab*, con herramientas específicas como *Simescape Multibody* y *Simulink*.

- El sistema de control del vehículo, deberá ser capaz de mantener la estabilidad vertical mediante la acción conjunta de diversos sensores.

- Se utilizarán controladores con ganancias ajustables para lograr una respuesta dinámica óptima y adaptabilidad a diferentes condiciones de funcionamiento.

- El control manual y automático del prototipo deberán ser precisos y sensibles a las entradas del usuario para garantizar un desplazamiento suave y seguro.

- Se implementará un algoritmo de control de trayectorias que permita modificar la secuencia sin realizar cambios significativos durante el proceso.

- Los valores de las ganancias de los controladores PID se mantendrán iguales durante todas las etapas de la simulación, aunque se añadan reguladores adicionales para realizar tareas complementarias a las anteriores.

- Se llevarán a cabo pruebas y se obtendrán gráficas para validar el funcionamiento del prototipo en diversas condiciones, incluyendo control manual, control automático de velocidad y giro, y seguimiento de trayectorias complejas.

Requerimientos de la implementación

- El principio de control de la implementación será similar al de la simulación: el vehículo deberá ser capaz de mantener la estabilidad mediante la acción coordinada de sensores integrados en la IMU, junto con un controlador ajustable para optimizar la respuesta dinámica.

- Se utilizará un microcontrolador con puertos de entrada y salida suficientes para realizar las conexiones necesarias.

- Se empleará una IMU de alta precisión para recopilar datos en tiempo real y detectar desviaciones en el ángulo de inclinación del vehículo.

- Se requerirán motores de corriente continua con una reductora lo suficientemente alta y con capacidad de respuesta y momento suficientes para proporcionar los impulsos necesarios en las ruedas.

- Las pruebas de validación del robot se realizarán en entornos controlados.

- Se diseñará una aplicación con una interfaz (UI) sencilla para que cualquier usuario pueda realizar con facilidad un control manual del robot.

Limitaciones del Proyecto

1. Limitaciones del diseño:

    - El diseño 3D del vehículo puede tener limitaciones en la precisión y detalle debido a la aproximación de características físicas y dinámicas.

2. Limitaciones de la simulación:

    - La precisión del controlador y la respuesta del vehículo en la simulación pueden diferir de la realidad debido a la simplificación de los modelos.

3. Limitaciones de la implementación:

    - La implementación puede tener limitaciones en la precisión y la ejecución respecto a la simulación del vehículo.
    - La capacidad de respuesta y precisión de los sensores integrados en la IMU puede limitar la efectividad del sistema de control para mantener la estabilidad.
    - La elección de motores de corriente continua con una reductora puede limitar la velocidad máxima y la capacidad de respuesta del vehículo.

1.3.2. Normativa aplicable al proyecto

Con el objetivo de realizar un proyecto completo y garantizar la fiabilidad y seguridad del prototipo que se va a desarrollar, se considerarán diversas normativas que afectan a sistemas eléctricos y electrónicos, así como a proyectos con fines educativos y experimentales. Ordenadas por normativa general y específica del proyecto, se muestran en la Tabla 1.3 (AENOR, 2024) (Agencia Estatal Boletín Oficial del Estado, 2024)

Normativa	**Descripción**
Normativa general	
Ley 12/1986, del 1 de abril	Define las atribuciones de los Ingenieros Técnicos.
Real Decreto 1215/97, del 18 de julio	Normativa sobre la seguridad y salud en el trabajo.
Ley Orgánica 5/2002, del 19 de junio	Establece las cualificaciones profesionales.
Real Decreto 208/2005, del 25 de febrero	Gestión de residuos de componentes electrónicos
Normativa Específica	
UNE-EN IEC 60086	Estándares para baterías primarias.
UNE-EN IEC 60469	Señales de pulsos y comunicación eficiente.
UNE-EN IEC 61190	Materiales de fijación y aleaciones de soldadura electrónica.
UNE-EN ISO 898	Características de acero para pernos y tornillos.
UNE-EN ISO 1043	Nomenclatura estandarizada de plásticos.
UNE-EN ISO/IEC 80000	Guía para el uso de magnitudes físicas y medidas.
UNE-EN IEC 61960	Características de baterías de iones de litio.
UNE-EN ISO 13482	Especificaciones para diseño de robots no industriales.
ISO 8373	Definición de términos relacionados con robots.
ISO 9787	Sistemas de coordenadas para robots y movimientos básicos.
ISO 19649	Términos para robots móviles en aplicaciones industriales.
ISO DIN 13	Tamaños y métricas normalizadas para tornillos y tuercas.
Dibujo técnico y planos	
UNE-EN ISO 5455	Definición y tipos de escalas.
UNE 1-026-83 (2)	Reglas generales para la obtención de formatos de los planos.
UNE 1-035-95	Cajetines y cuadros de rotulación
UNE 1-027-95	Plegado de planos
UNE 1-032-82	Clases de líneas, grosores.
UNE 1-034-75	Escritura en los planos

Tabla 1.3: Normativa aplicable al proyecto.

1.4. Planteamiento de alternativas y solución adoptada

En el proceso del diseño y el desarrollo de este prototipo, es fundamental analizar algunas de las diferentes opciones consideradas para el diseño general del vehículo. Por ello, en esta sección se van a examinar alternativas al enfoque del diseño, así como los sistemas de control disponibles para poder identificar la solución más adecuada que cumpla con el objeto del proyecto (véase Apartado 1.1.1: Objeto del proyecto). Finalmente, se justificará la elección de los componentes electrónicos, ya sea por sus características o por su mejor adaptación a las necesidades del proyecto.

1.4.1. Alternativas de diseño

Para el diseño de las bandejas, se consideraron diferentes opciones de materiales. El aluminio puede ofrecer mayor resistencia y durabilidad, pero por otro lado, es más caro y su mecanizado es más complejo que el de otros materiales. El plástico ABS (Acrilonitrilo Butadieno Estireno), por el contrario, es bastante fácil de mecanizar y económico, aunque su resistencia estructural no es elevada y tiende a deformarse a altas temperaturas, por lo que tampoco se consideró una opción viable. La madera, al igual que el ABS, es económica y su mecanizado es simple, pero su durabilidad y resistencia a la humedad hacen que esta opción no sea la más efectiva. Además, la resistencia estructural de la madera no es elevada, ya que puede deformarse o desgastarse con el tiempo. Finalmente, para la construcción de las bandejas se optó por usar **placas de metacrilato de 3mm** de grosor. Las principales ventajas y desventajas del uso del metacrilato se muestran en la Tabla 1.4.

En cuanto al diseño mecánico de las bandejas, se ha optado por una **forma rectangular** debido a que maximiza el uso del espacio disponible y facilita la organización de los componentes. Además, permite una buena resistencia estructural del robot.

Uso del metacrilato en la mecanización de las bandejas	
Ventajas	**Desventajas**
Peso ligero, por lo que no dificulta la tarea de estabilidad	Menor resistencia estructural.
Transparencia, por lo que facilita la inspección visual de componentes internos	Puede rayarse fácilmente.
Facilidad de mecanizado	Mayor coste en comparación con otros materiales como la madera.
Estética visual y profesional	Mayor fragilidad en su manipulación en comparación con algunos plásticos.

Tabla 1.4: Ventajas y desventajas del uso del metacrilato.

1.4.2. Alternativas de control

En cuanto a los sistemas para controlar la estabilidad del robot, hoy en día existen diversas alternativas, aunque su viabilidad y rentabilidad pueden variar. Algunas opciones incluyen el control basado en redes neuronales o en aprendizaje automático, dada la creciente utilización de la inteligencia artificial en diversos campos. Sin embargo, estos enfoques pueden no ser adecuados para este tipo de robots en un ámbito educativo, donde se requiere un control más simple y accesible.

Otra alternativa sería el control basado en visión por computadora o robótica inteligente, utilizando cámaras y técnicas de procesamiento de imágenes para el control automático del robot. Sin embargo, esta opción suele implicar costes elevados y aumenta la complejidad del proyecto al requerir instalaciones previamente preparadas para este propósito.

Finalmente, se ha optado por un **control PID** utilizando sensores como giroscopios y acelerómetros integrados en una IMU. Esta elección ofrece ventajas económicas y una menor complejidad en la programación del control, por lo que será más adecuado para proyectos en el ámbito educativo.

Ventajas y desventajas del control PID	
Ventajas	**Desventajas**
Proporciona un control estable y preciso del robot en tiempo real.	Requiere una configuración inicial y conocimientos básicos en el campo del control.
Es adecuado para una amplia variedad de aplicaciones.	Puede ser sensible a vibraciones y movimientos bruscos.
Permite la compensación de errores y perturbaciones externas en tiempo real, mejorando la robustez del sistema.	No es adecuado para aplicaciones donde se requiere un control predictivo o basado en el aprendizaje.
Es relativamente barato en comparación con otros métodos de control.	

Tabla 1.5: Ventajas y desventajas del control PID.

1.4.3. Alternativas de los componentes electrónicos

En esta sección, se van a valorar los componentes que se emplearán en la implementación real del robot. Se explorarán alternativas a estos dispositivos electrónicos, justificando detalladamente la selección de los componentes finales y las características técnicas de cada uno de ellos. Posteriormente, (véase Apartado 1.6.1: Materiales utilizados y funcionamiento de los componentes) se detallará el funcionamiento de cada uno y las conexiones necesarias para realizar la implementación.

Sensor IMU

Una IMU (Unidad de Medición Inercial) es un dispositivo electrónico que incorpora diferentes tipos de sensores para medir la aceleración, orientación y velocidades angulares. Las más comunes están compuestas por 3 acelerómetros y 3 giroscopios, y en algunos casos, también por 3 magnetómetros. La presencia de 3 sensores de cada tipo permite medir en cada uno de los 3 ejes del vehículo: guiñada (*yaw*), cabeceo (*pitch*) y alabeo (*roll*). Una IMU no mide ángulos directamente, sino que utiliza acelerómetros y giroscopios para recopilar datos de aceleración y velocidades angulares. Un procesador integrado realiza cálculos con estos datos para determinar los ángulos de orientación. Todos estos componentes pueden estar integrados en un solo encapsulado, haciendo que esta sea compacta y eficiente. (SBG Systems, 2024)

En el mercado se pueden encontrar diferentes tipos de IMUs en función de su principio de funcionamiento:

- **Basados en FOG**: Giroscopio de fibra óptica.
- **RLG**: Giroscopio láser de anillo.
- **Basados en la tecnología MEMS** (*Micro Electro-Mechanical Systems*): Tienen un alto rendimiento y su potencia no es elevada, por ello permite reducir los costes al mismo tiempo que facilita el rendimiento. Suelen integrar los acelerómetros y giroscopios en el mismo chip, lo que los hace compactos y versátiles.

Por ello, dentro de las IMU basadas en la tecnología MEMS, se ha optado por el **Módulo GY-521 Acelerómetro y Giroscopio MPU-6050** (ver Figura 1.16)[20]. Este sensor ha sido seleccionado debido a su precio reducido y su capacidad para ofrecer 6 grados de libertad, proporcionando datos muy precisos, lo cual es esencial para esta aplicación. Además, cuenta con un procesador integrado, lo que permite derivar parte del procesamiento al módulo, aliviando la carga del procesador principal si fuera necesario.

Otra opción que se consideró fue el ITG3200/ADXL345, sin embargo, su precio es significativamente más alto y ofrece características similares a las del GY-521.

[20]Fuente: `https://naylampmechatronics.com/57-sensores-posicion-inerciales-gps`

Figura 1.16: Módulo GY-521 Acelerómetro y Giroscopio MPU-6050.

Algunas de las especificaciones técnicas del sensor de detallan en la Tabla A.1 del Anexo A (Naylamp Mechatronics, 2024b).

Motores

Para mantener la estabilidad del robot autobalanceado, se deberán de utilizar dos motores. Dentro de este ámbito, podemos encontrar diferentes tipos en los cuales se podría basar el proyecto:

- **Motores de Corriente continua (DC)**: Los motores de corriente continua convierten energía eléctrica en energía mecánica mediante un campo magnético generado por una señal de corriente continua. Son comunes en aplicaciones que requieren control de velocidad y dirección, como es el caso de los robots. Pueden girar en ambas direcciones al invertir la polaridad de sus terminales, y se controla la velocidad y el par mediante la corriente suministrada o PWM. Sin embargo, no permiten controlar su posición directamente. Para reducir la velocidad y aumentar el par, se utilizan cajas reductoras. (Solorobotica, 2014)

- **Motores paso a paso**: Son precisos tanto en posición como en velocidad, ya que su rotación se divide en pasos equitativos, medidos en grados. Estos motores convierten impulsos eléctricos en desplazamientos angulares discretos. Sin embargo, su uso es necesario en aplicaciones donde se requiere precisión, no siendo adecuados para aplicaciones donde lo más importante es la fuerza o el par generado, como en este caso. (stepperonline, 2024)

- **Servomotores**: Es un dispositivo que controla y mantiene su posición mediante un sistema de retroalimentación, compuesto por un motor de corriente continua, una reductora, un potenciómetro y circuitería electrónica. Utiliza pulsos para regular su posición y velocidad, gracias a un *encoder*. Sin embargo, su ángulo de giro limitado, generalmente de 180^{o}, lo hace inadecuado para aplicaciones que requieren un movimiento continuo o mayor rango de giro. Su precio, para las características necesarias, es bastante elevado en comparación con otros tipos de motores. Por estas razones, queda descartado para esta aplicación. (Aula21, 2023)

Por lo tanto, se ha optado por seleccionar motores de corriente continua. Específicamente, se eligieron dos **motorreductores 30:1 Metal Gearmotor 37Dx52L mm 12 V** de la marca **Pololu** (ver Figura 1.17)[21], debido a que son económicos y tienen las características y potencia necesarias para este tipo de aplicaciones. Además, estos motores ofrecen una ventaja notable al contar con una amplia gama de accesorios disponibles, como soportes para su montaje en el chasis del robot, una amplia variedad de ruedas que se acoplan perfectamente y la posibilidad de utilizar *encoders* para el control preciso de la posición y velocidad.

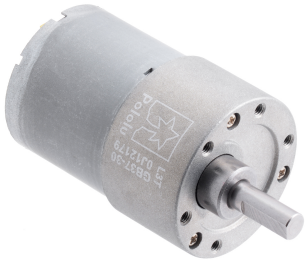


Figura 1.17: Motorreductor 30:1 *Metal Gearmotor* 37Dx52L mm 12V.

Algunas de las especificaciones técnicas de este motor se detallan en la Tabla A.2 del Anexo A (Pololu, 2024b).

Módulo control motores (*driver*)

Para el control de los motores escogidos anteriormente (ver Figura 1.17), existen diversos tipos de *Drivers* o controladores, cada uno con sus características y ventajas propias, las cuales se detallan a continuación:

- **Puente H (*H-Bridge*)**: Es uno de los *drivers* más utilizados en el control de los motores de corriente continua ya que permite controlar la velocidad y la dirección del motor al invertir la polaridad de la corriente que fluye a través de este. El modelo más común de driver puente H es el L298N.
- **Controladores PWM (Modulación de Ancho de Pulso)**: Este tipo de *drivers* son eficientes cuando se necesita un control preciso y eficiente de la velocidad del motor. En este caso, uno de los modelos más utilizados sería el BTS790B, ya que ofrece una protección contra sobrecorrientes y un control de velocidad bidireccional.

Para el prototipo a desarrollar, y teniendo en cuenta lo mencionado anteriormente, se ha optado por utilizar el ***driver* puente H L298N** (ver Figura 1.18)[22], ya que es uno de los módulos más comunes y económicos que existen en el mercado. Al tener un uso tan frecuente, se puede encontrar gran cantidad de documentación y ejemplos de

[21]Fuente: `https://www.pololu.com/product/4742/specs`
[22]Fuente: `https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html`

uso, lo que facilitará la labor de investigación en el momento de la implementación. El controlador PWM en este caso no sería necesario, ya que supondría un incremento del coste y no es necesario un control tan preciso de la velocidad ni una protección contra sobrecorrientes, debido a que no se va a trabajar con intensidades tan elevadas como para que esto sea necesario.

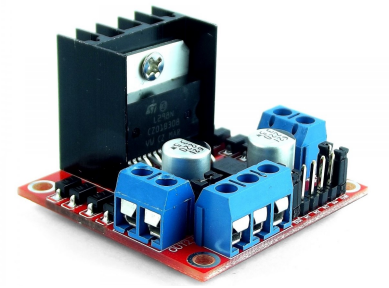


Figura 1.18: *Driver* puente H modelo L298N.

Algunas de las especificaciones técnicas de este *driver* de detallan en la Tabla A.3 del Anexo A (Naylamp Mechatronics, 2024a).

Microcontrolador

Para realizar las conexiones correspondientes y programar el control del robot, es necesario el uso de un microcontrolador. Para ello, se han barajado diversas opciones y tipos, las cuales se exponen a continuación:

- **Microcontroladores Arduino**: Arduino es una plataforma de código abierto muy conocida a nivel global por su facilidad de uso y por ser intuitiva. La gama de microcontroladores Arduino comprende una amplia variedad de placas, que varían en cuanto a la velocidad de procesamiento o el número de puertos E/S.

- **Raspberry Pi**: Es una computadora de bajo coste que puede ejecutar un sistema operativo completo. Es muy potente, por lo que está pensado para proyectos que requieren de mucho procesamiento, además de ofrecer una amplia gama de opciones de conectividad y es compatible con una amplia variedad de lenguajes de programación.

- **Microcontrolador ESP32**: Es un microcontrolador de bajo coste que incorpora Wifi y Bluetooth. Además, también es compatible con el entrono de programación de Arduino. Es ideal para proyectos que requieren de una conectividad inalámbrica (Ej. aplicaciones de control remoto).

- **Microcontroladores STM32**: Forman parte de los microcontroladores ARM Cortex-M desarrollados por STMicroelectronics.Tienen un alto rendimiento y una amplia gama de periféricos. Su programación generalmente se realiza en el entorno de desarrollo ATM32CubeIDE.

Para el control del robot autobalanceado se ha optado por utilizar el microcontrolador **Arduino Due** (ver Figura 1.19)[23], debido a su buena compatibilidad e integración con otros sensores. Este, destaca por su potente microcontrolador ARM Cortex-M3 de 32 bits, y ofrece una mayor capacidad de gestionar periféricos y sensores ya que proporciona una amplia gama de puertos de entrada/salida (E/S) que permitirán conectar de forma sencilla los componentes mencionados anteriormente. Para programarlo se utilizará **Arduino IDE** (ver Figura 1.1(b)), un entorno de desarrollo integrado de Arduino el cual incluye un editor y compilador de código, además de gestión de bibliotecas y placas y un monitor serie, con el que enviar y recibir datos del microcontrolador en tiempo real.

Figura 1.19: Microcontrolador Arduino Due.

Algunas de las especificaciones técnicas de este microcontrolador de detallan en la Tabla A.4 del Anexo A (I+D Electrónica, 2024).

Baterías

Las baterías se utilizarán para alimentar tanto la placa Arduino Due como el *driver* que controla los motores del robot. Se han implementado dos tipos de baterías para este propósito. Si bien es posible utilizar una sola batería junto con un regulador de voltaje, esta configuración puede no proporcionar la corriente suficiente para alimentar todo de manera adecuada. Además, si está todo alimentado desde la misma fuente, se pueden producir picos de corriente y causar daños a todo el equipo, por lo que no es lo más apropiado. Por ello, se ha decidido separar las etapas de potencia y control con sus respectivas baterías para así garantizar un suministro de energía adecuado y estable para ambos componentes del sistema. Las baterías utilizadas se detallan a continuación:

[23]Fuente: `https://www.didacticaselectronicas.com/index.php/sistemas-de-desarrollo/arduino/arduino-2/a000062-detail`

- **Batería de litio recargable 7,4 V a 6000 mAh**: Se utilizará para alimentar los motores desde las entradas del driver L298N (ver Figura 1.20).
- **Batería tipo *powerbank* 5 V a 4400 mAh**: Esta batería, mediante un conector Micro USB, alimentará el microcontrolador.

Figura 1.20: Batería de litio 7,4 V 6000 mAh.

Módulo Bluetooth

Este módulo permite establecer una conexión mediante la tecnología Bluetooth, lo que posibilita el control remoto del robot desde el *Smartphone*. Aunque existen varios módulos de este tipo en el mercado, los módulos JY-MCU, como el HC-06 (ver Figura 1.21), son populares por su accesibilidad y bajo consumo de energía. Estos módulos proporcionan una solución fácil de integrar con placas Arduino, ofreciendo los pines necesarios para la comunicación serie.

El HC-06 actúa exclusivamente como esclavo, lo que lo hace ideal para este proyecto, donde el Smartphone actúa como maestro. En contraste, el HC-05 puede funcionar como maestro o esclavo y sería más adecuado para establecer conexiones entre varios robots Arduino. La elección del HC-06 se basa en su capacidad para cumplir con los requisitos específicos del proyecto, garantizando una comunicación eficiente y fiable entre el robot y el dispositivo de control externo. (Víctor Esteban Falconi Loja, 2015)

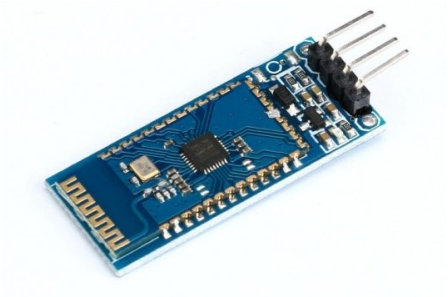


Figura 1.21: Módulo Bluetooth HC-06.

1.5. Simulación del movimiento del robot

La simulación del robot estará dividida en varias fases o etapas. Se comenzará con el diseño de la estructura del robot en un *software* de CAD 3D, para luego exportar las piezas a las herramientas de *Matlab* y proceder con la simulación. En la Figura 1.22 se presenta un esquema u organigrama que ilustra las fases de esta sección de forma visual.

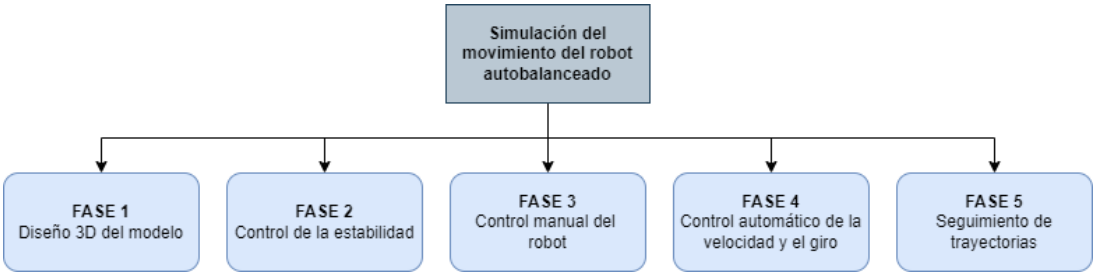


Figura 1.22: Fases de la simulación.

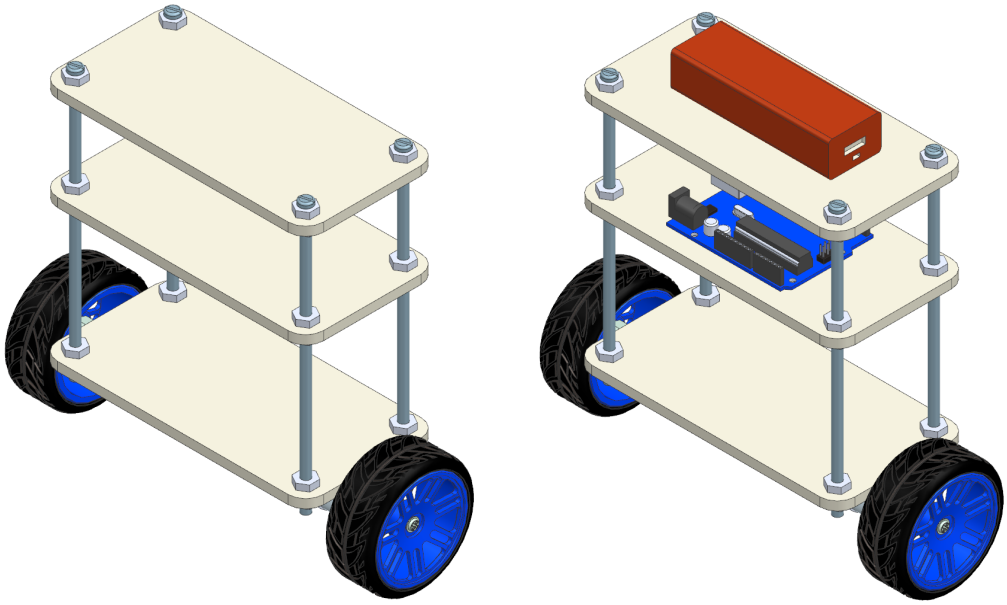
1.5.1. Diseño 3D del modelo

Para realizar correctamente la simulación del robot en *Matlab*, es necesario crear un modelo que sea lo más similar posible al vehículo que se utilizará en la implementación real. Esto permitirá exportar las piezas a la herramienta *Simscape Multibody* y asignarles sus características dinámicas correspondientes, asegurando que la simulación se asemeje lo máximo posible a la realidad.

Para el ensamblaje de las piezas, se ha utilizado el programa **Siemens NX**. Este *software* de diseño CAD 3D, desarrollado por la empresa *Siemens*, es uno de los más utilizados a nivel global en el ámbito empresarial debido a su interfaz extensa y a sus aplicaciones integradas de alto rendimiento. Este programa permite asignar propiedades dinámicas a cada pieza, asegurando que el material se asemeje en gran medida a sus características en la realidad.

Para el diseño del modelo, tal y como se ha explicado en el apartado correspondiente (véase Apartado 1.2.4: Modelo matemático del robot), según la teoría del péndulo invertido, el robot debe tener su centro de masas por encima de las ruedas para estabilizarse. Para lograr esto, se ha diseñado un prototipo con dos bandejas en la parte superior, donde se colocarán todos los componentes electrónicos. De esta forma, se asegura que el centro de masas del robot se encuentra por encima de las ruedas, ya que la mayor parte del peso estará situada en la parte superior del vehículo.

A continuación se muestra la estructura del robot ensamblada (ver Fig 1.23(a)) y la estructura final del robot con la mayoría de los componentes electrónicos necesarios para la simulación. (ver Fig 1.23(b))



(a) Estructura del robot sin los componentes

(b) Estructura del robot con el microcontrolador y la batería

Figura 1.23: Modelo 3D del robot en Siemens NX.

Por otro lado, la bandeja inferior se ha diseñado con el propósito de anclar los motores a su parte inferior mediante *brackets* en L. A su vez, en cada motor se colocará un *hub* que permitirá anclar la rueda al eje del motor. Las ruedas, por lo tanto, se colocarán en los extremos del robot, en la parte inferior de la bandeja de la base (ver Fig. 1.23). Para la sujeción de la estructura en general, se ha optado por utilizar cuatro varillas roscadas M6 dispuestas en las esquinas de las bandejas, fijadas con tuercas y arandelas. Como resultado, se obtiene una estructura robusta y resistente a golpes y caídas.

Posteriormente, será necesario seleccionar las piezas del ensamblaje de forma individual para exportarlas a un formato *.step* compatible con la herramienta de *Matlab*. Esto permitirá realizar correctamente la simulación del robot.

1.5.2. Control de estabilidad

Para comenzar con la simulación del vehículo, lo primero que se va a realizar es el el control de la estabilidad. Para ello, lo primero será conocer el entorno de la herramienta de *Matlab*, *Simescape Multibody* para así introducir las piezas que conformarán el prototipo.

Lo primero será insertar el bloque '**File Solid**'. Este bloque permite importar piezas o ensambles como cuerpos rígidos, además de proporcionar a cada pieza características

dinámicas como pueden ser masa, densidad o inercia. En esta simulación, las piezas del robot se introducirán en varias partes: en primer lugar se introducirá un bloque para el chasis al que se le irán uniendo partes, ya que es conveniente crear toda la estructura del robot como un bloque rígido. Por otro lado, se introducirán por separado los motores, las ruedas y los soportes para los motores, además de la batería y el microcontrolador. Todo esto irá unido a la estructura principal mediante la creación de diferentes *frames*.

- En primer lugar, se creará un *frame* F1 que irá conectado al primer motor.
- En segundo lugar, se creará un *frame* F2 que irá conectado al segundo motor.
- Por otra parte, se crearán los *frames* F3 y F4 que irán conectados al Arduíno y a la batería, respectivamente.

En la Figura 1.24 se puede observar el bloque *File Solid* de la estructura principal, a la que se le ha asignado una masa aproximada para que el programa pueda calcular los valores derivados de esta. También se pueden observar los diferentes *frames* creados, a los que irán conectadas el resto de partes.

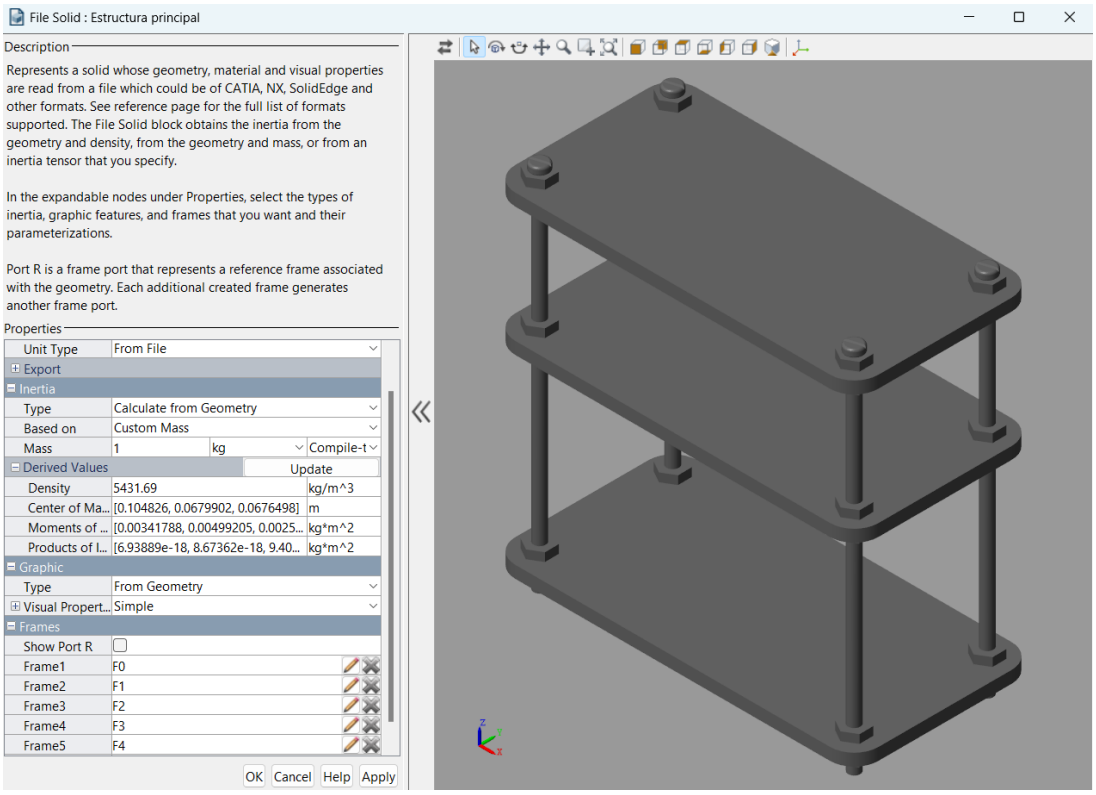


Figura 1.24: Ventana bloque *File Solid* de *Matlab*.

A partir de este punto, ya se podrán introducir en bloques distintos el resto de piezas exportadas desde *Siemens NX* para poder modelar el robot en la simulación. En el programa, no basta con unir las piezas, si no que mediante el uso de algunas herramientas específicas, habrá que añadir distintos bloques y articulaciones para poder unir y orientar las piezas correctamente.

Algunos de los bloques que se utilizarán para modelar el robot de la simulación serán (Mathworks, 2024):

- **Infinite Plane**: Este bloque representa un plano de unas medidas determinadas por el usuario en el que se situará el robot. En cuanto a los ejes de coordenadas, este plano tendrá la normal en la dirección del eje Z positivo. Este bloque es útil para simular una superficie 'infinita' sobre la cual se moverá el robot.
- **Rigid Transform**: Define una transformación rígida 3D entre los dos *frames*, es decir, sirve para situar la pieza rígida en una determinada posición u orientación. Con este bloque se consiguen rotar las piezas o darles una determinada separación (translación) en el espacio para poder implementar el prototipo correctamente.
- **Cartesian Joint**: Este bloque representa una articulación cartesiana entre dos *frames*. La articulación tiene tres grados de libertad, que se corresponden con los tres ejes. Esto, permite el movimiento en los tres planos. Por ejemplo, una articulación cartesiana podría permitir que un brazo robot se mueva hacia adelante/atrás (eje x), izquierda/derecha (eje y), y arriba/abajo (eje z).
- **Revolute Joint**: Representa una articulación de revolución entre dos piezas. Esta articulación proporciona un grado de libertad de rotación sobre un determinado eje y hace que los orígenes y la dirección de un eje coincidan. En este caso, la rotación solo se podrá realizar en el eje z, por lo tanto, si se desea rotar otro eje, habrá que combinarlo con el bloque *Rigid Transform*.
- **Spatial Contact Force**: Este bloque aplica fuerzas normales y de fricción a las dos geometrías a las que está conectado. Las fuerzas aplicadas son iguales y opuestas, y se encuentran a lo largo de una línea de acción común. Es esencial para simular interacciones físicas entre objetos en un entorno de simulación, haciendo que esta sea lo más real posible.

Una vez detalladas las utilidades de cada bloque, se procederá con el modelado del robot. Se inicia con un plano infinito que servirá como suelo, con dimensiones iniciales de 2x2 metros. Se conecta al suelo una transformación rígida para establecer una separación de 0,15 cm en el eje z. Esto garantiza que el robot no comience la simulación en contacto con el suelo y pueda estabilizarse al principio. Posteriormente, se agrega una articulación de tipo cartesiano, que proporcionará al robot los tres grados de libertad necesarios. Esta articulación, mediante la activación de sensores específicos, permitirá medir la posición y la velocidad del robot en los ejes xy.

Posteriormente, para medir el ángulo de estabilidad, se colocará una transformación rígida (*Rigid Transform*) con una rotación de 90° en el eje y. Esto permitirá que el eje normal al robot sea el eje z, para luego añadir una articulación de revolución que permitirá medir el ángulo de estabilidad. Esta secuencia es necesaria para garantizar que la articulación de revolución funcione correctamente sobre el eje z. Primero, se rotan los ejes de la figura para que el eje z esté en la posición deseada para la articulación, y luego se aplica la articulación de rotación. Después de esto, se aplicará otra transformación rígida, pero esta vez con una rotación de -90° en el eje y, para establecerlos en su orientación original.

Todo lo anterior se integrará en la estructura principal como el *Frame* 0, es decir, el marco que posibilita los movimientos del robot en las direcciones adecuadas y permite la medición del ángulo de estabilidad del robot respecto a su vertical.

Para simular las ruedas, primero se utilizará el bloque *Rigid Transform* para separar los motores del chasis inferior y ubicarlos en la posición deseada con respecto a la estructura principal. Además, se colocará el soporte del motor en la posición correcta en relación con el chasis. Todo esto se unirá al motor. A continuación, este se conectará a la rueda mediante una articulación de revolución, a través de la cual se introducirá el par de cada rueda como un valor determinado.

La ruedas irán unidas al suelo mediante el bloque *Spatial Contact Force* o fuerzas de contacto espacial. Este bloque permite modelar las fuerzas de contacto entre dos piezas de la simulación. para ello, será necesario especificar algunos parámetros.

Los parámetros relacionados con las fuerzas normales al robot serán:

- '**Stiffness**': Contante que modifica la rigidez del robot con el suelo.
- '**Damping**': Coeficiente de amortiguamiento entre el robot y el suelo.
- '**Transition Region Width**': Zona en la cual la fuerza del resorte alcanza su máximo valor.

Los parámetros relacionados con las fuerzas de fricción entre el robot y el suelo serán:

- '**Coeficient of Static Friction**' o coeficiente de fricción estática. Este parámetro, establece la conexión entre la intensidad de la fuerza de fricción y la fuerza normal cuando la velocidad tangencial se aproxima a cero.
- '**Coeficient of Dynamic Friction**' o coeficiente de fricción dinámica. Este coeficiente, asocia la fuerza de fricción con la fuerza normal en situaciones de alta velocidad tangencial.
- '**Critical Velocity**' o valor de velocidad crítica: Área donde la fuerza del resorte alcanza su punto máximo. Es la velocidad tangencial particular que determina el coeficiente de fricción que será aplicado.

Los valores que se le han asignado a cada una de las fuerzas de contacto quedan definidos en la Figura 1.25.

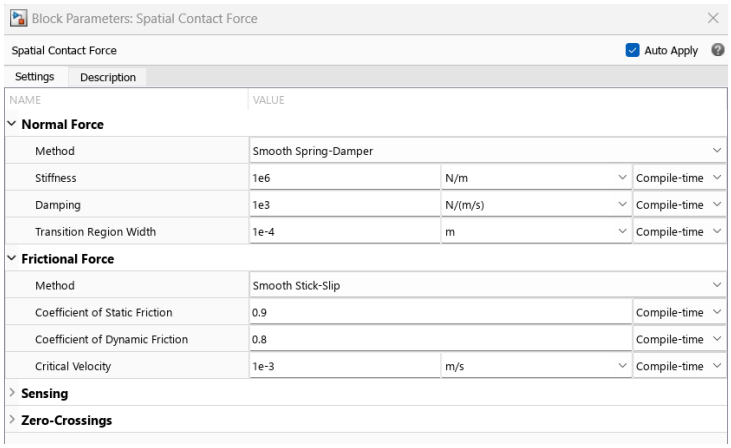


Figura 1.25: Ventana del bloque *Spatial Contact Forces* de *Simescape Multibody*.

El diagrama de bloques de Simescape Multibody para modelar la estabilidad del robot queda definido en la Figura 1.26:

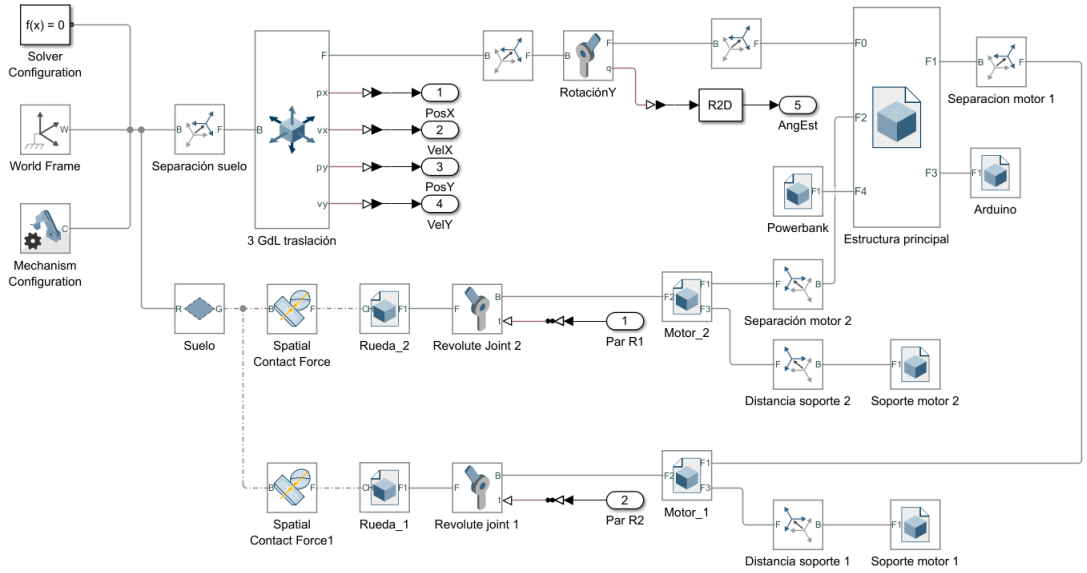


Figura 1.26: Diagrama de bloques para el modelado de la estabilidad del robot.

Por lo tanto, las entradas son los pares R1 y R2, que se aplicarán a cada rueda. Estos pares, estarán definidos por el ángulo de estabilidad obtenido a la salida del controlador PD, el cual, a su vez, recibe como entrada el ángulo de estabilidad real medido por el robot. Esto se puede visualizar en la Figura 1.27, donde se han añadido como salidas las posiciones y las velocidades en los ejes xy. Aunque estas salidas no son útiles en esta etapa, ya que el robot tiene estabilidad pero no movimiento, serán útiles en simulaciones posteriores, donde se podrán realizar gráficas de las diferentes posiciones y velocidades en función del tiempo.

Para el diseño del control de la estabilidad, se ha decidido implementar un controlador PD. No se ha optado por un regulador PI, ya que la referencia es 0 y, por lo tanto, no es necesario corregir este valor. Es decir, el controlador PI se utilizaría si se quisiera mantener el robot en una posición específica del ángulo de inclinación, pero ese no es el objetivo en este caso. El objetivo es corregir el ángulo de inclinación para acercarlo lo máximo posible a la referencia.

Para determinar los valores de las ganancias K_p y K_d, se ha realizado un ajuste experimental. Primero, se ha fijado K_d en 0 para hallar un valor adecuado de la ganancia proporcional K_p, de manera que el robot adquiera suficiente fuerza para no caerse. Una vez ajustado este valor, se ha incrementado gradualmente la ganancia derivativa K_d para optimizar el movimiento. Finalmente, los valores obtenidos han sido: $\mathbf{K_p = 1{,}50}$ y $\mathbf{K_d = 0{,}10}$

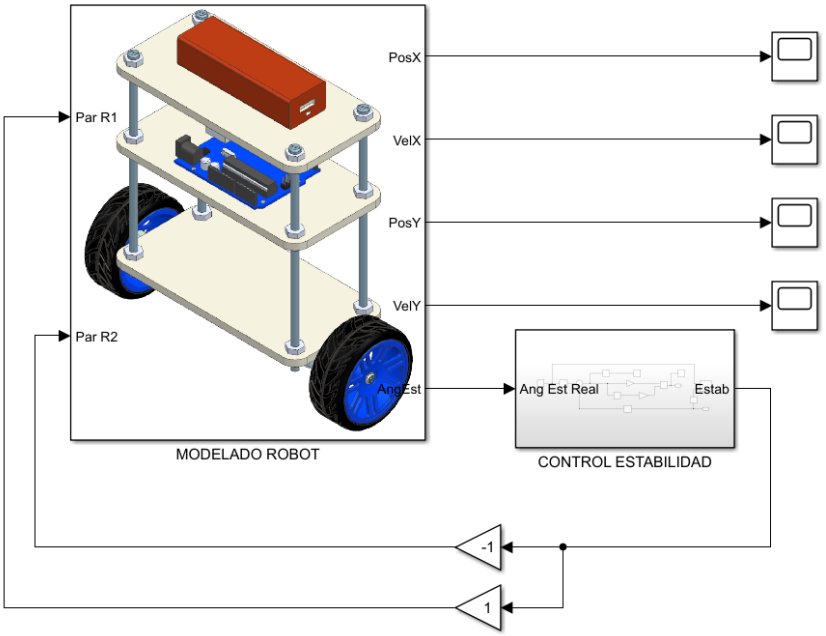


Figura 1.27: Modelo general para el control de la estabilidad del robot.

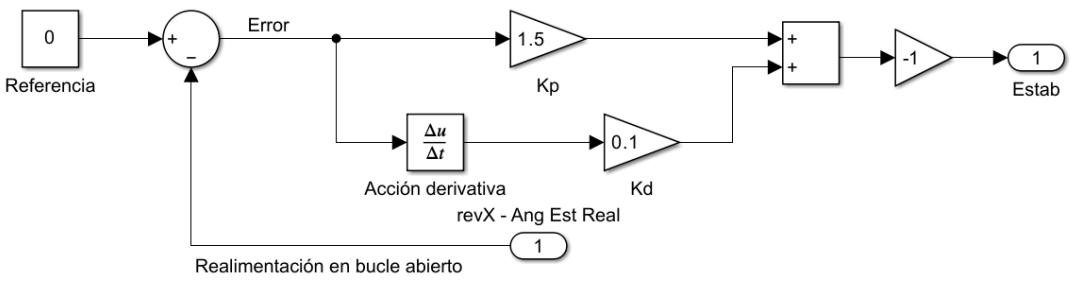
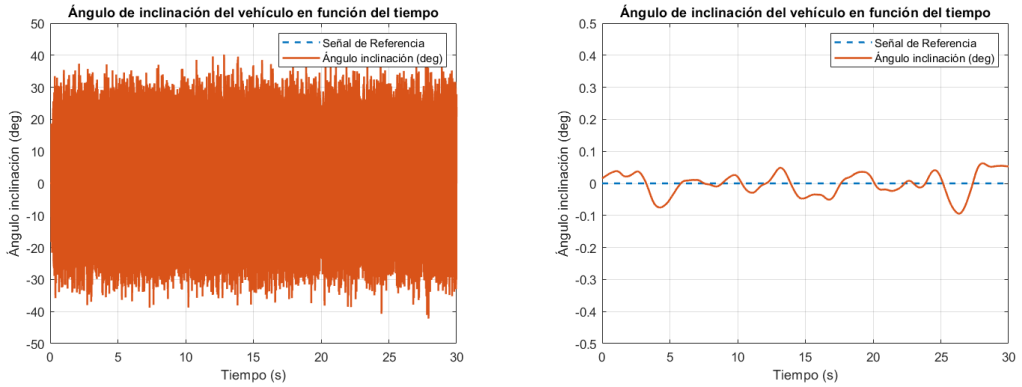


Figura 1.28: Regulador PD para la estabilidad del robot.

Después de implementar el controlador PD y ajustar correctamente los valores de las ganancias, se logra que el robot mantenga la estabilidad en todo momento (ver Figura 1.30). A pesar de las perturbaciones externas, el robot las corrige para volver al valor de referencia cero. En la Figura 1.70 se observan las gráficas del ángulo de inclinación del vehículo en función del tiempo. Se destaca que al visualizar las gráficas se presenta un considerable nivel de ruido, por lo que se aplica un **filtro Butterworth** para suprimir ciertas frecuencias. Para este propósito, se establece una frecuencia de muestreo $\mathbf{f_s = 1000Hz}$, una frecuencia de corte $\mathbf{f_c = 0{,}25Hz}$ y un filtro de **orden 2**. Este filtro está diseñado para para proporcionar una respuesta plana en la banda de paso y luego atenuar gradualmente las frecuencias por encima de la frecuencia de corte.



(a) Gráfica del ángulo de inclinación sin filtro (b) Gráfica del ángulo de inclinación con filtro

Figura 1.29: Gráfica del ángulo de inclinación respecto al tiempo.

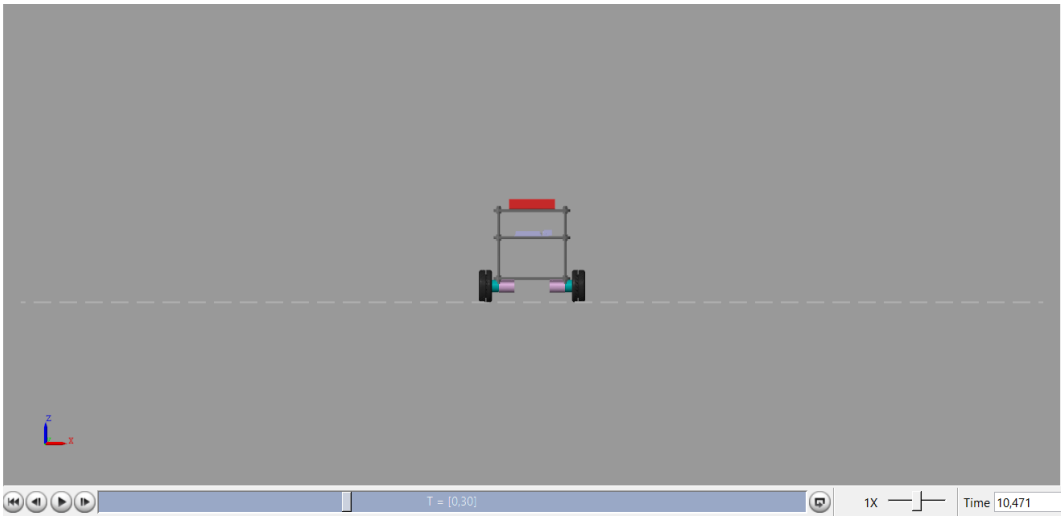


Figura 1.30: Simulación de la estabilidad del robot.

1.5.3. Control manual del robot

El control manual del robot se realizará mediante el *joystick* de un *gamepad* o mando para juegos. A través de este, mediante las **acciones del usuario**, se mandarán unos determinados valores, que, multiplicados por una ganancia introducirán un valor de giro o de dirección, a una velocidad constante. Para ello lo único que habrá que tener en cuenta es que para una rueda el giro se sumará a la velocidad lineal y para la otra rueda, se restará. La resta se realiza para permitir el movimiento del giro del robot, ya que mientras una rueda gira en una dirección, la otra girará en sentido contrario, permitiendo así un control direccional del robot.

Las funciones que se han implementado en el control manual del robot quedan definidas en la Figura 1.31:

Figura 1.31: Funciones del mando para el control manual.

Es importante destacar que, aunque se realice un control manual del robot, la prioridad principal siempre será el control de la estabilidad, por ello nunca debe de comprometerse. En cualquier situación, el controlador implementado actuará para garantizar que el robot permanezca estable y en equilibrio, incluso mientras se realizan otras funcionalidades como las que aparecen en el mando de la Figura 1.31.

Para realizar el diagrama de bloques en *Simecape Multibody* del control manual del robot, se tomará como base el control de estabilidad (véase Apartado 1.5.2: Control de estabilidad), realizando algunas modificaciones sobre estos esquemas.

Una vez se han definido las funcionalidades que se van a implementar, lo primero será modificar el esquema de control para poder obtener todas las magnitudes necesarias y añadir los bloques necesarios. En la Figura 1.33 se muestra el esquema de las articulaciones que unen el suelo con el chasis del robot. Como la velocidad en este caso será constante, solo se activarán los sensores de posición de los ejes xy del bloque 'Cartesian Joint', desactivando los de la velocidad. Aparte, se ha añadido una articulación de revolución en el eje z, para así permitir que el robot gire entorno a este eje. Por otro lado, se ha añadido un bloque antes de obtener el giro en el eje z (ver Figura 1.32) y su función es la de dar la posición angular entre $\pm 180^{o}$ (diente de sierra) para cuando el robot de una vuelta completa sobre sí mismo, si no se irían sumando todos los ángulos consecutivos (180^{o}, 360^{o}, ...) y el resultado no será correcto.

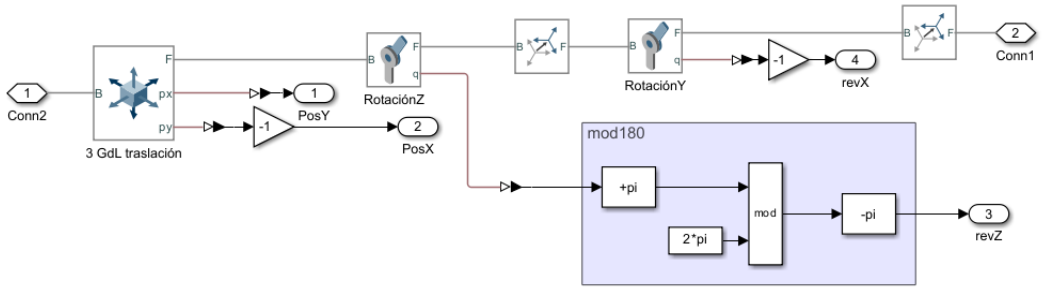


Figura 1.32: Grados de libertad en las articulaciones para el control manual.

En el esquema de bloques de control del robot (ver Figura 1.33) se ha añadido la función que realizará el botón 'A', que será la de introducir perturbaciones. Para ello, el valor leído (será 1 en el caso de que el botón se mantenga pulsado) se convertirá a un dato de tipo *double* para introducir ese dato a un bloque denominado '**External Force and Torque**', con el que se realizará una fuerza en el eje x sobre la parte más alta del robot, en este caso, la *powerbank*.

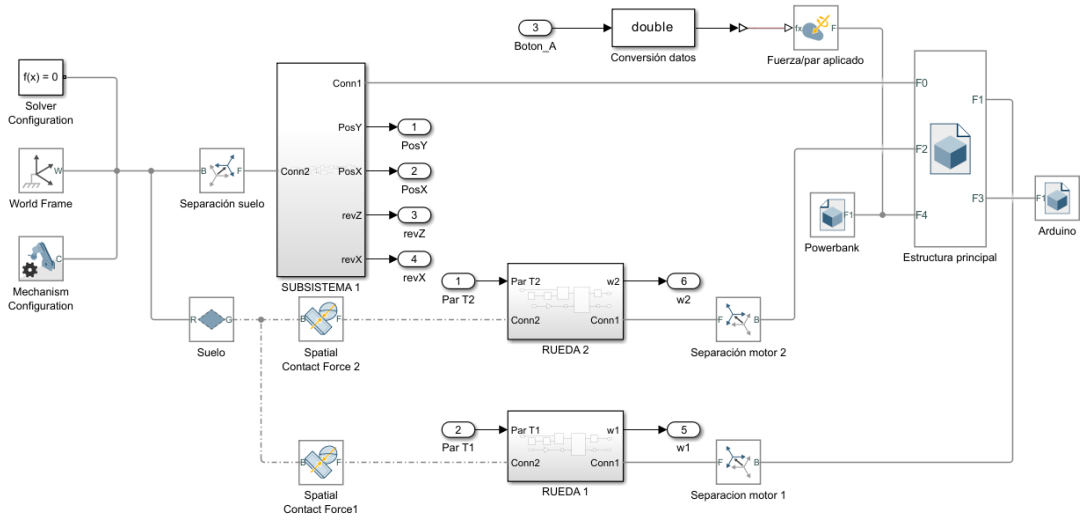


Figura 1.33: Diagrama de bloques para el modelado del control manual del robot.

Lo último será implementar los controles del mando para realizar las funciones del control manual relacionadas con el movimiento. Para ello, se hará uso del bloque '**Joystick Input**', el cual reconoce automáticamente el mando que se conecta al ordenador mediante un puerto USB. Este bloque tiene tres puertos de salida:

1. '**Axes**': Proporciona las salidas correspondientes a los *joysticks*.
2. '**Buttons**': Proporciona las salidas correspondientes a los botones.
3. '**Point of view**': Proporciona las salidas relacionadas con los botones de movimiento en forma de cruz.

Según la Figura 1.31, se utilizarán el *joystick* izquierdo, el trasero y el botón 'A', como se muestra en la Figura 1.34. Es importante destacar que, para poder utilizar las salidas del bloque de control del mando, se emplearán dos bloques '**Demux**'. Estos bloques se encargan de descomponer la señal de entrada, que se presenta en forma de vector, permitiendo generar señales individuales separadas.

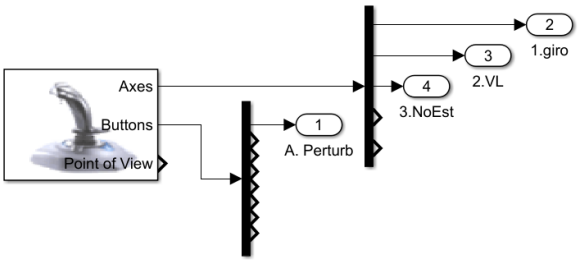


Figura 1.34: Implementación del bloque para el control del *gamepad*.

Por último, se realizará el esquema general de las conexiones entre las entradas y las salidas para realizar el control manual del robot (ver Figura 1.35). Las salidas del mando para la velocidad lineal y giro se multiplicarán por una ganancia para evitar la introducción de valores excesivos. Es importante destacar que la velocidad se sumará o restará de igual manera a ambas ruedas, mientras que el giro se sumará a una rueda y se restará de la otra, como se ha explicado anteriormente. Por otro lado, la salida del botón 'A' se conectará directamente a la entrada correspondiente del botón (ver Figura 1.33).

En cuanto al botón de emergencia para anular la estabilidad, su control difiere del resto (ver Figura 1.35). Cuando este botón no está pulsado, emite un valor de 0. Este valor, sumado a una constante de valor 1 y multiplicado por la estabilidad, no introducirá ninguna modificación en la estabilidad. Sin embargo, cuando este botón se pulsa, se obtienen valores negativos. Al sumar una constante de valor 1 a estos valores negativos, se obtiene un valor positivo que, multiplicado por la estabilidad, resultará en su desactivación mientras el botón se mantenga accionado.

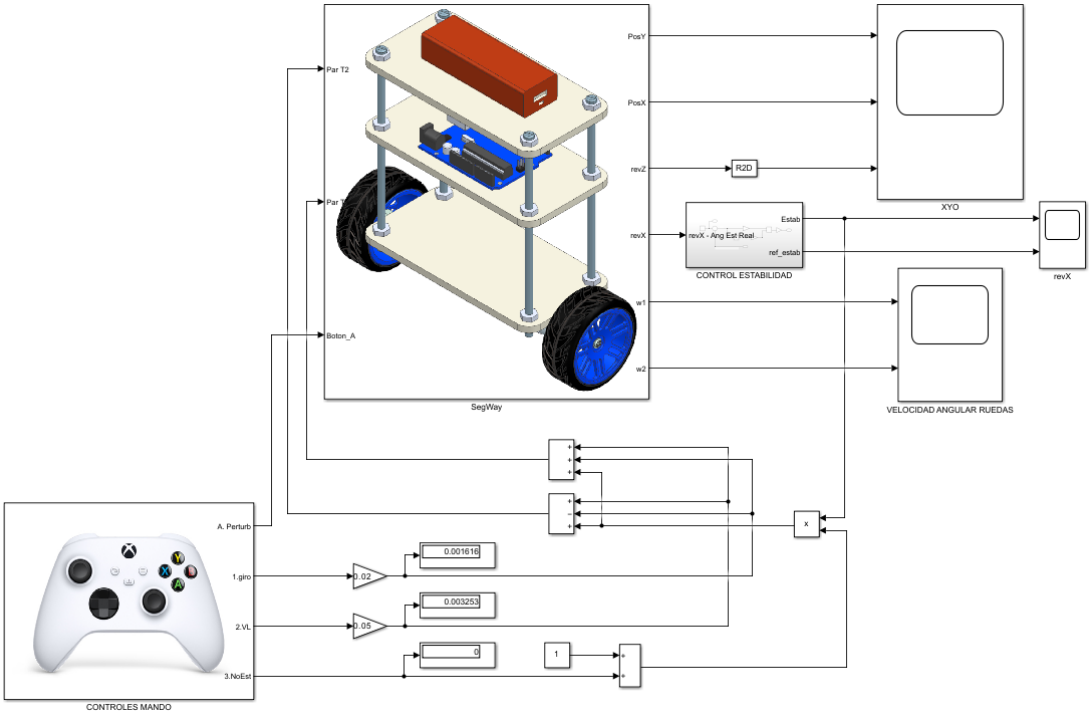


Figura 1.35: Modelo general para el control manual del robot.

Los bloques '**scope**' muestran las señales que se generan durante la simulación. Estos bloques se han añadido en algunos puntos de la simulación para poder visualizar el correcto funcionamiento mediante diversas gráficas.

A continuación, se puede visualizar la gráfica del ángulo de inclinación (estabilidad) en radianes respecto al tiempo en segundos, comprobando así que el robot mantiene su estabilidad a pesar de realizar movimientos y giros. El cambio en el ángulo de inclinación de positivo a negativo corresponde a un cambio de dirección del robot, como se muestra en la Figura 1.36. Los valores del ángulo de inclinación son muy bajos, aunque es difícil que estén más cerca de 0 como ocurría en el control de la estabilidad, dado que en este caso, el robot está en constante movimiento.

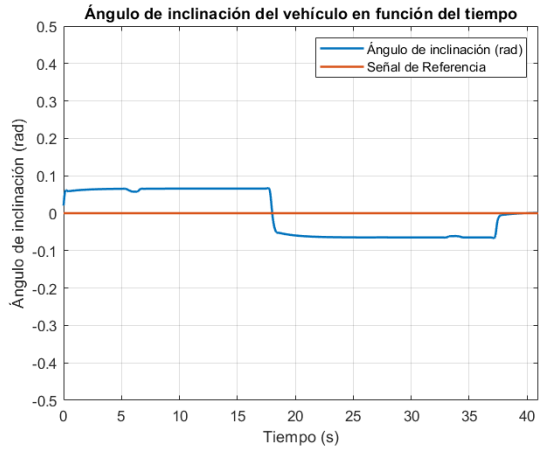
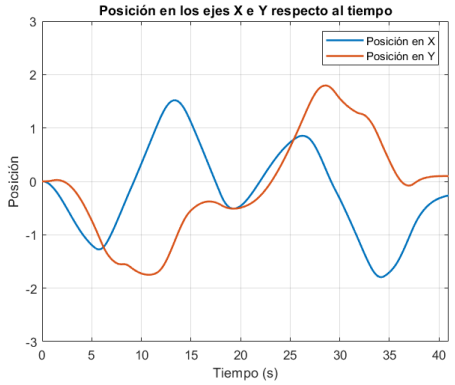
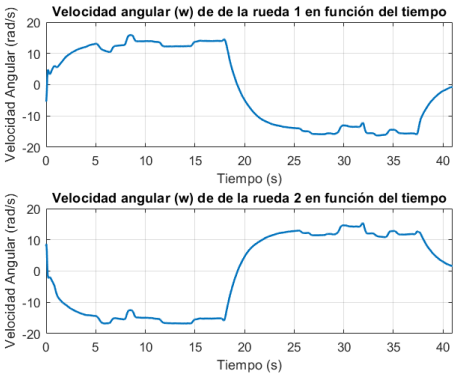


Figura 1.36: Gráfica del ángulo de inclinación respecto al tiempo en el control manual.

Por último, se pueden visualizar dos gráficas que muestran la posición del robot en el plano xy respecto al tiempo y la velocidad angular de las ruedas, que es igual pero de signo contrario.



(a) Gráfica posición en metros planos XY



(b) Gráfica de la velocidad angular de las ruedas

Figura 1.37: Gráficas de la posición xy y velocidad angular de las ruedas.

1.5.4. Control automático de la velocidad lineal y angular

El control automático toma como base el control manual del robot, aunque con pequeñas modificaciones. En este caso, las velocidades ya no serán constantes, es decir, contra más alto sea el valor enviado por el *joystick* o por la referencia de velocidad, el robot acelerará hasta llegar a alcanzarla.

Para modificar el esquema de control se comenzará modificando el subsistema correspondiente a las articulaciones, para así poder obtener las salidas necesarias para las entradas a los nuevos bloques que se van a implementar o para realizar diferentes mediciones. La modificación incluye el ajuste de los valores positivos o negativos mediante el uso de ganancias que multiplicarán los valores a la salida de los bloques, para obtener así una salida acorde al movimiento del vehículo.

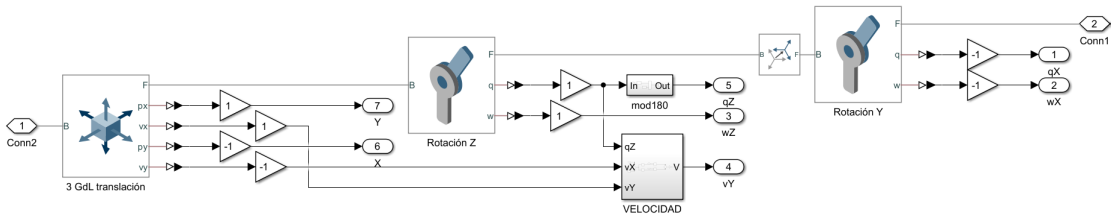


Figura 1.38: Grados de libertad en las articulaciones para el control automático.

En la Figura 1.38, se ha implementado un bloque que tiene como entradas las velocidades leídas por la articulación que proporciona los tres grados de libertad al robot, junto con la posición de la articulación que permite la rotación en el eje z. Este bloque se ha implementado debido a la necesidad de obtener la velocidad relativa al robot, no la velocidad respecto al 'mundo'. Para calcular la velocidad respecto al robot, es necesario considerar la velocidad en los ejes xy y la posición angular de la rotación en el eje z. Esto permitirá obtener la velocidad del robot en relación con su propio marco de referencia (ver Figuras 1.39 y 1.40)

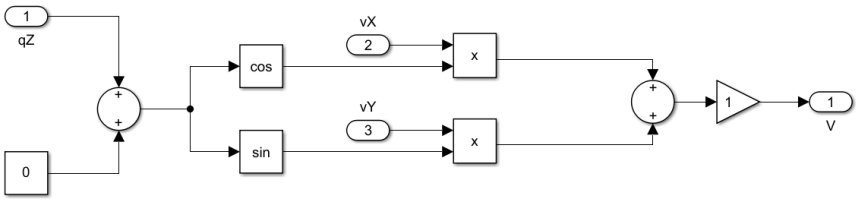


Figura 1.39: Cálculo de la velocidad relativa al robot.

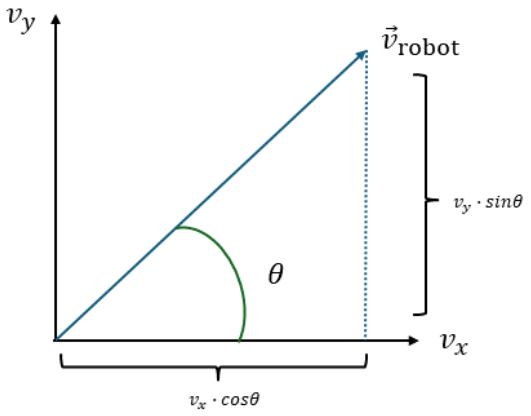


Figura 1.40: Cálculo de la velocidad relativa al robot.

Con los cambios anteriores realizados sobre el esquema del modelo del robot, este quedaría definido de la siguiente forma: (ver Figura 1.41)

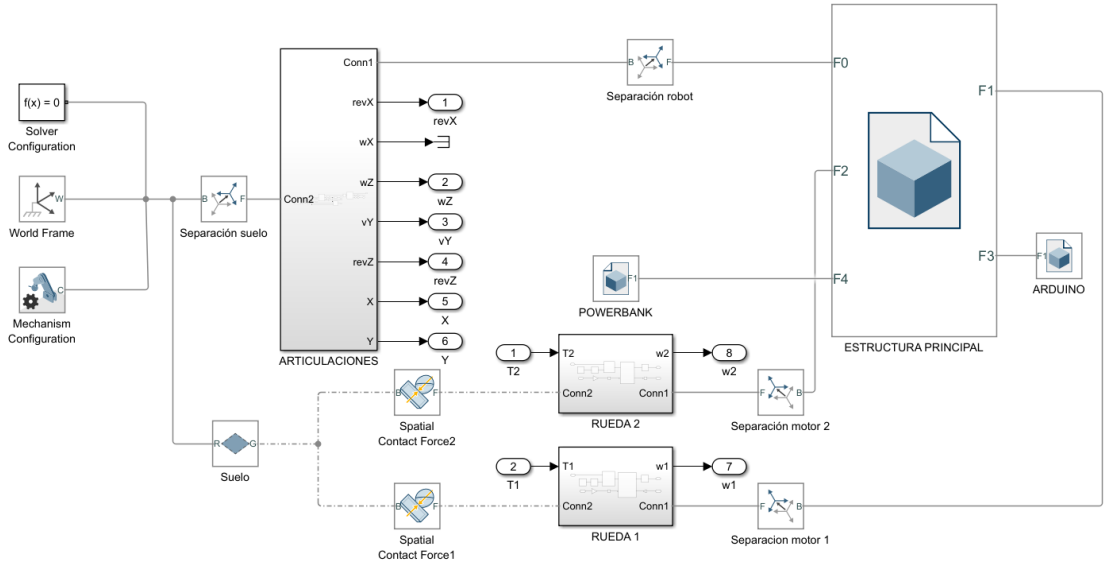
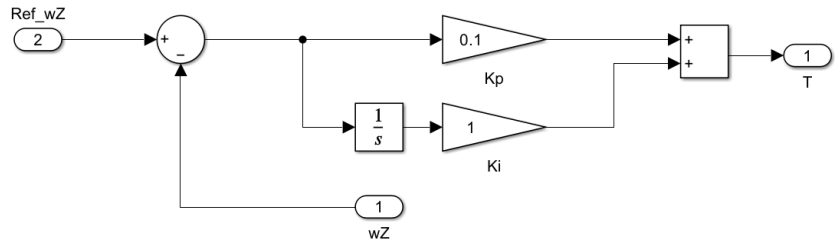


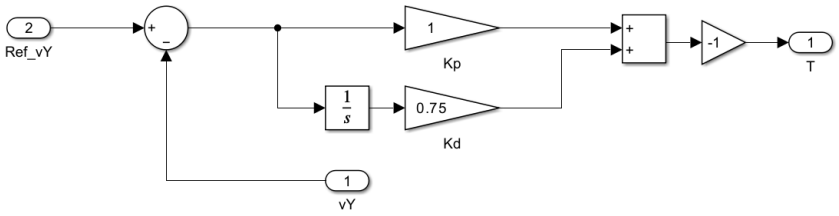
Figura 1.41: Diagrama de bloques para el modelado del control automático del robot.

Como se puede observar en la figura anterior, se han añadido como salidas la velocidad angular w_z, la velocidad calculada v_y (ver Figura 1.39) y la velocidad angular de cada una de las ruedas. En este caso, como se va a realizar un control automático de la velocidad y el giro, estas medidas son necesarias ya que serán las referencias de los dos controladores que se implementarán para ello. Todas las salidas de estos controladores se conectarán directamente a las ruedas mediante sumadores, teniendo en cuenta que la salida del control (giro) se tiene que sumar en una rueda y restar en la otra.

Para el **control de orientación** se utilizará un controlador PI (ver Figura 1.42(a)), ya que permite corregir errores de orientación de una forma precisa y eliminar el error en régimen permanente. La acción proporcional da una respuesta rápida al error actual, mientras que la acción integral corrige el error acumulado a lo largo del tiempo. Por otro lado, para el **control de la velocidad** se utilizará un controlador PD (ver Figura 1.42(b)) ya que permite una respuesta rápida y reduce la sobreoscilación, de tal forma que este regulador es necesario para mantener la velocidad estable sin oscilaciones pronunciadas.



(a) Controlador PI para la velocidad angular



(b) Controlador PD para la velocidad lineal

Figura 1.42: Controladores PI y PD para el control de la velocidad angular y lineal.

Para el ajuste de las ganancias de los controladores, en el caso del **control de orientación**, se comienza con una ganancia proporcional (K_p) de un valor bajo y se aumenta poco a poco hasta que el robot comience a corregir su orientación de manera eficiente. Después de ajustar K_p, se aumenta la ganancia integral (K_i) hasta alcanzar un valor suficiente para mantener la estabilidad del robot.

Por otro lado, en el caso del **control de velocidad lineal**, se sigue el mismo enfoque que para el control de orientación. Primero se comienza con una ganancia proporcional (K_p) reducida y se va aumentando hasta que el robot alcance la velocidad deseada sin sobreoscilar. Una vez conseguido esto, se incrementa la ganancia derivativa (K_d) hasta un valor suficiente para no perder la estabilidad, ya que es el objetivo principal.

Con los controladores implementados, el esquema general de las conexiones del robot quedaría definido de la siguiente forma: (ver Fig 1.43)

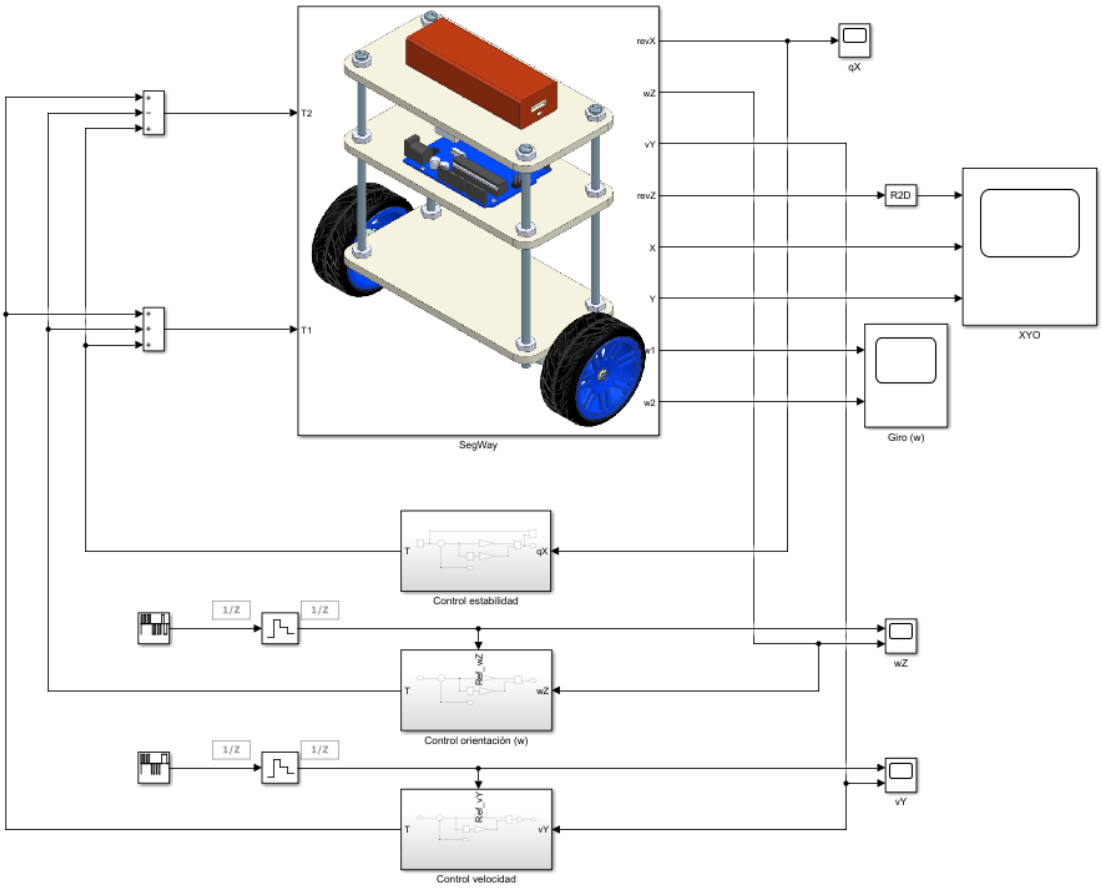


Figura 1.43: Modelo general para el control automático de la velocidad y el giro.

Por último, faltaría introducir las referencias de ambos controladores (ver Figura 1.42). En este caso, se han optado por dos opciones que serían igualmente válidas. La primera opción (véase Apartado 1.5.3: Control manual del robot), sería que el usuario, mediante un *joystick* controle manualmente el vehículo. La diferencia respecto a la simulación anterior es, que ahora contra más se incremente el valor introducido en el *joystick*, el robot acelerará o girará más rápido.

La segunda opción y por la que se ha optado finalmente consiste en programar un vector de valores que se introducirán mediante el bloque '**Repeating Sequence Stair**' seguido de un **retenedor de orden cero** (*zero order hold*) para suavizar esos cambios bruscos en la entrada. En este caso y para realizar la secuencia que se muestra en la Figura 1.45, los valores por los que se han optado quedan definidos en la Figura 1.44 con un tiempo de muestreo de $2ms$.

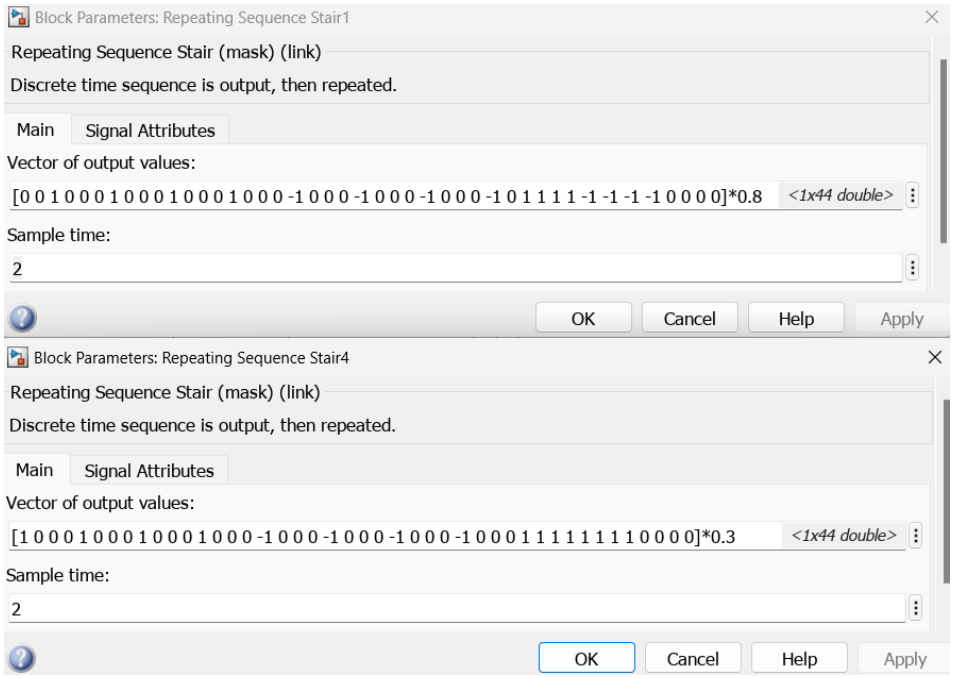
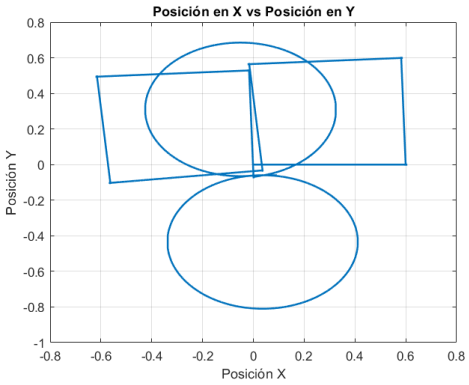


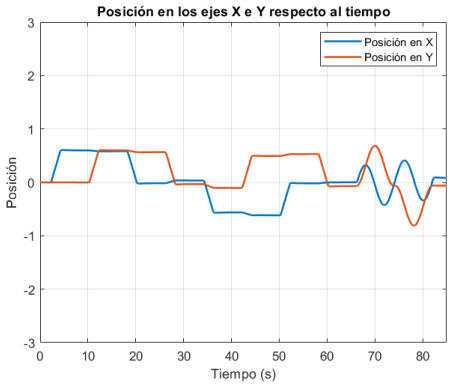
Figura 1.44: Valores introducidos en la referencia de los controladores.

Realizando todo lo anterior, se ha conseguido un control automático de la velocidad lineal y el giro. Al igual que se ha realizado en el control manual, se han implementado algunos bloques (*Scope*) para poder visualizar correctamente diversos valores durante la simulación.

Los valores introducidos en la referencia anteriormente hacen que el robot realice unos movimientos que quedan reflejados en la Figura 1.45(a), donde se muestra una vista cenital del plano xy. Como se puede observar, los movimientos no son muy precisos ya que se le están introduciendo determinados valores de velocidad y giro, no se le está introduciendo una trayectoria con unos puntos que seguir. También se muestran en la Figura 1.45(b) los valores de la posición del robot en metros en los ejes xy, que se corresponden con las referencias introducidas anteriormente.



(a) Vista cenital del plano XY del control automático.



(b) Valores en los planos XY de la posición en metros

Figura 1.45: Gráficas de la posición xy del robot.

En la Figura 1.46 se puede observar el control de orientación durante esta simulación. Como se puede apreciar, este cambia ligeramente respecto a las anteriores simulaciones. Esto se debe a que cada vez que se realiza un giro o un movimiento brusco, el valor del ángulo de inclinación aumenta considerablemente. Sin embargo, gracias a la acción de los controladores implementados para la velocidad lineal y angular, el valor del ángulo de inclinación se reduce hasta alcanzar prácticamente la referencia, manteniendo así la estabilidad en todo momento.

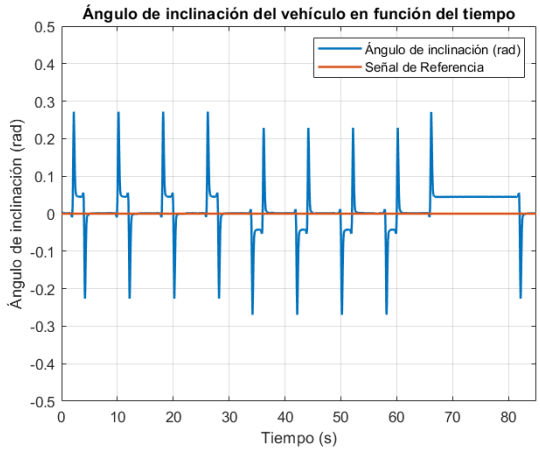
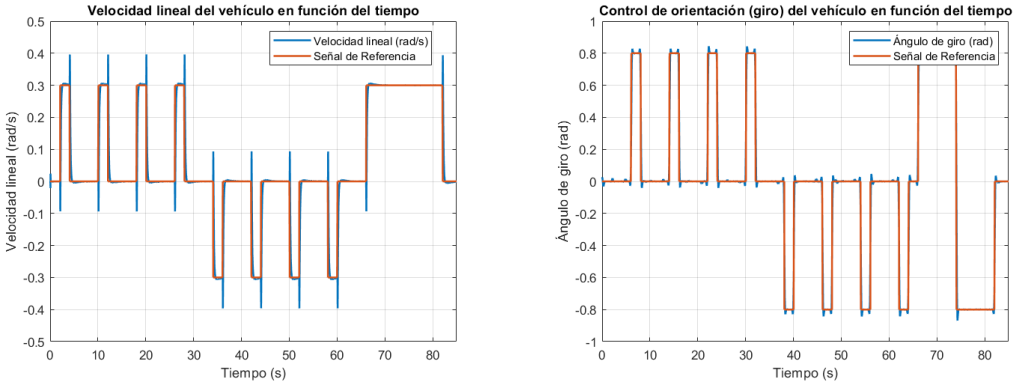


Figura 1.46: Gráfica del ángulo de inclinación respecto al tiempo en el control automático.

En la Figura 1.47 se puede observar que el robot sigue en gran medida las referencias para el control de orientación y para el control de la velocidad lineal. Como se ha mencionado anteriormente, aparecen algunos 'picos' que indican que el robot alcanza un valor de giro o de velocidad lineal demasiado elevado. Sin embargo, los controladores implementados anteriormente permiten corregir rápidamente estos valores y ajustarlos al valor de la referencia introducido.



(a) Velocidad lineal del robot.

(b) Velocidad angular del robot.

Figura 1.47: Gráficas de velocidad lineal y angular junto a la referencia introducida.

Al igual que en los casos anteriores, en la Figura 1.48 se puede observar que la velocidad angular de las ruedas es igual pero de signo contrario, para mantener la estabilidad en todo momento.

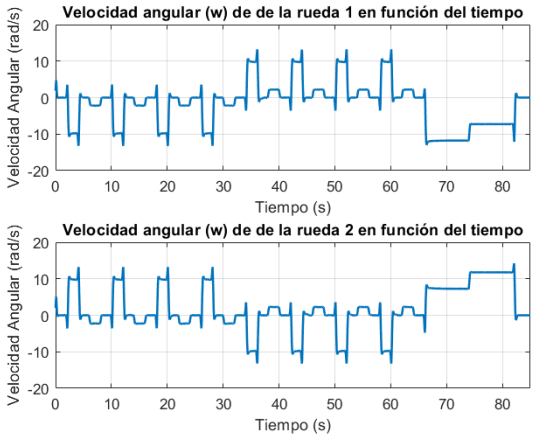


Figura 1.48: Gráfica de la velocidad angular de las ruedas en el control automático.

1.5.5. Control de trayectorias

Por último, para implementar en *Simscape Multibody* el control del robot para el seguimiento de trayectorias, harán falta algunas modificaciones respecto a la simulación anterior. En este caso, el robot ya no seguirá referencias de velocidad lineal y angular, sino que pasará por determinados puntos que conformarán la trayectoria completa del robot. Es importante destacar que la simulación realizada, mediante cambios mínimos en el código de la programación de las trayectorias, funcionará para cualquier tipo de forma o curvas. En este caso, a diferencia de las simulaciones anteriores, el usuario no tendrá que realizar ningún control sobre el robot, tan solo definir la trayectoria que se quiera realizar, y este, la seguirá automáticamente.

En cuanto al esquema del del modelo del robot, será el mismo que en el apartado anterior (véase Apartado 1.5.4: Control automático de la velocidad lineal y angular), aunque realizando algunas modificaciones sobre este (ver Figura 1.49). En primer lugar, se ha añadido un bloque '**Spline**' que cogerá los valores de los puntos introducidos para la trayectoria y los dibujará en el suelo, para así comprobar que el robot la está realizando correctamente. El segundo cambio que se ha realizado es la aparición de un bloque cilíndrico en el punto destino al que se va a dirigir el robot. Esto se hará introduciendo las posiciones de destino xy en una '**Planar Joint**', es decir, una articulación que permite el movimiento en el plano, y conectando a su vez este bloque con un '**Cilyndrical Solid**', es decir, un sólido con forma cilíndrica.

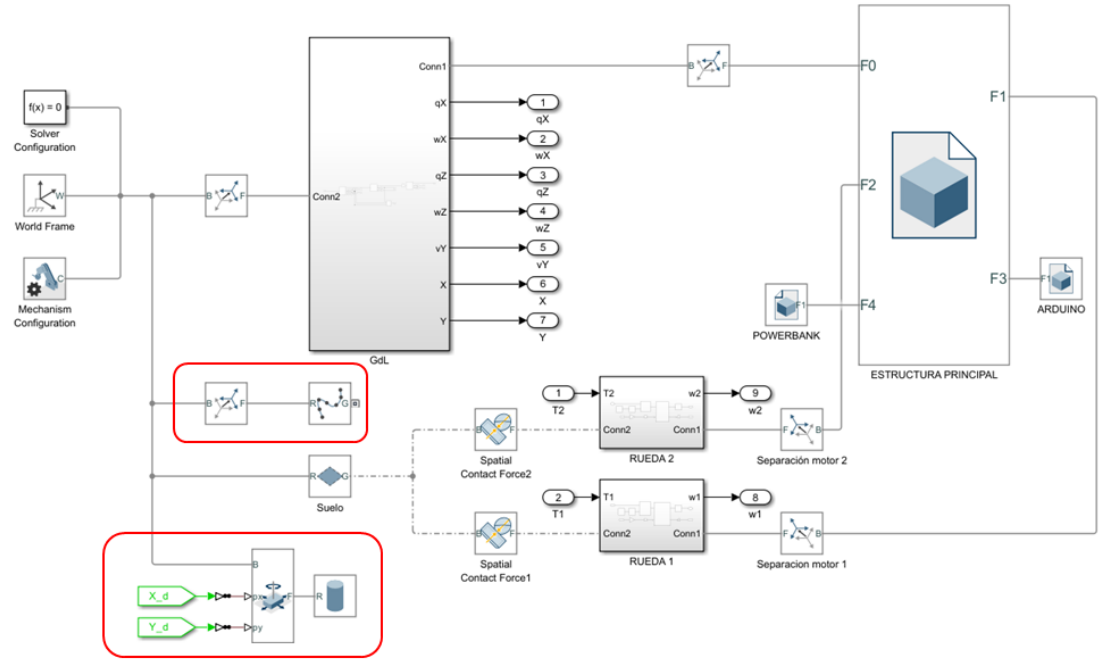


Figura 1.49: Diagrama de bloques para el modelado de las trayectorias.

En este caso, la velocidad del robot será constante durante toda la trayectoria, por lo tanto, la referencia del control de velocidad será un valor constante. Se ha considerado que 0.25 es un valor considerable en el que el robot podía realizar todo tipo de trayectorias. Para realizar correctamente la simulación, también habrá que tener en cuenta que se introducirá en el control de giro (velocidad angular), el valor de la posición angular q_z en vez de con w_z, como se había realizado en simulaciones anteriores.

Para obtener un control de trayectorias óptimo, es crucial que el robot se oriente hacia su próximo destino, que será el punto consecutivo al actual. Luego, el robot se dirigirá hacia este punto para alcanzar los valores deseados de posición. Con este fin, se han creado dos vectores que contienen las coordenadas xy de las posiciones a las que el robot debe llegar. Este enfoque permite que una vez que el robot alcance un destino, se defina una nueva ubicación hacia la cual dirigirse, y así sucesivamente, hasta completar la trayectoria deseada.

Para ello, en la Figura 1.50 se muestra el esquema general del robot para el seguimiento de trayectorias con los subsistemas creados para ello, los cuales se detallarán a continuación.

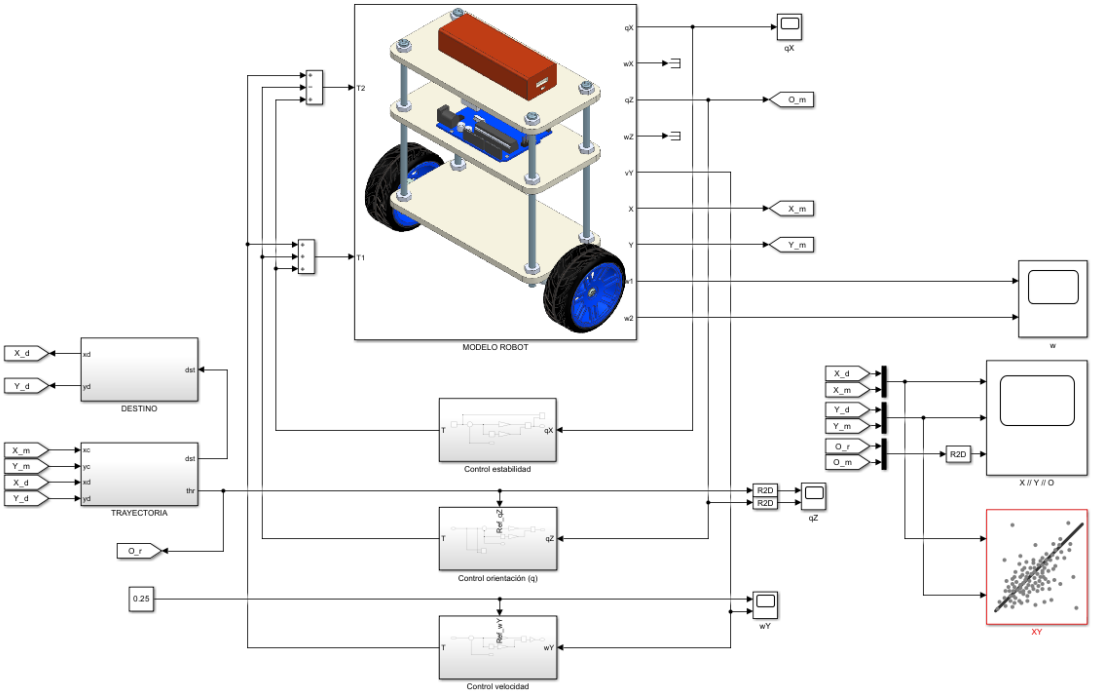


Figura 1.50: Modelo general para el control de trayectorias del robot.

El primer subsistema que se ha creado ha sido el que generará posición de destino del robot (ver Figura 1.51), es decir, este subsistema tendrá como entrada las posiciones del robot (X_c ,Y_c) y las posiciones de destino del robot (X_d ,Y_d) para obtener como salida; por un lado el ángulo hacia el que el robot se orientará (que se introducirá como referencia del control de orientación), y por otro lado, la distancia hasta el siguiente punto, la cual será la entrada del subsistema que calculará la trayectoria (ver Figura 1.52).

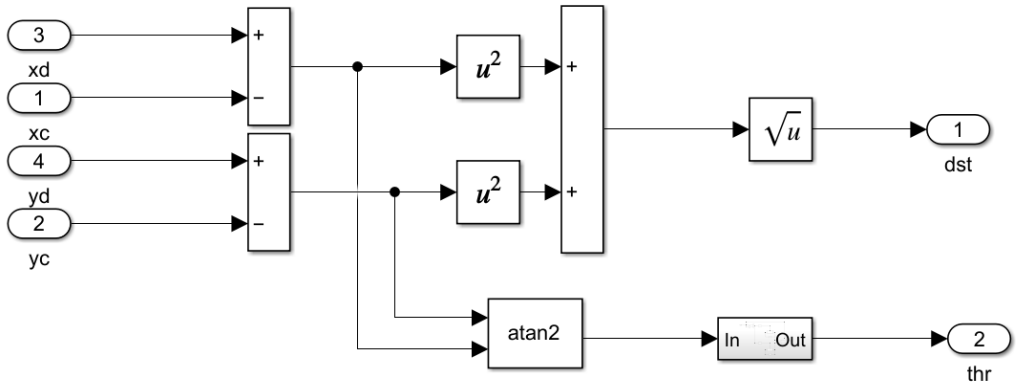


Figura 1.51: Subsistema para el cálculo de la posición de destino.

Las posiciones de destino (X_d ,Y_d) vendrán dadas por el subsistema que calcula la trayectoria (ver Figura 1.52), a partir de la distancia desde un punto al siguiente.

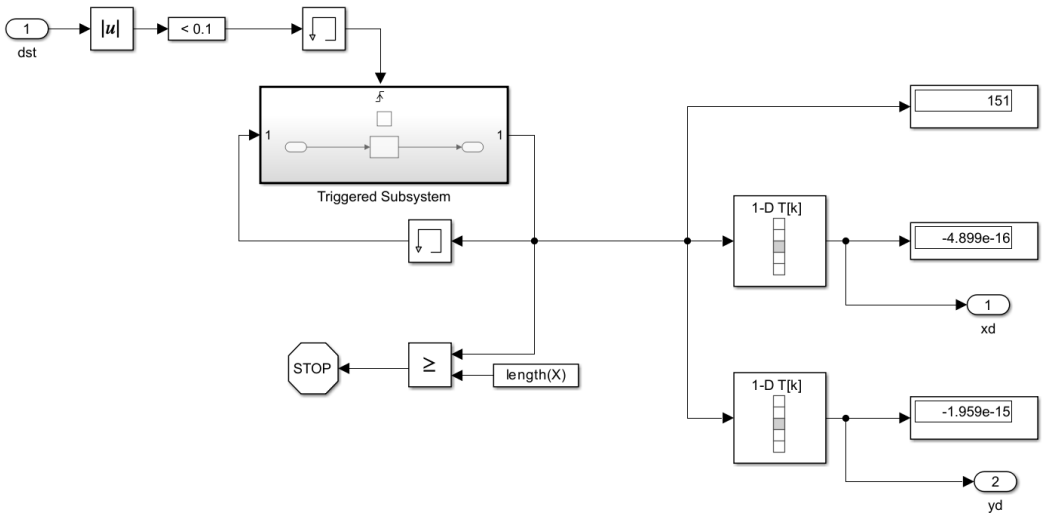


Figura 1.52: Subsistema para el cálculo de la trayectoria.

En la Figura 1.51, el cálculo de la posición objetivo se determina mediante la Ecuación (1.46). Esta ecuación involucra las coordenadas de destino (X_d, Y_d) y las coordenadas actuales o conocidas (X_c, Y_c), como se ha mencionado previamente. Este cálculo está condicionado por la posición actual del robot y su próximo destino. Por otro lado, la determinación del ángulo al que el robot se orientará viene dada por la Ecuación (1.47).

$$d = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2} \tag{1.46}$$

$$\theta = \arctan\left(\frac{y_d - y_c}{x_d - x_c}\right) \tag{1.47}$$

Posteriormente, en la Figura 1.51, en el bloque para el cálculo de la arco-tangente, de donde se obtiene el ángulo de orientación thr, se ha implementado un subsistema cuyo objetivo es que el robot pueda ajustar su orientación de forma correcta, para así evitar hacer giros que no serían correctos y realizarlos en la dirección más corta hacia su posición de destino. Esto se realiza introduciendo el valor del ángulo de giro en la entrada al subsistema, el cual comprueba si este es menor que $-\pi$ o mayor que π, sumándoles o restándoles 2π, respectivamente. De esta forma, el ángulo obtenido estará entre $[-\pi, \pi]$, realizando así siempre el giro más corto.

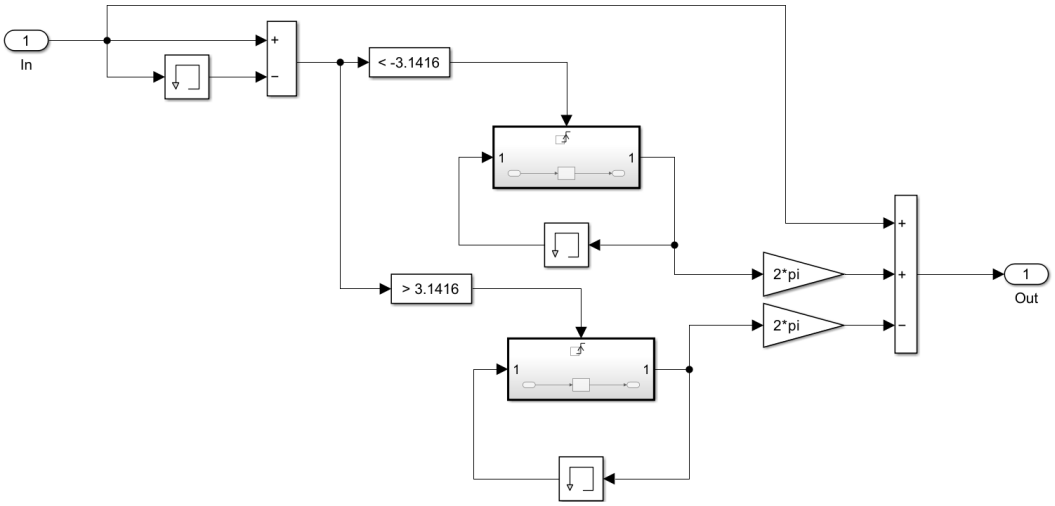


Figura 1.53: Subsistema para normalizar el valor del ángulo de giro.

Por otro lado, en la Figura 1.52, se emplea como entrada la distancia calculada anteriormente, la cual se compara con un umbral de 0,1. Si esta distancia es menor al umbral, se activa una señal de disparo o *trigger*, que a su vez genera una nueva ubicación incrementando un contador. Este contador actualizado se utiliza para acceder a los vectores de coordenadas xy y así generar una nueva posición. Estas nuevas ubicaciones son esenciales para alimentar al bloque que calcula la siguiente posición deseada, como

se muestra en la Figura 1.51. Una vez que se han recorrido todos los valores del vector, se establece una condición de parada (STOP) para que el robot siga esa trayectoria una sola vez.

Cuando se definan todos los subsistemas y se hayan realizado las conexiones necesarias entre ellos (ver Figura 1.50) se procederá a implementar el código para la programación de las trayectorias. Para ello, se utilizarán las **curvas de Lissajous** (ver Figura 1.58)[24], las cuales son el resultado del movimiento de dos osciladores armónicos simples cuyas direcciones son perpendiculares (Universidad Complutense de Madrid, 2023). Implementando el código en la función *InitFcn* del *Model Properties* de *Matlab*:

```
pp=5 ; aa=2 ;
X1=aa*sin(2*pi*0.1*[0:1/pp:10]) ;
Y1=aa*sin(2*pi*0.2*[0:1/pp:10]) ;

pp=10 ; aa=2 ;
X2=aa*sin(2*pi*0.1*[0:1/pp:10]) ;
Y2=aa*sin(2*pi*0.4*[0:1/pp:10]) ;

X=[X1 X2(2:end)] ; Y=[Y1 Y2(2:end)] ;
```

Listing 1.1: Código para la implementación del algoritmo de las Curvas de Lissajous.

Considerando que *pp* representa la cantidad de puntos en cada uno de los períodos de la onda senoidal y *aa* es la amplitud de dicha onda, al combinar dos curvas se determina la trayectoria del robot. Como se puede observar en la figura 1.54, se muestra una vista cenital del plano xy donde el robot sigue en gran medida la trayectoria definida.

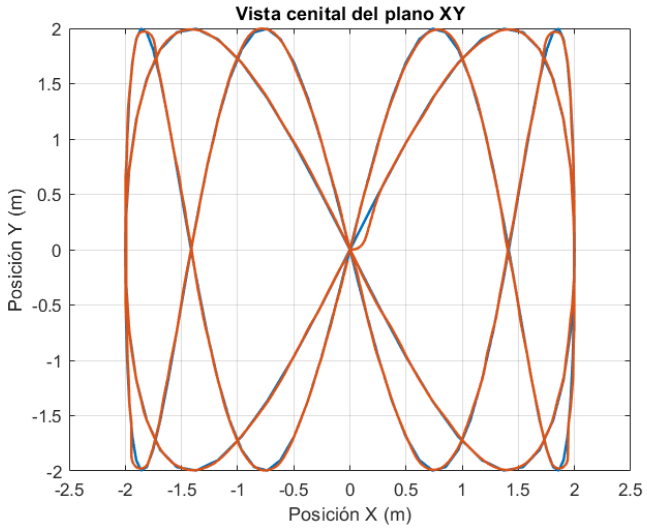


Figura 1.54: Vista cenital del plano xy para la visualización de la trayectoria realizada

[24]Fuente: `https://es.wikipedia.org/wiki/Archivo:Lissajous_relaciones.png`

En la Figura 1.55 se muestra la posición en los ejes xy respecto del tiempo, donde se observa que el robot va siguiendo la referencia de los vectores de puntos introducidos.

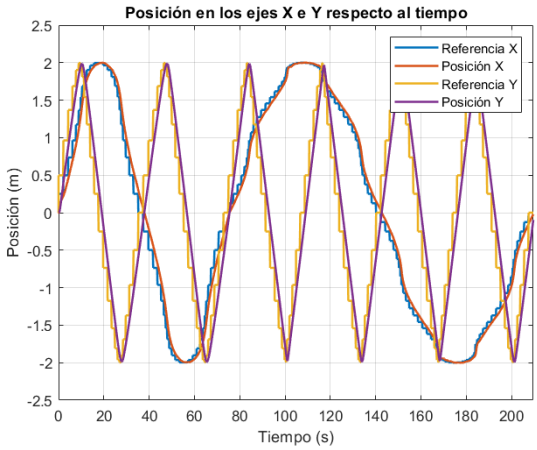


Figura 1.55: Gráfica de la posición xy del robot respecto al tiempo.

Por último, en la Figura 1.56 se muestran dos aspectos importantes: por un lado, se ha representado el control de orientación o giro, junto con la referencia a seguir; mientras que por otro lado, se muestra la velocidad lineal también junto a la referencia. A excepción del inicio, donde el robot es soltado y experimenta un pico inicial, durante todo el trayecto la velocidad lineal se mantiene estable y sigue la referencia de velocidad establecida inicialmente.

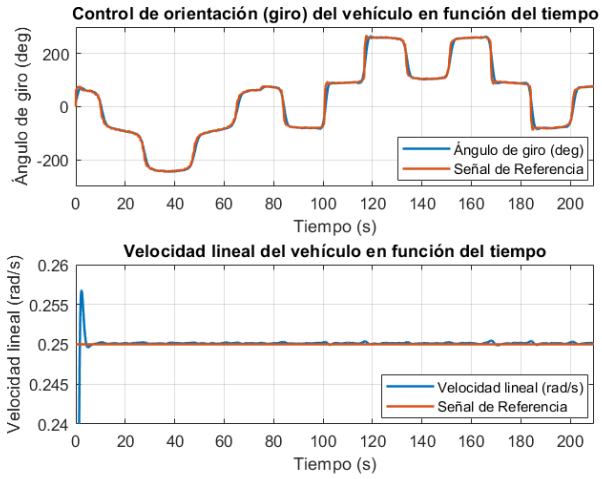


Figura 1.56: Gráfica del control de la velocidad angular y lineal.

La simulación de la trayectoria visualizada anteriormente, queda definida en la Figura 1.57, donde se muestra un instante de tiempo de la simulación del control de trayectorias del robot autobalanceado.

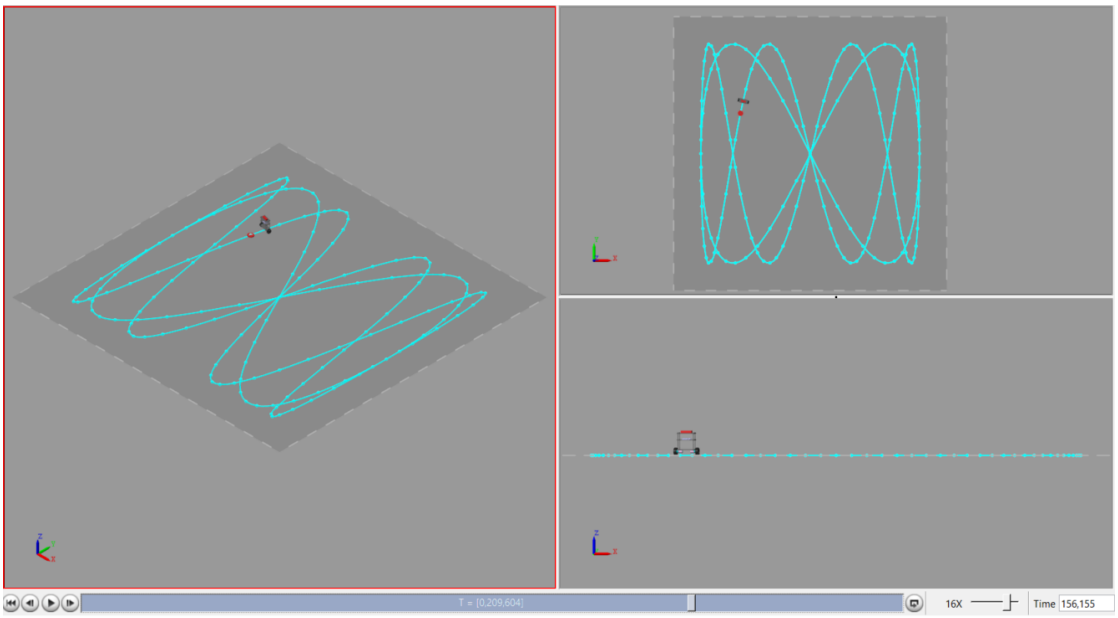


Figura 1.57: Visualización y simulación de la trayectoria implementada.

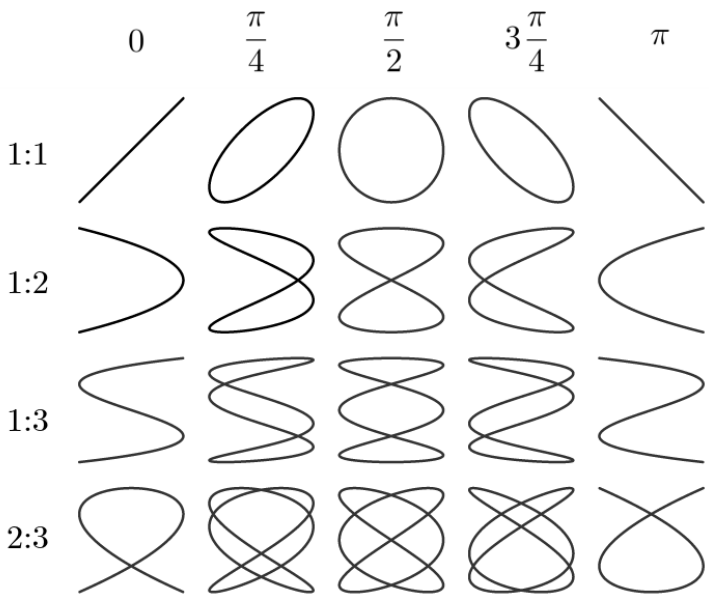
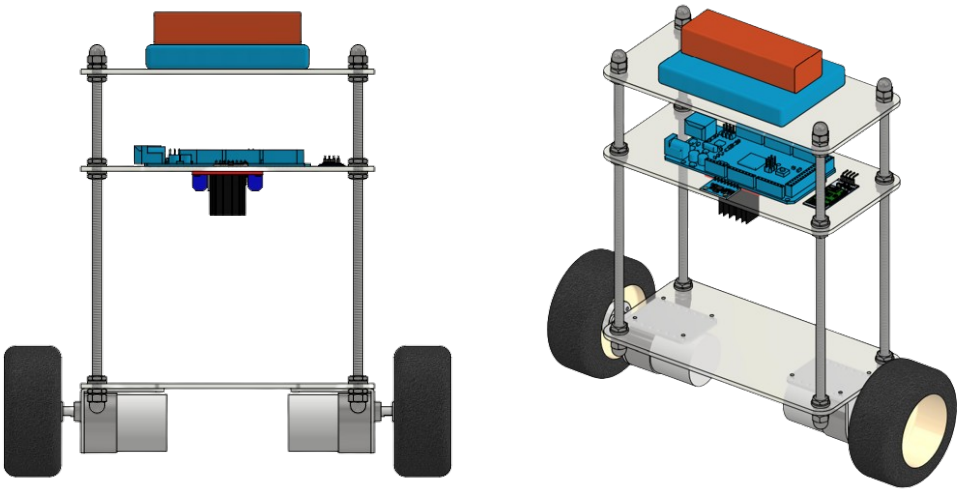


Figura 1.58: Algoritmo de las Curvas de Lissajous.

1.6. Implementación real del robot

Para realizar la implementación real del robot, una vez seleccionados los componentes (véase Apartado 1.4.2: Alternativas a los componentes utilizados y solución adoptada) se procede a realizar el montaje real del robot y la programación del microcontrolador, junto con la integración conjunta de los sensores y los motores. Para realizarlo correctamente, se ha tomado como base el modelo 3D del robot desarrollado con el *software Siemens NX*, realizando algunas modificaciones, ya que en *Simescape Multibody*, debido a los sensores proporcionados por los bloques, no ha sido necesario implementar los componentes en su totalidad. A continuación, en la Figura 1.59 se muestra el modelo del robot diseñado en el programa de CAD 3D *Inventor*, que se ha utilizado debido a su mejor compatibilidad con los modelos de los componentes. En esta representación, se pueden visualizar todos los componentes implementados.



(a) Vista frontal del robot (b) Vista en perspectiva del robot

Figura 1.59: Modelo 3D del robot diseñado en *Inventor* con los componentes electrónicos.

1.6.1. Montaje y funcionamiento de los componentes

En este apartado se detallará el principio de funcionamiento de los componentes del robot y los materiales utilizados para su implementación, con el fin de poder realizar posteriormente la programación del control de estabilidad y el control manual de las trayectorias. También se mostrarán esquemas de las conexiones de cada uno de los componentes con el fin de visualizar de una forma intuitiva y clara la implementación de todos ellos.

En la Figura 1.60 se muestra un esquema del sistema completo con los componentes que se implementarán. Este sistema contará con una primera etapa de sensado (GY-521), seguida de una etapa de control (Arduino Due), para finalizar con una etapa de potencia (*driver* y motores).

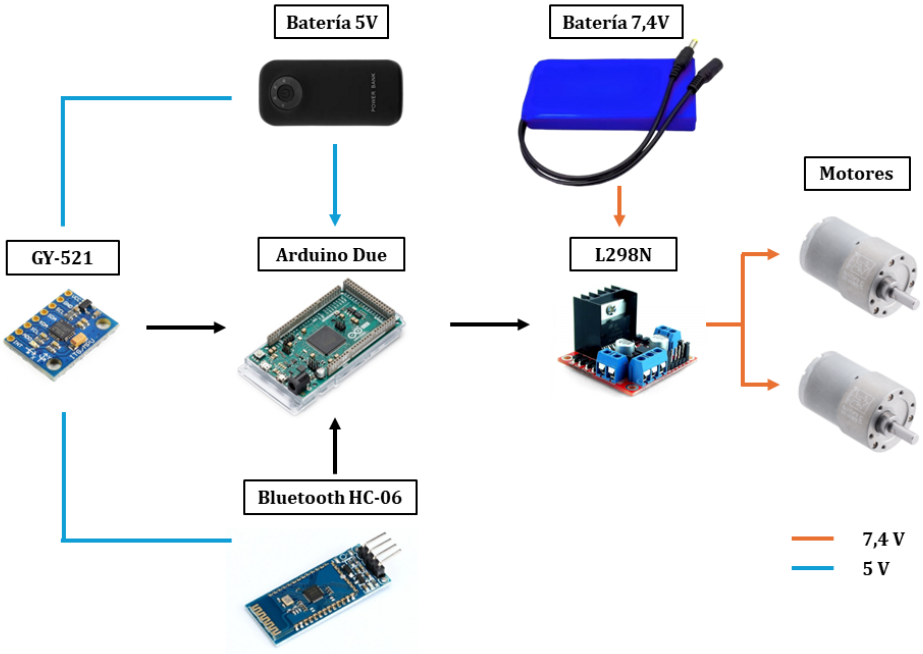


Figura 1.60: Esquema montaje y alimentaciones componentes.

Sensor GY-521

Como se ha mencionado anteriormente, (véase Apartado 1.4.2: Alternativas a los componentes utilizados y solución adoptada) el sensor que se utilizará en este proyecto será el Módulo GY-521 Acelerómetro y Giroscopio MPU-6050. Este sensor de 6 grados de libertad, incluye un acelerómetro y un giroscopio MEMS, ambos integrados en un chip. El sensor trabaja a una tensión de 3.3 V, pero gracias al regulador que este lleva integrado, se puede alimentar desde el pin de 5V del microcontrolador.

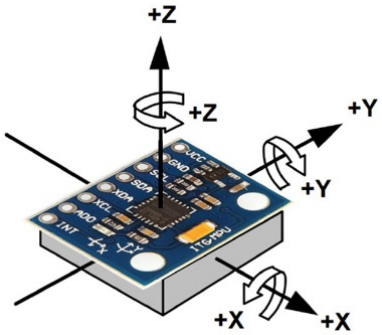


Figura 1.61: Ejes sensor MPU-6050.

Mediante este sensor será posible obtener el ángulo de inclinación del vehículo (*roll*) tomando como referencia su vertical. Por lo tanto, a pesar de los 6 grados de libertad que proporciona el sensor, en este proyecto solamente será necesario el grado de libertad que proporciona el eje x del giroscopio. (ver Figura 1.61)[25]

En cuanto a las conexiones de este sensor con el microcontrolador, se debe considerar que la comunicación del módulo es I2C. Este hecho facilita la compatibilidad con distintos modelos de microcontroladores. Por defecto, el pin AD0 del módulo está conectado a masa, por lo que la dirección I2C será 0x68. Si se hubiera conectado este pin a VCC, la dirección I2C sería 0x69 (Naylamp Mechatronics, 2024b). En la Figura 1.62 se puede visualizar el esquema de conexiones del sensor con el microcontrolador Arduino Due.

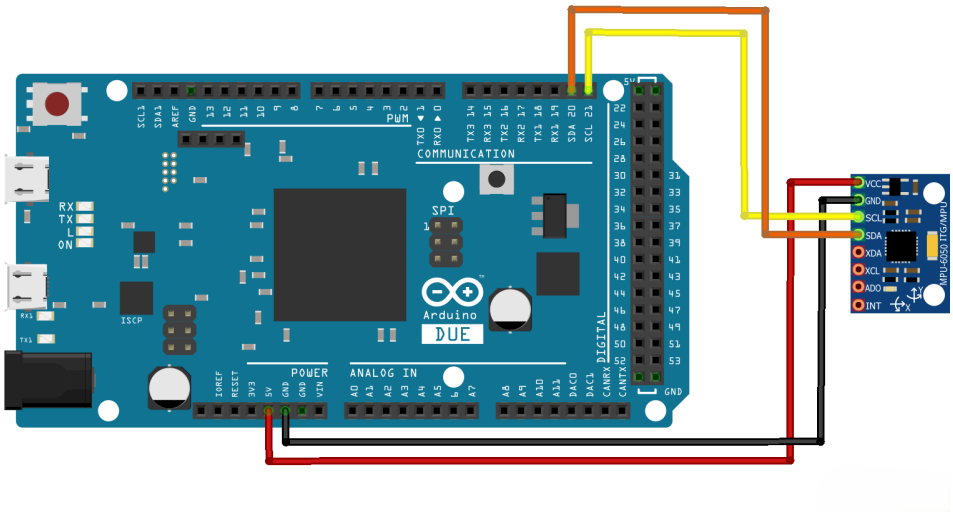


Figura 1.62: Conexiones sensor MPU-6050.

[25]Fuente: `https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html`

Según el esquema de conexiones anterior, al utilizar la dirección I2C 0x68, desde la IMU se conextará VCC con la salida de 5V del Arduino. Las masas GND se conectarán entre sí, y finalmente, los pines SDA y SCL del sensor se conectarán con los pines 20 (SDA) y 21 del Arduino (SCL), respectivamente. En el caso de este proyecto, el pin INT (interrupción) no se conectará ya que no será necesario el uso de esta herramienta.

Motores CC y driver L298N

Como se ha mencionado anteriormente (véase Apartado 1.4.2: Alternativas a los componentes utilizados y solución adoptada) los motores que se utilizarán serán motorreductores 30:1 Metal Gearmotor modelo 37Dx52L mm a 12 V. Ambos motores irán conectados al microcontrolador mediante el driver L298N. Este componente estará alimentado por una batería externa de 7,4 V como se detalla en la Figura 1.63. El driver tiene dos salidas PWM, las cuales se conectarán a los pines 8 y 10 del Arduino Due, respectivamente. Los 4 puertos para las conexiones de los motores irán conectados a las salidas digitales del microcontrolador.

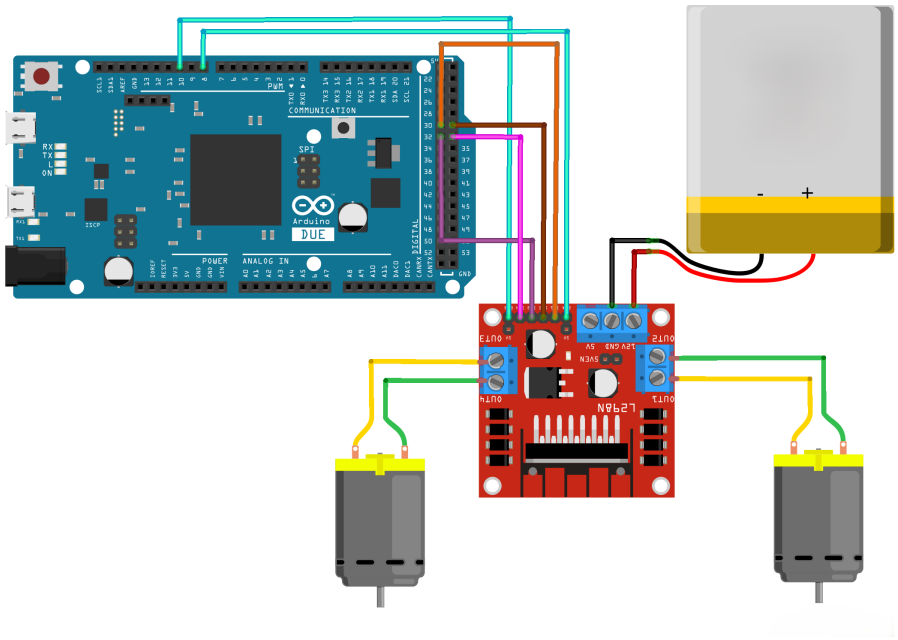


Figura 1.63: Conexiones de los motores y el driver al microcontrolador.

Módulo bluetooth HC-06

A través de una aplicación móvil se enviarán datos y comandos para controlar el robot de forma remota. Este componente se alimenta mediante el puerto de 3,3V de la placa Arduino Due y posee dos pines TXD y RXD, los cuales se conectarán a los pines RXD y TXD del microcontrolador, respectivamente. Es importante tener en cuenta que

el pin TXD del módulo debe conectarse al pin RXD del microcontrolador, y viceversa, tal y como queda reflejado en la Figura 1.64.

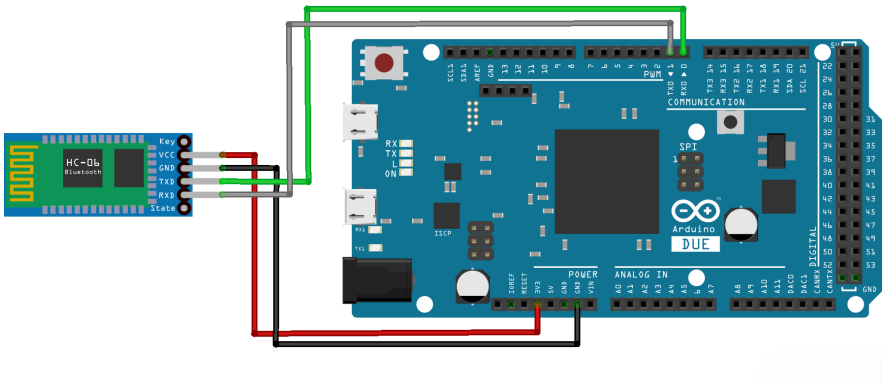
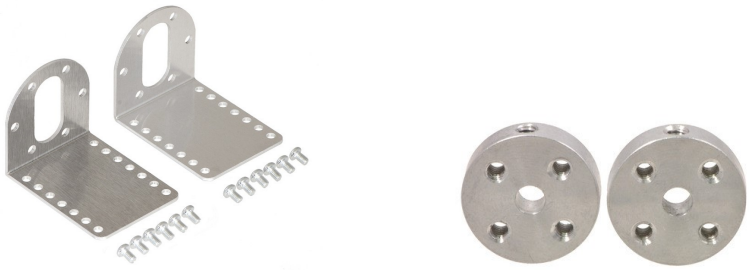


Figura 1.64: Conexiones del módulo bluetooth HC-06 al microcontrolador.

Otros materiales utilizados para el montaje

Para la mecanización de las bandejas diseñadas para el montaje del robot, se han optado por utilizar placas de metacrilato de 3mm de grosor (véase Apartado 1.4.1: Alternativas al proyecto y solución adoptada). Para el montaje y el ensamblado de estas placas se han utilizado varillas roscadas, tuercas y arandelas, todas ellas de métrica M6.

Para el montaje de las ruedas se han utilizado *brackets* en L de Polulu (ver Figura 1.65(a))[26], que se encargarán de unir los motores a los cuales irán atornillados, al chasis del robot (Pololu, 2024c). Para la unión de las ruedas con los motores, se han utilizado *hubs* de la marca Polulu (ver Figura 1.65(b))[27]. Estos permitirán unir el eje del motor con la rueda, a la que irán atornillados (Pololu, 2024a). Este montaje queda detallado en el CAPÍTULO 2: Planos.



(a) *Brackets* aluminio Polulu (b) *Hubs* aluminio Polulu M3

Figura 1.65: Materiales para el montaje de las ruedas y los motores.

[26]Fuente:`https://www.pololu.com/product/1084`
[27]Fuente:`https://www.pololu.com/category/137/pololu-universal-mounting-hubs`

1.6.2. Programación del robot

En este apartado se detalla la programación realizada mediante el *software* Arduino IDE para el control de la estabilidad y el manejo manual del vehículo autobalanceado. Se comenzará presentando un diagrama de flujo del código general (ver Figura 1.66), lo cual servirá como base para la realización del código correspondiente.

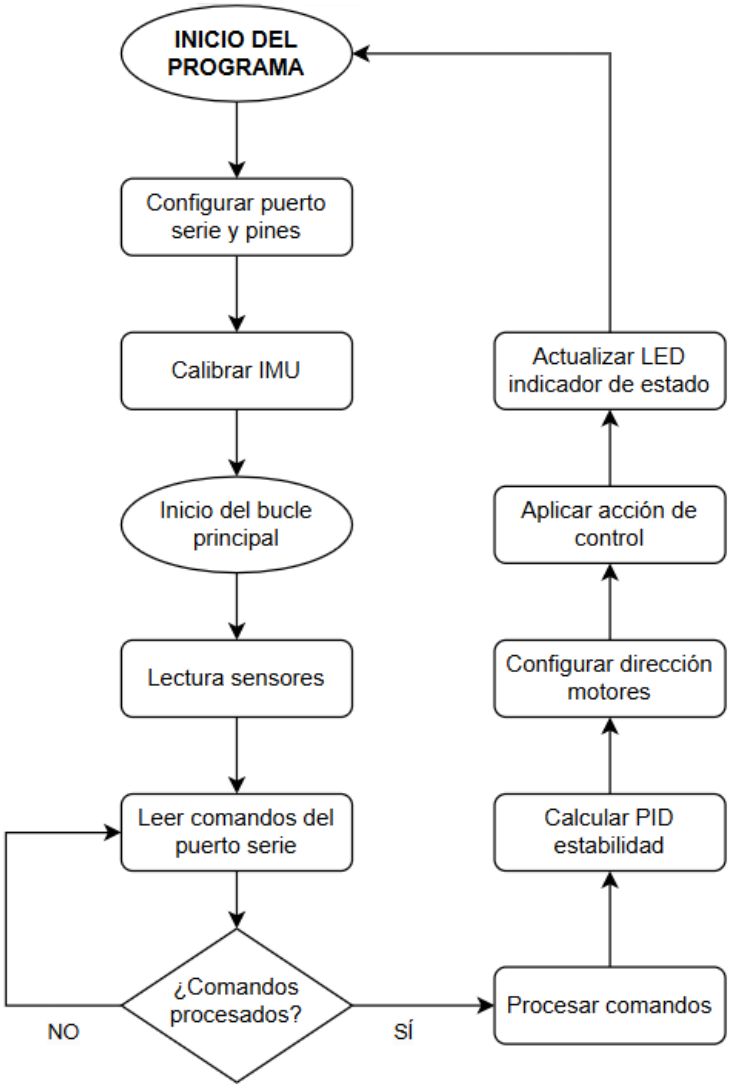


Figura 1.66: Diagrama de flujo para la programación del robot.

Para comenzar con la implementación del código, este se realizará en dos partes. En primer lugar, se realizará el control de la estabilidad, ajustando los valores de las ganancias del controlador de forma que el robot se mantenga lo más estable posible. Una vez implementada esta parte del código, se continuará con el control manual, mediante el uso de una aplicación que permitirá realizar el control del robot mediante dos *joysticks* que el usuario controlará.

Control de la estabilidad

Para realizar el control de estabilidad, lo primero será realizar una calibración inicial de la MPU 6050. Para ello, se posicionará el robot lo más vertical posible y se registrarán 1000 muestras. Se calculará la media de los últimos 800 valores para mayor precisión, dado que aproximadamente los primeros 100 valores de la IMU son de inicialización y no son correctos. Esta media obtenida se considerará como un valor relativo de cero, es decir, se tomará como referencia para establecer el ángulo de inclinación en ese punto.

Para llevar a cabo este proceso, en primer lugar se declararán las variables y se realizará la configuración inicial, que será idéntica a la utilizada para el control manual (véase ANEXO B: Código Arduino para la programación del robot). La inicialización consistirá en completar la función '**'setup()**', que se ejecutará una sola vez al inicio del programa. En esta configuración inicial, se inicializará la comunicación Serial a 115200 baudios. Además, se configurarán el LED y los pines del *driver* conectados a los motores como salidas. Por último, se llevará a cabo una configuración inicial del acelerómetro, seguida de una llamada a la función '**Calibrado()**'.

Después de completar todas las inicializaciones necesarias, se procederá a desarrollar la función '**'Muestra()**', que leerá los valores proporcionados por el acelerómetro y calculará el ángulo de inclinación (ver Ecuación 1.48) en grados. Posteriormente, se iniciará una nueva transmisión para leer los valores del giroscopio y calcular el ángulo correspondiente. Para obtener lecturas precisas, el filtro complementario combinará las lecturas de ambos sensores (ver Ecuación 1.49). Para establecer el valor relativo de cero en la vertical, se ajustará este ángulo restando el valor inicial y, finalmente, se actualizará el valor del ángulo para la próxima iteración.

$$\text{Acc} = \tan^{-1}\left(-\frac{\text{AcX}}{A_R}\Big/\sqrt{\left(\frac{\text{AcY}}{A_R}\right)^2+\left(\frac{\text{AcZ}}{A_R}\right)^2}\right)\times \text{RAD\_TO\_DEG} \tag{1.48}$$

$$\theta = A\cdot(\theta_{\text{prev}}+\omega_{\text{gyro}}\cdot\Delta t)+B\cdot\theta_{\text{accel}} \tag{1.49}$$

A continuación se muestra el código correspondiente de la función **'Muestra()'**:

```
void Muestra() {
  int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;
  float Acc, Gy;

  // Leer los valores del Acelerometro de la IMU
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  AcX = Wire.read() << 8 | Wire.read();
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();

  Acc = atan(-1 * (AcX / ACCEL_RATIO) / sqrt(pow((AcY / ACCEL_RATIO), 2)
    + pow((AcZ / ACCEL_RATIO), 2))) * RAD_TO_DEG;

  // Leer valores IMU
  Wire.beginTransmission(MPU);
  Wire.write(0x43);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 4, true);

  GyX = Wire.read() << 8 | Wire.read();
  GyY = Wire.read() << 8 | Wire.read();

  // Calcular angulo IMU
  Gy = GyY / GIRO_RATIO;

  // Filtro complementario
  rollAngle = (0.98 * (angR_prev + Gy * T_muest_seg) + 0.02 * Acc) ;
  rollAngle = rollAngle - initialRoll ;
  angR_prev = rollAngle + initialRoll;
}
```

Listing 1.2: Función **Muestra()**.

Por otro lado, se realizará la función para obtener los 1000 valores de calibración, como se ha mencionado anteriormente. En la función '**Calibrado()**', se activa el LED de estado para verificar que la calibración se está realizando correctamente. Luego, se inicializan las variables y se inicia un bucle que recogerá 1000 valores llamando a la función '**Muestra()**', imprimiéndolos por pantalla. Finalmente, se hace una media de los 800 últimos valores.

A continuación se muestra el código correspondiente de la función Calibrado():

```
void Calibrado() {
  digitalWrite(LED, HIGH);
  Serial.print("Valores iniciales");
  float angleSum = 0.0;
  int cc = 0;
  for (int ii = 1; ii <= 1000; ii++) {
    Muestra();

    if(ii>=200) {
      angleSum += rollAngle;
      delay(10);
      Serial.print(rollAngle) ; Serial.print("\t") ; Serial.println(ii) ;
      if (cc>=100) {Serial.print(".") ;
      digitalWrite(LED,HIGH) ; delay(50) ; digitalWrite(LED,LOW) ; delay
    (50) ;
      cc=0 ;} else {cc++ ;}
    }
  }

  initialRoll = angleSum / 800.0;
  Serial.print("Alabeo inicial: "); Serial.println(initialRoll);
  delay(1000) ;
}
```

Listing 1.3: Función **Calibrado()**.

Una vez completado todo lo anterior, el siguiente paso será desarrollar la función '**loop()**', la cual se ejecutará de manera continua mientras el programa esté en funcionamiento. En primer lugar, se llamará a la función '**Muestra()**' para obtener los datos correspondientes de la IMU. Una vez obtenidos estos valores, se implementará el controlador PD para la estabilidad, definiendo los valores de las ganancias K_p y K_d.

Para ajustar apropiadamente ambas ganancias, es esencial comprender el papel de cada una. La ganancia proporcional K_p determinará la rapidez con la que el robot responde a la desviación de la referencia. Es decir, cuando el robot se incline hacia un lado, deberá corregir ese movimiento de manera eficiente. Por otro lado, la ganancia derivativa K_d será responsable de suavizar el movimiento para lograr alcanzar la referencia de manera precisa, minimizando la sobresocilación.

A continuación, se ajustará la dirección de los motores, que en este caso será la misma para ambos, ya que el objetivo es lograr la estabilidad del robot. Esto significa que si el robot se inclina hacia un lado, ambas ruedas se moverán en la misma dirección que indica la inclinación del robot. Finalmente, se limitará el valor de la acción de control dentro de unos umbrales apropiados para garantizar un sistema estable y se obtendrán valores adecuados para controlar la velocidad del robot.

A continuación se muestra el código correspondiente de la función **'loop()'** para la estabilidad:

```
void loop() {

  Muestra();

    // PD ESTABILIDAD
    K_prop = 5.05;
    K_der = 0.22;

    AngRef = 0;
    Error = AngRef - rollAngle;
    Der_error = rollAngle / T_muest_seg;
    acc = K_prop * Error + K_der * Der_error;


  // DIRECCION DE LOS MOTORES
  if (acc > 0.00) {
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
  }
  else if (acc < 0.00) {
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
  }

  // ACCION DE CONTROL DE LOS MOTORES
  if(acc >= MAX_U) {acc = MAX_ACC;}
  if(acc <= MIN_U) {acc = MIN_ACC;}

  acc_1 = map(abs(acc) * 10, 0, MAX_ACC * 10, 700, 4095);
  acc_2 = acc_1;

  analogWrite(PWM1, acc_1);
  analogWrite(PWM2, acc_2);

  Serial.print("rollAngle=") ; Serial.print(rollAngle) ; Serial.print("\t
    ") ;
  Serial.print("acc_1=") ; Serial.print(acc_1) ; Serial.print("\t") ;
  Serial.print("acc_2=") ; Serial.print(acc_2) ; Serial.println("\t") ;

  // FUNCIONAMIENTO LED
  time2=millis() ;
  if (time2-time1>T_muest) {digitalWrite(LED,HIGH) ;}
  else {digitalWrite(LED,LOW) ;}
  while (time2-time1<T_muest) {time2=millis() ;}
  time1=millis() ;
}
```

Listing 1.4: Función **loop()** para el control de estabilidad

Una vez que el código anterior haya sido implementado en el entorno de desarrollo Arduino IDE y los valores de las ganancias sean ajustados correctamente, el robot será capaz de mantener su estabilidad en gran medida, incluso frente a pequeñas perturbaciones. Sin embargo, en este punto es importante tener en cuenta que el robot experimentará pequeñas derivas de movimiento, ya que alcanzar una estabilidad absoluta sería muy complicado y requeriría una calibración extremadamente precisa. Estas pequeñas derivas pueden corregirse mediante el control manual. En la Figura 1.67 se muestra una gráfica que permite visualizar la estabilidad del robot, donde se puede observar que sigue la referencia en gran medida.

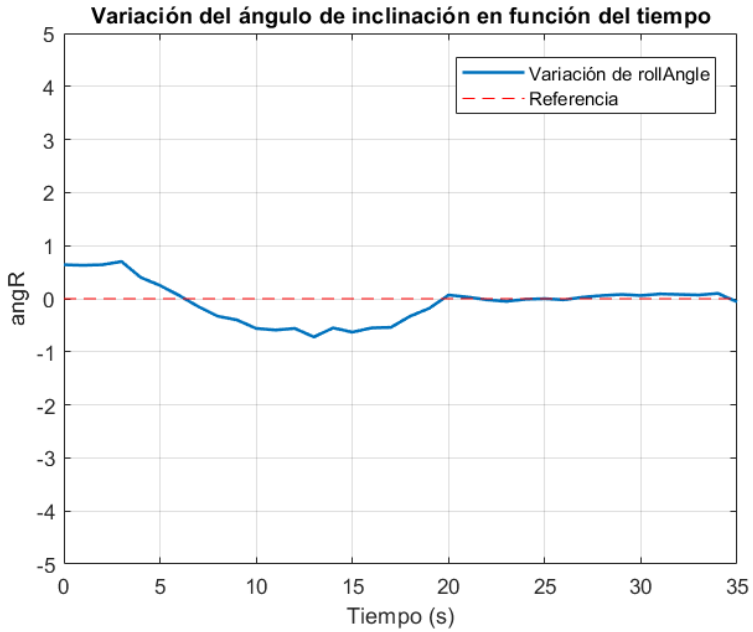


Figura 1.67: Gráfica para visualizar el ángulo de inclinación y la estabilidad del robot.

Control manual

En este apartado, se llevará a cabo el control manual del robot, permitiendo al usuario controlarlo mediante dos *joysticks* a través de una aplicación móvil. Para este propósito, se utilizará la aplicación '**Bluetooth Electronics**', que se comunicará con el módulo HC-06. Se creará una interfaz con un *joystick* para controlar los movimientos hacia delante/atrás y otro para realizar giros en diferentes direcciones. La Figura 1.68 muestra la interfaz (UI) creada en la aplicación.

Ambos *joysticks* se han configurado con un valor mínimo de 10 y un valor máximo de 99, y enviarán los datos en formato de coordenadas xy. El *joystick* para el avance/retroceso comenzará con una 'V' y el de giro con una 'D'. Por lo tanto, los datos procedentes de la aplicación se recibirán en forma de $VXddYdd$ y $DXddYdd$.

El siguiente código se encarga de extraer los datos enviados por la aplicación, de modo que se obtienen valores entre [-1,1] al mover el *joystick* de avance/retroceso hacia arriba o hacia abajo, y el de giro hacia los lados.

Figura 1.68: Interfaz aplicación *Bluetooth Electronics* para el control manual.

```
1
2   if (Serial1.available()>=7) {
3     for (int ii=1 ; ii<=7 ; ii++) {
4       inByte[ii]=Serial1.read() ;
5     }
6
7     if (inByte[1]=='V') {
8       if (inByte[2]=='X' && inByte[5]=='Y') {
9         vx=(inByte[3]-48)*10+(inByte[4]-48) ;
10        vxf=map(vx,10,99,-100,100)/100.0 ;
11        vy=(inByte[6]-48)*10+(inByte[7]-48) ;
12        vyf=map(vy,10,99,-100,100)/100.0 ;
13        vel=-1*vyf ;
14      }
15    }
16
17    if (inByte[1]=='D') {
18      if (inByte[2]=='X' && inByte[5]=='Y') {
19        dx=(inByte[3]-48)*10+(inByte[4]-48) ;
20        dxf=map(dx,10,99,-100,100)/100.0 ;
21        dir=dxf ;
22        dy=(inByte[6]-48)*10+(inByte[7]-48) ;
23        dyf=map(dy,10,99,-100,100)/100.0 ;
24      }
25    }
26  }
```

Listing 1.5: Código para la extracción de los datos de la *app Bluetooth Electronics*.

Una vez obtenidos los valores de dirección y giro, a diferencia del control de estabilidad en el que la acción de control era igual para ambos motores, en el control manual la acción de control será la misma pero con el cambio de signo de la dirección, como se refleja en el código. Es decir, en el control manual las acciones de control serán diferentes para cada motor, lo que permite que el robot pueda girar. A continuación se presenta el código correspondiente a esta parte, con acciones de control diferenciadas para cada motor.

```

    // PD ESTABILIDAD
    K_prop = 5.05;
    K_der = 0.22;
    float Km = 1.0;
    float Kt = 6.0;

    AngRef = 0;
    Error = AngRef - rollAngle;
    Der_error = rollAngle / T_muest_seg;
    acc = K_prop * Error + K_der * Der_error;

    acc_1 = acc + Kt*vel + Km*dir ;
    acc_2 = acc + Kt*vel - Km*dir ;

  // DIRECCION MOTORES
  if (acc_1 > 0.00) {digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);}
  if (acc_2 > 0.00) {digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);}

  if (acc_1 < 0.00) {digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);}
  if (acc_2 < 0.00) {digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);}

  // ACCION DE CONTROL MOTORES
  if(acc_1 >= MAX_ACC) {acc_1 = MAX_ACC;}
  if(acc_2 >= MAX_ACC) {acc_2 = MAX_ACC;}
  if(acc_1 <= MIN_ACC) {acc_1 = MIN_ACC;}
  if(acc_2 <= MIN_ACC) {acc_2 = MIN_ACC;}

  analogWrite(EN_A,map(abs(acc_1)*10,0,MAX_ACC*10,700,4095));
  analogWrite(EN_B,map(abs(acc_2)*10,0,MAX_ACC*10,700,4095));

  Serial.print("rollAngle=") ; Serial.print(rollAngle) ; Serial.print("\t
    ") ;
  Serial.print("acc_1=") ; Serial.print(acc_1) ; Serial.print("\t") ;
  Serial.print("acc_2=") ; Serial.print(acc_2) ; Serial.println("\t") ;
```

Listing 1.6: Código de la acción de control de los motores y el controlador.

El código completo para realizar el control manual del robot se encuentra en el ANEXO B: Código Arduino para la programación del robot

1.7. Análisis de resultados

En esta sección se comentarán y compararán los resultados obtenidos durante las distintas simulaciones realizadas, además de analizar los resultados obtenidos mediante la implementación real del vehículo autobalanceado.

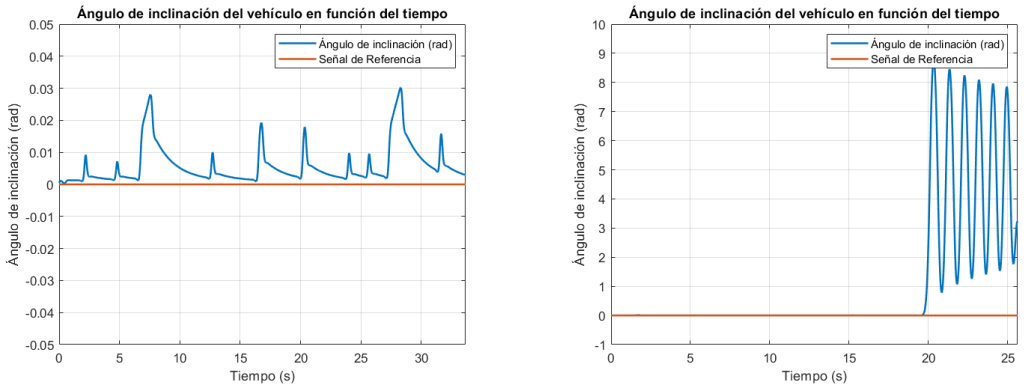
1.7.1. Resultados y pruebas de la simulación

Para comprobar distintos tipos de funcionamiento de este tipo de robots, y de menor a mayor dificultad de control, se han realizado diversas simulaciones para evaluar diferentes aplicaciones. La simulación del control de la estabilidad se ha llevado a cabo al principio, ya que, como se ha mencionado durante el trabajo, este es el objetivo principal del robot. Esta ha servido como base para las posteriores, ya que el resto, que requerían la implementación de otros elementos de control, están basados en esta simulación inicial, realizando modificaciones sobre ella.

Durante la simulación del **control de estabilidad** (véase 1.5.2: Control de estabilidad) se han podido comprobar algunos de los principios matemáticos que rigen este modelo (véase 1.2.4: Modelo cinemático del robot) y se ha comprobado, que realizando un ajuste correcto de los parámetros internos del modelo y de las ganancias que componen el controlador PD de estabilidad, se puede realizar un control preciso ante perturbaciones.

Siguiendo con la simulación, la segunda que se ha llevado a cabo ha sido el **control manual** del robot, en el que un usuario, manualmente, puede dirigir los movimientos en el plano del robot, así como realizar diferentes acciones con los botones del *gamepad* (véase 1.5.3: Control manual del robot). Esta simulación ha servido para comprobar diferentes utilidades del robot, aunque no es muy precisa debido a que los *joysticks* no introducen un valor digital exacto, a diferencia de los botones, lo que causa que en determinados movimientos existan derivas o que las ruedas adquieran demasiada acción de control, por lo que el robot aceleraría demasiado hasta llegar al punto de poder perder la estabilidad.

Para comprobar la versatilidad del robot y su buen funcionamiento en las simulaciones mencionadas anteriormente, se han realizado una serie de pruebas. En la Figura 1.69(a) se muestra la primera prueba. En esta, se han ido introduciendo diversas perturbaciones mediante el botón 'A' del *gamepad* a lo largo de un intervalo de tiempo, lo que demuestra que el robot es resistente a ellas y se mantiene vertical en todo momento. Enlazado a esto, en la Figura 1.69(b) se demuestra que el robot es completamente estable hasta que el usuario acciona el 'Botón de emergencia', donde el robot pierde completamente la estabilidad y adquiere valores para el ángulo de inclinación que no son correctos.



(a) Gráfica del ángulo de inclinación ante perturbaciones externas

(b) Gráfica del ángulo de inclinación hasta que el robot se desestabiliza

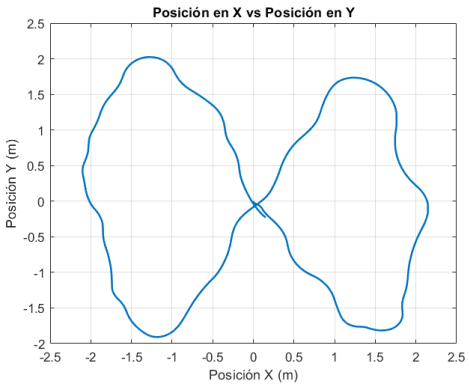
Figura 1.69: Pruebas realizadas para la estabilidad y el control manual.

Por otro lado, se ha llevado a cabo la simulación del **control automático** para la velocidad y el giro del robot (véase 1.5.4: Control automático de la velocidad lineal y angular). Como se detalla en el apartado correspondiente, esto implica que el robot puede seguir una secuencia de velocidad y giro introducida manualmente para trazar una serie de movimientos en el plano. Durante este proceso, también se ha evaluado el funcionamiento de la simulación con un *joystick*, observando que el robot adquiere mayor o menor velocidad lineal o angular según el valor introducido por el usuario. Esto contrasta con el **control manual**, que mantiene una velocidad constante en todo momento. Mediante esta simulación, se han logrado realizar movimientos similares a las trayectorias, aunque con ciertas diferencias, dado que los motores reciben órdenes de velocidad y giro en lugar de puntos de origen y destino.

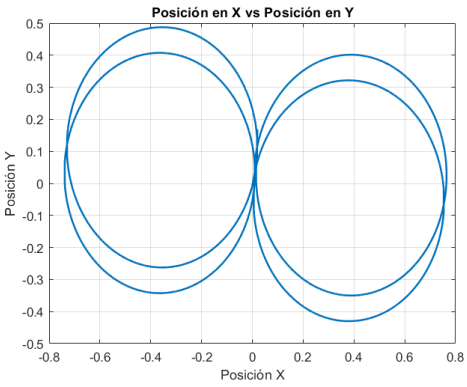
Por último, se ha realizado la simulación del **control de las trayectorias** del robot, implementando el algoritmo definido por las **Curvas de Lissajous** (véase 1.5.2: Control de trayectorias). Estas curvas ofrecen una gran versatilidad para generar trayectorias, ya que al cambiar dos parámetros de las ecuaciones generales, el robot realizará diferentes movimientos con distintas amplitudes de la onda senoidal. Esta simulación es la más precisa, ya que, a diferencia del control automático, donde los movimientos están determinados por referencias de velocidad y giro, en este caso, están basados en puntos de origen y destino. Por lo tanto, para completar la trayectoria final, el robot debe pasar por cada uno de los puntos definidos, lo que garantiza un movimiento muy preciso.

Como comparación entre las distintas simulaciones, se ha intentado realizar una trayectoria similar en todas para verificar visualmente las diferencias. La prueba consistió en que el robot realizara el símbolo de infinito. En la Figura 1.70(a), se observa que el control manual no es exacto para seguir trayectorias, ya que el usuario controla al robot mediante un *gamepad*. Por otro lado, en la Figura 1.70(b), se aprecia que el control

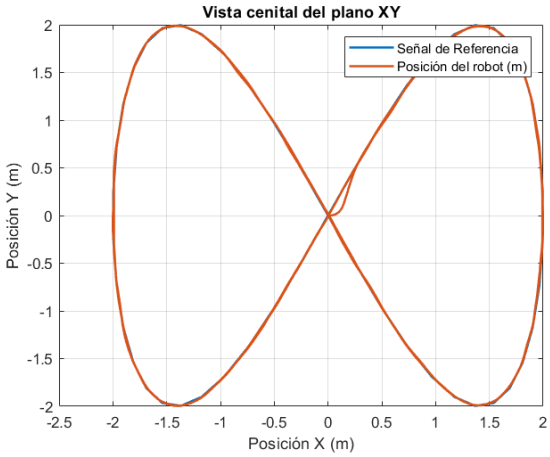
automático no es preciso para seguir trayectorias, ya que, para las mismas referencias de velocidad y giro, el robot no traza la misma secuencia. En contraste, en la Figura 1.70(c), se muestra que el control de trayectorias es preciso y sigue perfectamente la referencia de los puntos de destino marcados. En estos dos últimos casos, la secuencia se ha realizado dos veces, demostrando que el control automático no es exacto para realizar secuencias.



(a) Secuencia seguida por el robot en el control manual



(b) Secuencia seguida por el robot en el control automático



(c) Trayectoria realizada por el robot

Figura 1.70: Pruebas realizadas para los diferentes tipos de controles.

En cuanto a las simulaciones, el margen de mejora es muy amplio. Una vez obtenido el modelo dinámico del robot, se trata de realizar algunas variaciones sobre este modelo para que adquiera diferentes funcionalidades. En este proyecto, se han seleccionado simulaciones que permiten comprobar el comportamiento del sistema ante diferentes situaciones, pero hay diversas posibilidades de ampliaciones.

1.7.2. Resultados de la implementación real

En la implementación real del vehículo autobalanceado se ha optado por realizar un control de estabilidad seguido de un control manual, ya que a diferencia de las simulaciones, este tipo de vehículos tiene diversas limitaciones en el mundo real que hacen que sus aplicaciones sean más limitadas que en otro tipo de robots que no son inestables.

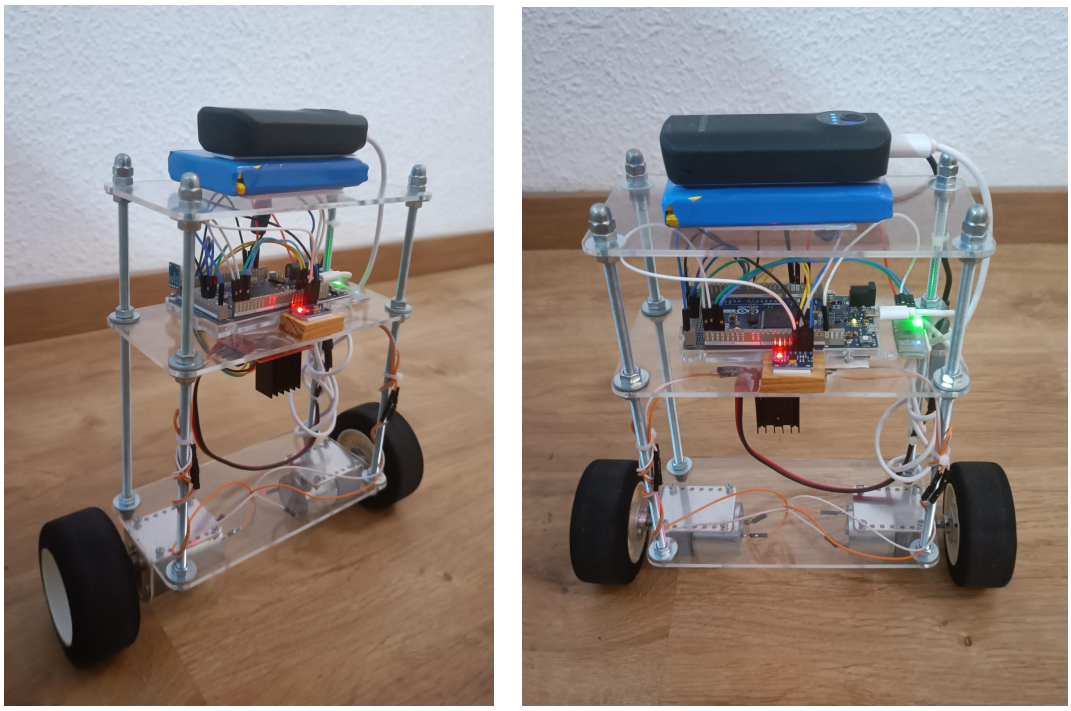
Para realizar esta implementación, primero se ha llevado a cabo un control de la estabilidad, ajustando bien las ganancias, de forma que el control manual sea más sencillo de implementar, al igual que se ha realizado durante las simulaciones. Previo a esto, se ha realizado el montaje de la estructura del robot y del circuito electrónico que lo compone. Antes de comenzar con el control de estabilidad, se realizaron pruebas del funcionamiento de los motores y de la IMU para que este no se viera afectado.

Después de realizar las pruebas con la IMU, se observó que esta tardaba unos segundos en empezar a coger mediciones reales desde que se activaba, por lo que a la hora de la calibración inicial, se tuvieron que eliminar diversos valores debido a esta limitación. La forma más fácil y práctica que se encontró para realizar este control, fue realizar antes de la puesta en marcha un proceso de calibración de unos 10 segundos donde el usuario, tiene que colocar el robot lo más vertical posible, para establecer ese valor como la referencia.

Finalmente, ajustando los valores del controlador PD de estabilidad, el robot la mantiene en gran medida, aunque con pequeñas derivas debido a que un ajuste donde el robot esté en el mismo sitio en la vida real, es muy complicado e improbable, de la forma en la que se ha planteado. Aun así, se ha observado que el robot seguía muy bien el valor de referencia y se estabilizaba correctamente.

Posterior a esto se ha realizado el control manual, donde el usuario puede controlar el robot mediante el *smartphone*, lo que hace que se puedan corregir esas pequeñas derivas causadas por el control de estabilidad. Finalmente, el robot ha respondido correctamente al control manual, ya que sigue en gran medida las indicaciones del usuario a través de la aplicación.

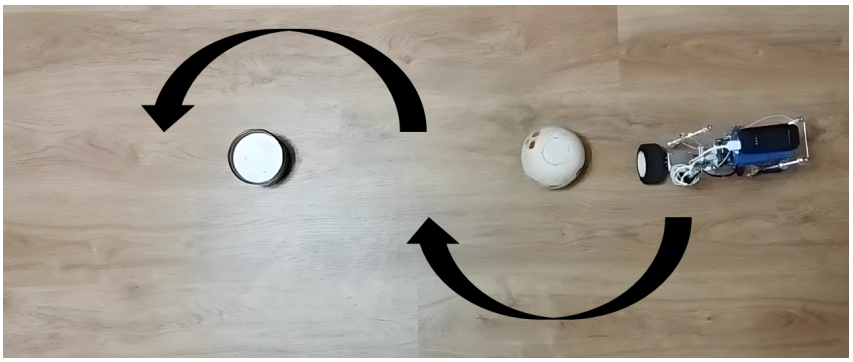
En la Figura 1.71 se puede observar el aspecto del robot autobalanceado con los componentes y la estructura que han sido necesarios para llevar a cabo la implementación.



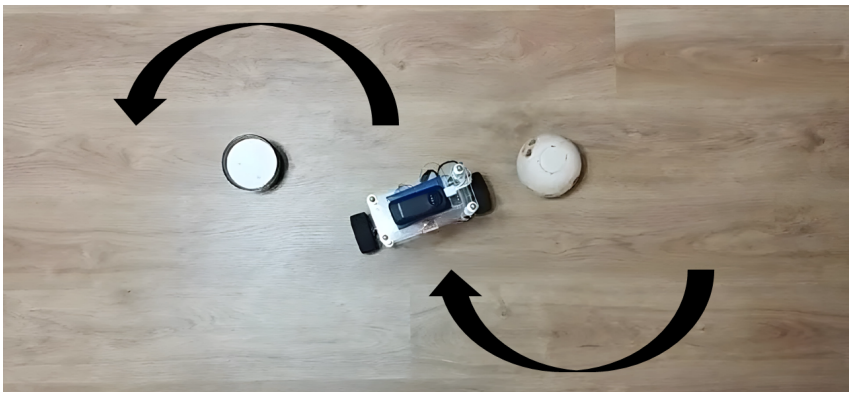
(a) Vista lateral en perspectiva del robot (b) Vista frontal en perspectiva del robot

Figura 1.71: Modelo final del vehículo autobalanceado tipo *Segway*.

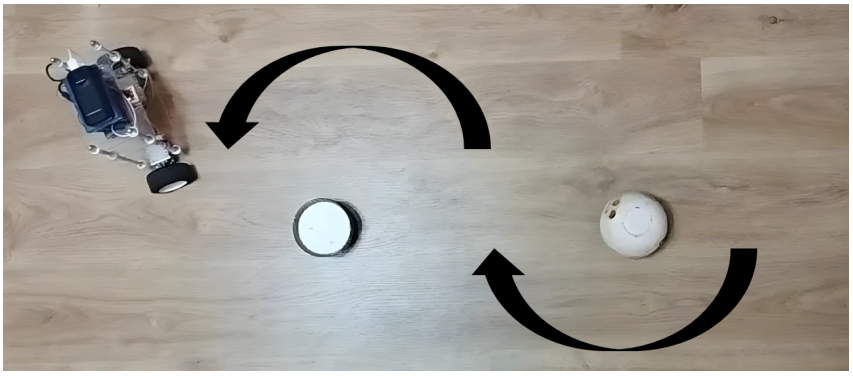
A continuación, en la Figura 1.72, se muestran una serie de imágenes que sirven para comprobar el buen funcionamiento del control manual del robot. En esta prueba se ha optado por realizar una secuencia, a modo *zigzag*, donde el usuario, tiene que hacer pasar al robot por unos huecos definidos.



(a) Posición 1 del *zigzag*



(b) Posición 2 del *zigzag*



(c) Posición 3 del *zigzag*

Figura 1.72: Trayectoria en forma de *zigzag* realizada para el control manual.

1.8. Conclusiones y mejoras futuras

El objetivo principal de este proyecto era diseñar, implementar y controlar un vehículo autobalanceado de dos ruedas tipo *Segway*, mediante una simulación e implementación real del prototipo. A través de este proyecto, se han logrado realizar un amplio abanico de simulaciones que permiten desde que el usuario controle el robot mediante un *gamepad*, pasando por un control de la velocidad y el giro, hasta un control preciso de trayectorias previamente definidas. En la parte de la implementación real, se ha logrado diseñar y construir un prototipo funcional, donde el usuario, mediante una aplicación móvil, es capaz de controlarlo manualmente, todo ello sin perder la estabilidad del vehículo. Tanto el trabajo como el proyecto en sí, han cumplido con los objetivos esperados de forma exitosa.

Este trabajo, englobado dentro del campo de la robótica y la mecatrónica ha abarcado varios campos vistos durante la formación universitaria, como pueden ser la electrónica, a través de la elección de los componentes y asegurando que todos ellos funcionen de forma conjunta y eficiente; la automática, ya que se han desarrollado diferentes algoritmos de control para mantener la estabilidad y controlar el robot; o la informática, para la programación del microcontrolador Arduino. También se han explorado otros campos como puede ser el diseño en *Software* de CAD 3D, como *Siemens NX* o *Inventor*, para el diseño mecánico del robot.

Algunas de las posibles mejoras futuras que se podrían implementar en el robot son:

- **Visión por computador**: Integrar cámaras para el control de estabilidad del robot, permitiéndole realizar tareas como la detección de obstáculos en tiempo real y la creación de mapas del entorno. Además, se podría mejorar la localización del robot mediante el uso de **GPS** o **balizas**. Aunque el GPS no es adecuado para posicionamiento en interiores, las balizas permitirían la triangulación de señales para determinar con precisión la posición del robot. Los sensores utilizados para medir el entorno exterior se denominan sensores **exteroceptivos**.

- Explorar la **odometría** como forma de estimar la posición del robot. Conociendo la posición angular de las ruedas y utilizando las ecuaciones de movimiento, junto con la geometría del vehículo, se puede realizar una estimación precisa de la ubicación actual del robot. Esto se podría realizar incorporando motores CC con ***encoders*** que permitirían realizar un control de trayectorias, al igual que se ha realizado en la simulación.

- Integrar **sensores** al robot para que realice diferentes aplicaciones. Por ejemplo, mediante un sensor de ultrasonidos, el robot puede detectar obstáculos y esquivarlos, o mediante un sensor sigue-líneas el robot puede seguir un determinado circuito marcado previamente. También se podrían haber integrado componentes como **matrices LED** o ***buzzers*** para emitir diferentes sonidos o luces de alarma en caso de que no funcione o que haya algún problema con la estabilidad.

- Implementar un algoritmo con un **filtro de Kalman**, que mejoraría la precisión de la estimación de la posición y la velocidad, además de eliminar el gran medida el ruido que pueda tener el sensor. Este tipo de filtros son útiles ya que pueden predecir estados futuros, de tal forma que permiten que el sistema de control anticipe movimientos y ajuste comandos de manera proactiva. Todo esto mejoraría la estabilidad y el control manual del robot de una forma significativa.

- Incorporar ***displays*** que indiquen los diferentes datos que está recogiendo el sensor. Este aspecto sería muy interesante ya que, cuando el robot está realizando la función de calibración, permitiría saber los valores que está midiendo el sensor, para saber si se está realizando correctamente.

- Añadir un **módulo WiFi** para establecer una comunicación inalámbrica con el robot, permitiendo así la transmisión de datos en tiempo real, o la integración en redes domésticas o industriales. También facilitaría el envío de datos a diferentes bases para su análisis.

- **Transporte de objetos**, mediante un mecanismo o soporte incluído en el robot.

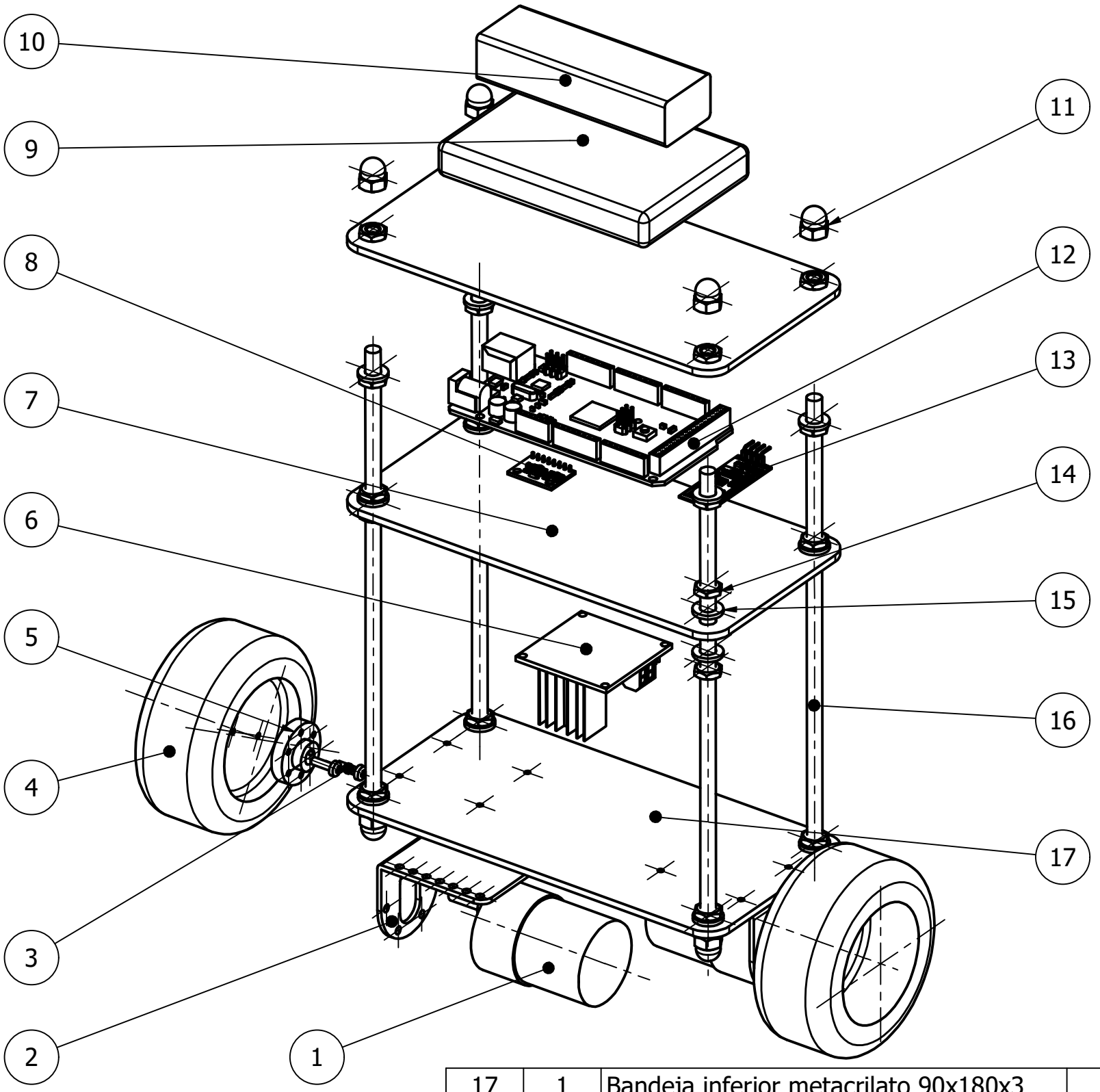
El presente proyecto contribuye con los siguientes Objetivos de Desarrollo Sostenible marcados por la ONU en relación con la Agenda 2030 (véase Anexo C: Objetivos de Desarrollo Sostenible).

Figura 1.73: Objetivos de Desarrollo Sostenible (ONU).

Capítulo 2

Planos

En este capítulo del presente proyecto se presentan los planos detallados del diseño del vehículo autobalanceado desarrollado. El conjunto de planos proporciona una visión general de la estructura, así como de las piezas que lo componen. Se incluye un planno de explosión con el desglose de cada uno de los componentes que forman parte del robot, seguido de un plano de vistas general desde distintas perspectivas. También se presentan dos planos de las bandejas que se han mecanizado y que componen el chasis del robot, junto con un plano del circuito electrónico general con las conexiones necesarias para la implementación.

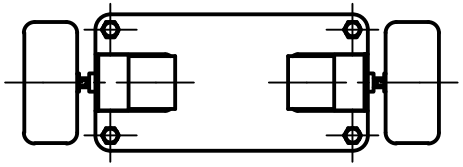


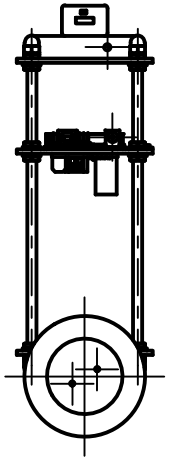
Nº	CANT	NOMBRE DE PIEZA	NORMA/MODELO/CARACT
17	1	Bandeja inferior metacrilato 90x180x3	Plano 4
16	4	Ejes roscados M6x21	-
15	16	Arandela Ø6	DIN 125
14	24	Tuerca M6	ISO 4035
13	1	Módulo bluetooth	HC-06
12	1	Microcontrolador Arduino Due	Procesador ARM Cortex-M3 de 32bits
11	8	Tuerca de tapa M6	DIN 1587
10	1	Batería recargable	5V a 4400mAh
9	1	Bateria de litio recargable	7,4V a 6000mAh
8	1	Sensor IMU	Módulo GY-521 Acelerómetro y giroscopio MPU-6050
7	2	Bandeja metacrilato 90x180x3	Plano 3
6	1	Driver puente H	L298N
5	2	Hub de montaje de aluminio para eje	1/4 M3
4	2	Rueda plástico Ø80	-
3	4	Tornillo M3x10	DIN 7985
2	2	Soporte L (bracket) aluminio	37D
1	2	Motor reductor 30:1	Metal Gearmotor 37Dx52L 12V

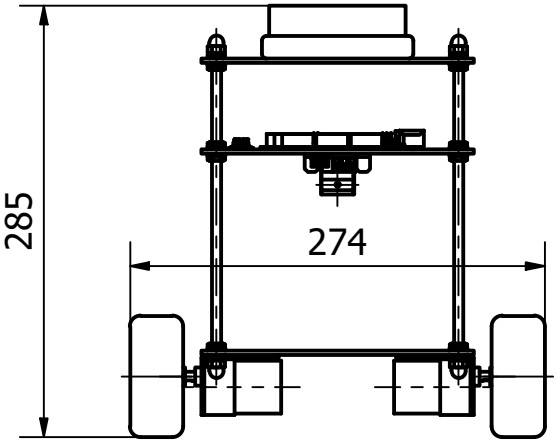
LISTA DE DESPIECE

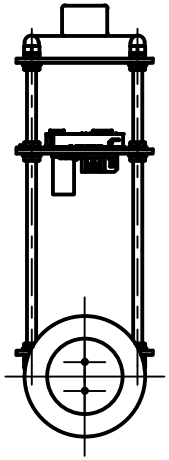
	Nombre	Fecha	Título del proyecto: **Diseño, Implementación Y Control De Un Vehículo Autobalanceado De Dos Ruedas Tipo Segway**
Dibujado	Adrián Sánchez	12/06/2024	ETSI Aeroespacial y Diseño Industrial
Comprobado	Vicente Casanova	12/06/2024	

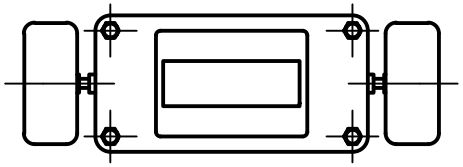
ESCALA	Denominación del plano	Nº de plano:	
1 : 2	**CONJUNTO ROBOT**	1	UNIVERSITAT POLITÈCNICA DE VALÈNCIA

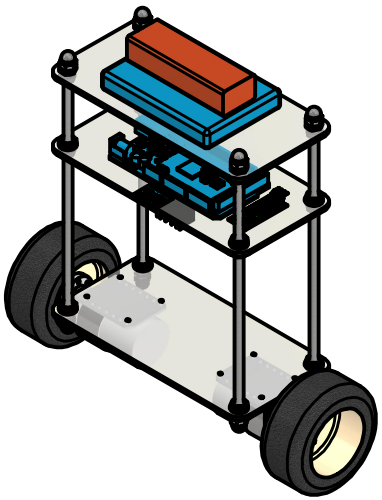












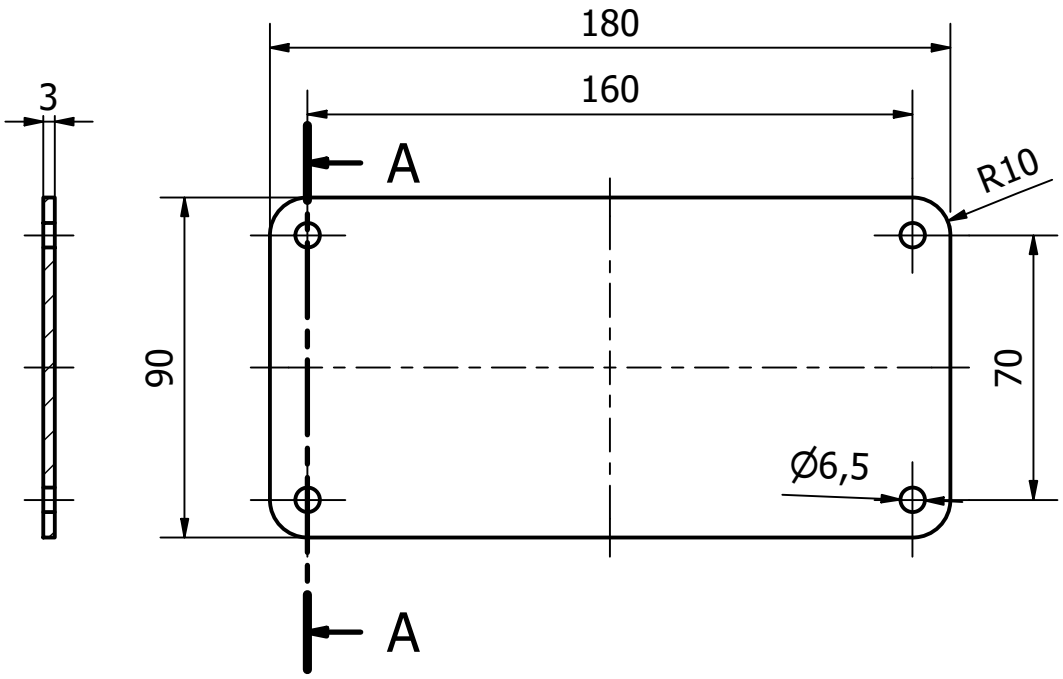
Escala 1:5

	Nombre	Fecha	Título del proyecto: **Diseño, Implementación Y Control De Un Vehículo Autobalanceado De Dos Ruedas Tipo Segway**
Dibujado	Adrián Sánchez	12/06/2024	
Comprobado	Vicente Casanova	12/06/2024	

ETSI Aeroespacial y Diseño Industrial

ESCALA	Denominación del plano	Nº de plano:
1 : 5	**VISTAS CONJUNTO**	2

UNIVERSITAT POLITÈCNICA DE VALÈNCIA



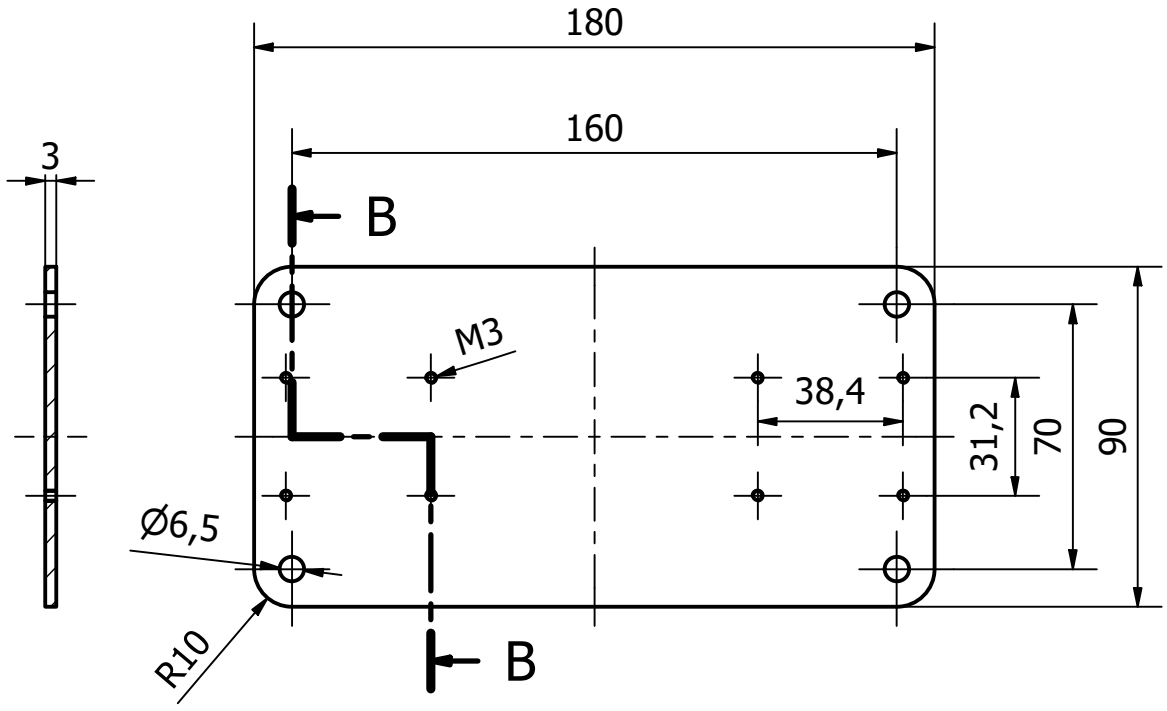
7	2	Bandeja metacrilato 90x180x3	Plano 3
Nº	CANT	NOMBRE DE PIEZA	NORMA/MODELO/CARACT

LISTA DE DESPIECE

	Nombre	Fecha	Título del proyecto: **Diseño, Implementación Y Control De Un Vehículo Autobalanceado De Dos Ruedas Tipo Segway**
Dibujado	Adrián Sánchez	12/06/2024	
Comprobado	Vicente Casanova	12/06/2024	

ETSI Aeroespacial y Diseño Industrial

ESCALA	Denominación del plano	Nº de plano:	
1 : 2	**BANDEJAS SUPERIORES**	3	UNIVERSITAT POLITÈCNICA DE VALÈNCIA



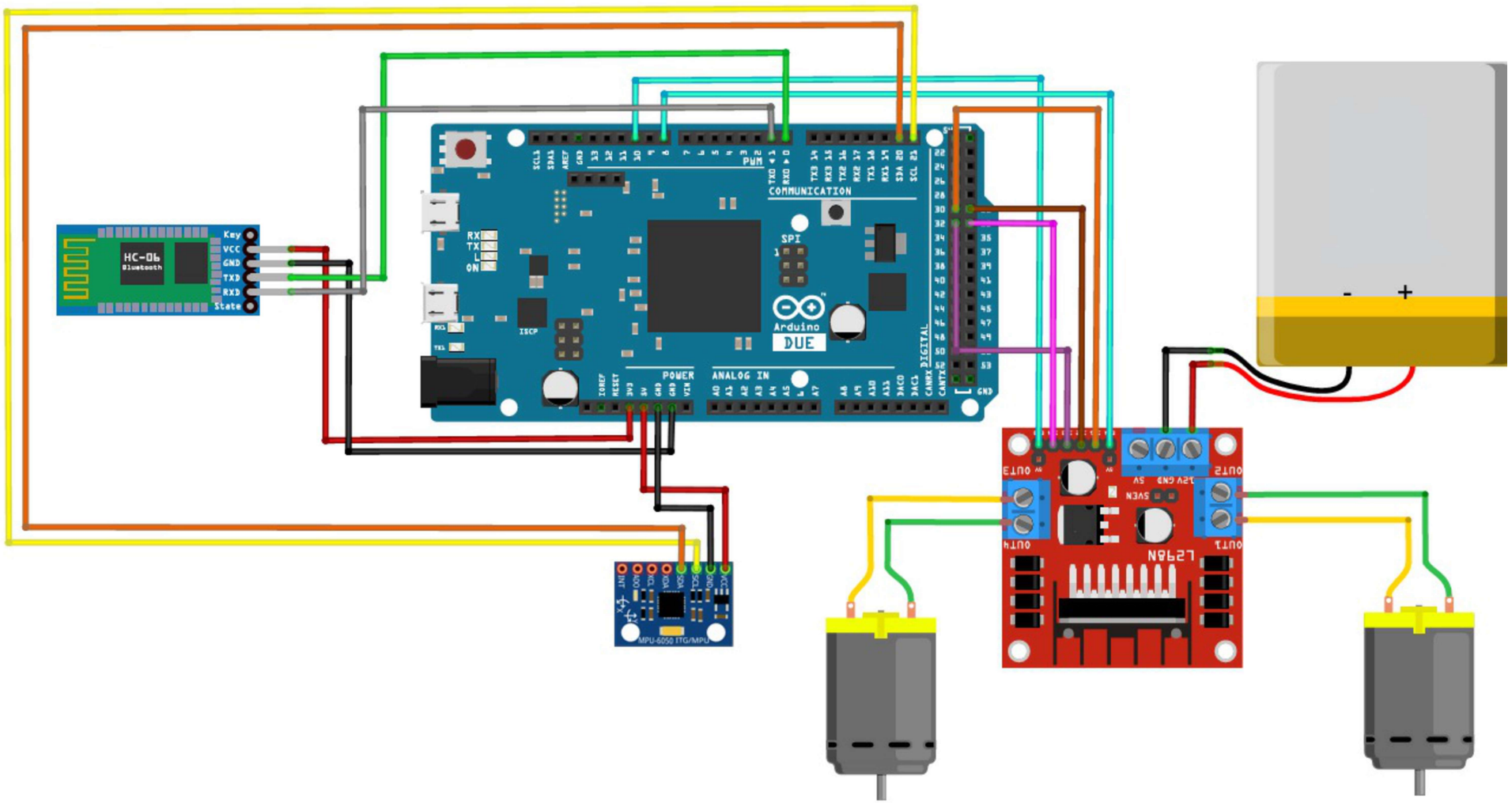
17	1	Bandeja inferior metacrilato 90x180x3	Plano 4
Nº	CANT	NOMBRE DE PIEZA	NORMA/MODELO/CARACT

LISTA DE DESPIECE

	Nombre	Fecha	Título del proyecto: **Diseño, Implementación Y Control De Un Vehículo Autobalanceado De Dos Ruedas Tipo Segway**
Dibujado	Adrián Sánchez	12/06/2024	
Comprobado	Vicente Casanova	12/06/2024	

ETSI Aeroespacial y Diseño Industrial

ESCALA	Denominación del plano	Nº de plano:	
1 : 2	**BANDEJA INFERIOR**	4	UNIVERSITAT POLITÈCNICA DE VALÈNCIA



	Nombre	Fecha	Título del proyecto: **Diseno, Implementación Y Control De Un Vehículo Autobalanceado De Dos Ruedas Tipo Segway**	ETSI Aeroespacial y Diseño Industrial
Dibujado	Adrián Sánchez	12/06/2024		
Comprobado	Vicente Casanova	12/06/2024		
ESCALA	Denominación del plano		Nº de plano: 5	UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CIRCUITO FINAL

Capítulo 3

Pliego de condiciones

En este capítulo se realizará la especificación técnica del proyecto.

3.1. Objeto

La presente especificación técnica se refiere al diseño e implementación de un prototipo de robot autobalanceado. El proyecto tiene como objetivo desarrollar un sistema de control que permita al robot mantener el equilibrio de manera autónoma. El ámbito de aplicación de este documento se extiende a todos los sistemas mecánicos y electrónicos que integran la presente instalación.

El alcance del proyecto incluye el diseño en 3D del robot, la simulación del comportamiento dinámico, y la implementación física del prototipo. Se asume que el equipo requerido para los procesos y los materiales electrónicos obtenidos de proveedores externos han sido sometidos a una evaluación previa de calidad.

3.2. Condiciones de los materiales

3.2.1. Bandejas de la estructura

Descripción

Las 3 bandejas de la estructura tendrán un grosor de 3mm y estarán fabricadas con polimetilmetacrilato (PMMA) de alta calidad. Las placas deberán ser transparentes, con una tolerancia de grosor de ± 0.1 mm y dimensiones según las especificaciones del diseño.

Control de calidad

Previamente a su uso, se verificará que las placas de metacrilato no presenten fisuras, burbujas ni impurezas. Se comprobará que las dimensiones cumplen con las especificaciones establecidas y que las placas tienen una alta resistencia a impactos y buena calidad óptica. Además, las placas deberán estar protegidas con una película plástica removible para evitar rayaduras durante su manipulación y montaje.

3.2.2. Tornillería y elementos de fijación

Descripción

Los elementos de fijación y tornillería seguirán las especificaciones detalladas en los planos de montaje proporcionados (véase Capítulo 2: Planos). Se utilizarán materiales estándar de alta calidad para garantizar la integridad estructural del robot.

Control de calidad

Antes del uso, se comprobará que todos los elementos de tornillería cumplen con las dimensiones especificadas y que no presentan defectos visibles como deformaciones o falta de recubrimiento cincado. También se verificarán las propiedades mecánicas y la resistencia a la corrosión conforme a las normas ISO DIN 13 para los tornillos y UNE-EN ISO 898 para las tuercas y arandelas.

3.2.3. Baterías

Descripción

Se utilizarán dos tipos de baterías. La primera es una batería de litio recargable de 7.4V destinada a proporcionar una alimentación estable y de larga duración para el *driver* que alimentará ambos motores. La segunda es una batería tipo *powerbank* de 5V que se utilizará para alimentar al microcontrolador Arduino Due mediante un cable micro USB. Ambas baterías están diseñadas para proporcionar un rendimiento confiable y duradero en diversas condiciones de operación.

Control de calidad

Antes del uso, se realizará una prueba exhaustiva de cada batería para asegurarse de que cumplan con las especificaciones de capacidad y voltaje indicadas. Se verificará que las baterías no presenten signos de daño físico ni defectos que puedan comprometer su rendimiento. Además, las baterías deberán cumplir con las normativas de seguridad y calidad establecidas en las normas UNE-EN IEC 61960 para asegurar su fiabilidad y seguridad durante el uso.

3.2.4. Componentes electrónicos

Descripción

Los componentes electrónicos utilizados en el prototipo del robot autobalanceado deberán ser de alta calidad y provenientes de fabricantes reconocidos. Se utilizarán microcontroladores, sensores y actuadores compatibles con el diseño y las especificaciones del proyecto.

Control de calidad

Se asume que los componentes electrónicos han pasado un control de calidad por parte del fabricante o del proveedor antes de ser adquiridos para su uso en el proyecto. Se verificará que los componentes recibidos cumplan con las especificaciones técnicas y las normativas aplicables antes de su integración en el prototipo del robot.

3.3. Condiciones de ejecución

3.3.1. Estructura del robot

Descripción

El ensamblaje de la estructura del robot se realizará de forma cuidadosa y precisa, siguiendo un proceso meticuloso para garantizar su correcto funcionamiento. Este montaje se realizará de acuerdo al plano del conjunto del robot (véase Capítulo 2: Planos).

Durante este proceso, se prestará especial atención a la correcta alineación y fijación de cada componente, asegurando una estructura robusta y estable que permita el equilibrio del robot durante su funcionamiento. Se seguirán las instrucciones de diseño detalladas previamente, asegurando que cada pieza se coloque en su posición correspondiente de acuerdo con el diseño realizado.

Control de calidad

Tras el montaje completo de la estructura del robot robot, se realizará una inspección exhaustiva para verificar que todas las partes estén correctamente ensambladas y que no haya ninguna anomalía que pueda afectar su funcionamiento. Se verificará la alineación de los componentes, la integridad estructural del chasis y la sujeción adecuada de los motores y sensores, de acuerdo con el modelo diseñado para la implementación.

3.3.2. Circuito electrónico

Descripción

Las conexiones del circuito electrónico se realizarán meticulosamente y de acuerdo con las especificaciones técnicas de los componentes. Se emplearán técnicas de soldadura de alta calidad y se seguirán las normativas y estándares aplicables para garantizar la confiabilidad y durabilidad de las conexiones. La soldadura se realizará en un banco que permita regular la temperatura de soldado, con un límite máximo de temperatura de 275°C. El montaje se realizará según el plano del circuito final (véase Capítulo 2: Planos).

Control de calidad

Antes del uso, se verificará que todas las conexiones estén realizadas correctamente. Se realizarán pruebas de continuidad y se medirá la resistencia de las conexiones para asegurar su integridad y fiabilidad. Además, se llevará a cabo una inspección visual para detectar posibles defectos o irregularidades en las soldaduras.

3.3.3. Programación del robot

Descripción

La programación del robot autobalanceado consistirá en el desarrollo del *software* necesario para controlar su funcionamiento, incluyendo el algoritmo de control de estabilidad, la interfaz de usuario y las funciones para el control manual. Se implementarán algoritmos de control PID para mantener el equilibrio del robot en diferentes condiciones y para realizar correcciones en tiempo real.

Control de calidad

Después de la programación del robot, se realizarán pruebas exhaustivas para verificar su funcionamiento correcto en diversas condiciones. Se probarán todas las funciones y características del *software*, y se corregirán cualquier error o anomalía identificada durante las pruebas. Se garantizará que el programa sea capaz de mantener el equilibrio del robot de manera autónoma y responder de manera adecuada a las entradas del usuario en tiempo real.

3.4. Prueba de servicio

La prueba de servicio se llevará a cabo realizando diversas validaciones destinadas a comprobar tanto el equilibrio y la estabilidad del robot como su capacidad para responder de manera precisa a las señales de control proporcionadas por el usuario a través de la aplicación.

Durante estas pruebas, se registrarán y analizarán meticulosamente una serie de parámetros clave, entre ellos la velocidad de desplazamiento, la aceleración, la inclinación y la estabilidad general del robot. Además, se realizarán ajustes y modificaciones según sea necesario con el fin de optimizar el rendimiento y garantizar la fiabilidad del sistema en diversas condiciones de operación.

Los resultados de estas pruebas proporcionarán información valiosa para mejorar el diseño y la funcionalidad del robot, garantizando así su eficacia y eficiencia.

En ciertas circunstancias, podrán considerarse alternativas a las soluciones especificadas en este pliego de condiciones técnicas, siempre y cuando se justifique su necesidad y no comprometan los estándares mínimos de calidad requeridos.

Capítulo 4

Presupuesto

En el presente capítulo se desarrollará el presupuesto del proyecto. Se realizará un análisis económico de los componentes electrónicos y de los materiales utilizados, así como de las licencias y programas de *software* empleados. Se presentarán los precios unitarios y descompuestos, para luego ofrecer un resumen general del presupuesto del proyecto.

4.1. Cuadro de precios unitarios

Cuadro de materiales			
Ref.	**Ud.**	**Descripción**	**Precio (€)**
E1	ud.	Bandeja metacrilato 90x180x3	2.05
E2	ud.	Rueda de 90 mm de diámetro.	4.30
E3	ud.	Varilla roscada diámetro M6 1m, cincado.	0.65
E4	ud.	Tornillo estándar diámetro M3 10 mm, cincado.	0.04
E5	ud.	Tuerca hexagonal estándar diámetro M3, cincado.	0.04
E6	ud.	Tornillo estándar diámetro M3 5 mm, cincado.	0.04
E7	ud.	Arandela plana estándar diámetro M6, cincado.	0.03
E8	ud.	Tuerca ciega estándar diámetro M6, cincado.	0.18
E9	ud.	Tuerca hexagonal estándar diámetro M6 cincado.	0.06
E10	ud.	Brackets en L de aluminio para motor de 37 mm, 2 ud.	10.26
E11	ud.	Hub de aluminio para eje de 1/4 rosca M3, 2 ud.	10.73
E12	ud.	Arduino Due	42.00
E13	ud.	Controlador Motor L298N	8.00
E14	ud.	Motorreductores 30:1 37Dx52L mm 12 V Polulu	33.37
E15	ud.	Módulo GY-521 Acelerómetro y Giroscopio MPU-6050	5.99
E16	ud.	Módulo Bluetooth HC-06	3.99
E17	ud.	Cables Macho-Macho, pack 10 unidades	0.70
E18	ud.	Cables Macho-Hembra, pack 10 unidades	0.70
E19	ud.	Cables Hembra-Hembra, pack 10 unidades	0.70
E20	ud.	Batería de litio recargable 7,4V a 6000 mAh	24.95
E21	ud.	Batería tipo powerbank 5V a 4400 mAh	4.50
E22	ud.	Cable Micro USB	1.50

Tabla 4.1: Cuadro de materiales.

Cuadro de maquinaria			
Ref.	**Ud.**	**Descripción**	**Precio (€)**
MA1	h.	Estación de soldadura	0.078
MA2	h.	Equipo informático (ordenador de sobremesa)	0.075
MA3	h.	Multímetro	0.003
MA4	h.	Licencia Siemens NX	3.6198
MA5	h.	Licencia Inventor	1.388
MA6	h.	Licencia Matlab	0.432
MA7	h.	Licencia Simulink	0.654
MA8	h.	Licencia Simescape Multibody	0.654

Tabla 4.2: Cuadro de maquinaria.

Cuadro de mano de obra			
Ref.	**Ud.**	**Descripción**	**Precio (€)**
MO1	h.	Ingeniero en Electrónica Industrial y Automática	18.00

Tabla 4.3: Cuadro de mano de obra.

4.2. Cuadro de precios descompuestos

Ref.	Ud.	Descripción	Precio(€)	Cant.	Total(€)
d1	ud.	Realización de la simulación del robot			
MA2	h.	Equipo informático (ordenador de sobremesa)	0.075	200	15.00
MA4	h.	Licencia Siemens NX	3.6198	10	36.20
MA6	h.	Licencia Matlab	0.432	190	82.08
MA7	h.	Licencia Simulink	0.654	190	124.26
MA8	h.	Licencia Simescape Multibody	0.654	190	124.26
MO1	h.	Ingeniero en Electrónica Industrial y Automática	15.00	200	3000.00
%		Mano de obra sobre costes directos	3381.80	2 %	67.64
		Total precio de ejecución del material			**3449.44**

Tabla 4.4: Partida 1 de precios descompuestos.

Ref.	Ud.	Descripción	Precio(€)	Cant.	Total
d2	ud.	Implementación real del robot: Programación del control			
E22	ud.	Cable Micro USB	1.50	1	1.50
MA2	h.	Equipo informático (ordenador de sobremesa)	0.075	110	8.25
MA3	h.	Multímetro	0.003	5	0.015
MA5	h.	Licencia Inventor	1.388	10	13.88
MO1	h.	Ingeniero en Electrónica Industrial y Automática	18.00	110	1650.00
%		Mano de obra sobre costes directos	2003.65	2 %	40.07
		Total precio de ejecución del material			**2043.73**

Tabla 4.5: Partida 2 de precios descompuestos.

Ref.	**Ud.**	**Descripción**	**Precio(€)**	**Cant.**	**Total(€)**
d3	ud.	Implementación real del robot: montaje del vehículo			
E1	ud.	Bandeja metacrilato 90x180x3	2.05	3	6.15
E2	ud.	Rueda de 90 mm de diámetro.	4.30	2	8.60
E3	ud.	Varilla roscada M6 1m.	0.65	1	0.65
E4	ud.	Tornillo estándar M3 10 mm.	0.04	12	0.48
E5	ud.	Tuerca hexagonal estándar M3.	0.04	8	0.32
E6	ud.	Tornillo estándar M3 5 mm.	0.04	12	0.48
E7	ud.	Arandela plana estándar M6.	0.03	20	0.60
E8	ud.	Tuerca ciega estándar M6.	0.18	8	1.44
E9	ud.	Tuerca hexagonal estándar M6.	0.06	24	1.44
E10	ud.	Brackets en L de aluminio, 2 uds.	10.26	1	10.26
E11	ud.	Hub de aluminio, 2 uds.	10.73	1	10.73
E12	ud.	Arduino Due	42.00	1	42.00
E13	ud.	Controlador Motor L298N	8.00	1	8.00
E14	ud.	Motorreductores 30:1 Metal Gearmotor 37Dx52L mm 12 V de Polulu	33.37	2	66.74
E15	ud.	Módulo GY-521 MPU-6050	5.99	1	5.99
E16	ud.	Módulo Bluetooth HC-06	3.99	1	3.99
E17	ud.	Cables Macho-Macho, 10 ud.	0.70	1	0.70
E18	ud.	Cables Macho-Hembra, 10 ud.	0.70	2	1.40
E19	ud.	Cables Hembra-Hembra, 10 ud.	0.70	1	0.70
E20	ud.	Batería de litio recargable 7,4V	24.95	1	24.95
E21	ud.	Batería tipo powerbank 5V	4.50	1	4.50
MA1	h.	Estación de soldadura	0.078	1	0.078
MA3	h.	Multímetro	0.003	5	0.015
MO1	h.	Ingeniero en Electrónica Industrial y Automática	15.00	20	300.00
%		Mano de obra sobre costes directos	500.21	2 %	10.01
		Total precio de ejecución del material			**510.22**

Tabla 4.6: Partida 3 de precios descompuestos.

4.3. Cuadro de mediciones y ejecución del material

Cuadro de mano de obra					
Ref.	**Ud.**	**Descripción**	**Precio(€)**	**Cant.**	**Total(€)**
d1	ud.	Realización de la simulación del robot	3499.44	1.000	3499.44
d2	ud.	Implementación real del robot: Programación del control	2043.73	1.000	2043.73
d3	ud.	Implementación real del robot: montaje del vehículo	510.22	1.000	510.22
				Total precio de ejecución del material	**6003.39**

Tabla 4.7: Cuadro de mediciones.

4.4. Resumen general del presupuesto

Tabla resumen general presupuesto		
Partidas	**Contenido**	**Importe (€)**
d1	Realización de la simulación del robot	3499.44
d2	Implementación real del robot: Programación del control	2043.73
d3	Implementación real del robot: montaje del vehículo	510.22
	Total precio de ejecución del material	**6003.39**
Coste general y beneficio industrial		
13 % de gastos generales		780.44
6 % de beneficio industrial		360.20
Suma parcial		7144.03
21 % IVA		1500.25
Total ejecución del material		**8644.28**
Honorarios proyecto		
5 % sobre presupuesto de ejecución de material		300.17
21 % IVA sobre honorarios proyecto		63.04
Total honorarios proyecto		363.21
Total honorarios ingeniero		**363.21**
TOTAL PRESUPUESTO GENERAL		**9007.49**

Tabla 4.8: Cuadro resumen general presupuesto.

Asciende el presupuesto general a la expresada cantidad de NUEVE MIL SIETE EUROS CON CUARENTA Y NUEVE CÉNTIMOS.

Anexo A

Especificaciones técnicas del sensor MPU6050	
Característica	**Valor**
Voltaje de operación	3V/3.3V hasta 5V DC
Regulador de voltaje en placa	Sí
Grados de libertad (DoF)	6
Rango Acelerómetro	2g/4g/8g/16g
Rango Giroscopio	250Grad/Seg, 500Grad/Seg, 1000Grad/Seg, 2000Grad/Seg
Sensibilidad Giroscopio	131 LSBs/dps
Interfaz	I2C
Conversor AD	16 Bits (salida digital)
Tamaño	2.0cm x 1.6cm x 0.3cm

Tabla A.1: Especificaciones técnicas del sensor MPU6050.

Especificaciones técnicas del motor 37Dx52L de Pololu	
Característica	**Valor**
Relación de engranajes	30:1
Velocidad en vacío @ 12V	330 rpm
Corriente en vacío @ 12V	0.2 A
Corriente de bloqueo @ 12V	5.5 A
Par de bloqueo @ 12V	14 kg·cm
Potencia máxima de salida @ 12V	12 W
Velocidad en vacío @ 6V	170 rpm
Corriente en vacío @ 6V	0.15 A
Corriente de bloqueo @ 6V	3.0 A
Par de bloqueo @ 6V	7.9 kg·cm
Tipo de motor	12V

Tabla A.2: Especificaciones técnicas del motor 37Dx52L de Pololu.

Especificaciones del Controlador de Motor L298N	
Característica	**Valor**
Chip	L298N
Canales	2 (soporta 2 motores DC o 1 motor PAP)
Voltaje lógico	5V
Voltaje de potencia (V motor)	5V - 35V DC
Consumo de corriente (lógico)	0 a 36mA
Capacidad de corriente	2A (picos de hasta 3A)
Potencia máxima	25W
Dimensiones	43 x 43 x 27 mm
Peso	30g

Tabla A.3: Especificaciones del controlador L298N.

Especificaciones técnicas del microcontrolador Arduino Due	
Característica	**Valor**
Microcontrolador	AT91SAM3X8E
Tensión de funcionamiento	3.3V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines de E/S digitales	54 (12 de ellos proporcionan salida PWM)
Pines de entrada analógica	12
Pines de salida analógica (DAC)	2
Corriente total de salida continua en todas las líneas de E/S	130 mA
Corriente DC por Pin 3.3V	800 mA
Corriente DC por Pin 5V	800 mA
Memoria Flash	512 KB disponibles para todas las aplicaciones de usuario
SRAM	96 KB (dos bancos: 64KB y 32KB)
Velocidad del reloj	84 MHz

Tabla A.4: Especificaciones técnicas del microcontrolador Arduino Due.

Anexo B

```
// ========================================================
//                      LIBRERIAS
// ========================================================

#include <Wire.h>                   // Libreria para comunicacion I2C

// ========================================================
//                      CONSTANTES
// ========================================================

#define LED 53                      // Pin del LED de funcionamiento
#define MPU 0x68                    // Direccion I2C del MPU6050
#define MIN_ACC -12.0               // Valor minimo de la acc_cont
#define MAX_ACC 12.0                // Valor maximo de la acc_cont

#define GND 47                      // Pin de masa
#define VOLT   46                   // Pin del voltaje

#define ACCEL_RATIO 16384.0         // Ratio de conversion para el accel
#define GIRO_RATIO 131.0            // Ratio de conversion para el gir
#define RAD_TO_DEG 57.295779        // Factor de conversion

// Pines de control del motor derecho
#define EN_A 8                      // Pin de habilitacion del motor der
#define IN1 30                      // Pin de entrada 1 del motor der
#define IN2 31                      // Pin de entrada 2 del motor der

// Pines de control del motor izquierdo
#define EN_B 10                     // Pin de habilitacion del motor izq
#define IN3 32                      // Pin de entrada 1 del motor izq
#define IN4 33                      // Pin de entrada 2 del motor izq

// ========================================================
//                      VARIABLES
// ========================================================

float t = 0.0;
long time1 = 0;
long time2 = 0;
int T_muest = 10;
float T_muest_seg = T_muest / 1000.0;
```

```
42 // ========================================================
43 //               VAR - ACCIONES CONTROL MOTORES
44 // ========================================================
45
46 float acc = 0.0;
47 float acc_1 = 0.0;
48 float acc_2 = 0.0;
49
50 // ========================================================
51 //                    PD ESTABILIDAD
52 // ========================================================
53
54 float K_prop = 0;                    // Ganancia proporcional
55 float K_der = 0;                     // Ganancia derivativa
56 float AngRef;                        // Referencia de angulo
57 float Error = 0.0;                   // Error
58 float Der_error = 0.0;               // Derivada del error
59
60 // ========================================================
61 //               ACELEROMETRO IMU MPU 6050
62 // ========================================================
63
64 float rollAngle = 0.0;              // Angulo de balanceo
65 float angR_nuevo = 0.0;
66 float initialRoll = 0.0;            // Angulo inicial de balanceo
67 float angR_prev = 0.0;
68
69 // ========================================================
70 //                   BLUETOOTH HC-06
71 // ========================================================
72
73 byte inByte[8] ;
74 int vx,vy,dx,dy ;
75 float vxf,vyf,dxf,dyf,vel,dir ;
76
77 // ========================================================
78 // ||                                                    ||
79 // ||               CONFIGURACION INICIAL                ||
80 // ||                                                    ||
81 // ========================================================
82
83 void setup() {
84
85   Serial.begin(115200) ;
86   Serial1.begin(115200) ;
87
88   pinMode(LED,OUTPUT) ;
89
90   analogWriteResolution(12);
91   analogReadResolution(12);
92
93
94
```

```
95    // CONFIGURACION DEL ACELEROMETRO
96    Wire.begin();
97    Wire.beginTransmission(MPU);
98    Wire.write(0x6B);
99    Wire.write(0);
100   Wire.endTransmission(true);
101
102   Calibrado();
103
104   // CONFIGURACION DE LOS PINES
105   pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
106   pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
107 }
108
109 // ============================================================
110 // ||                                                        ||
111 // ||                  BUCLE PRINCIPAL                       ||
112 // ||                                                        ||
113 // ============================================================
114
115 void loop() {
116
117   Muestra();
118
119   if (Serial1.available()>=7) {
120     for (int ii=1 ; ii<=7 ; ii++) {
121       inByte[ii]=Serial1.read() ;
122     }
123
124     if (inByte[1]=='V') {
125       if (inByte[2]=='X' && inByte[5]=='Y') {
126         vx=(inByte[3]-48)*10+(inByte[4]-48) ;
127         vxf=map(vx,10,99,-100,100)/100.0 ;
128         vy=(inByte[6]-48)*10+(inByte[7]-48) ;
129         vyf=map(vy,10,99,-100,100)/100.0 ;
130         vel=-1*vyf ;
131       }
132     }
133
134     if (inByte[1]=='D') {
135       if (inByte[2]=='X' && inByte[5]=='Y') {
136         dx=(inByte[3]-48)*10+(inByte[4]-48) ;
137         dxf=map(dx,10,99,-100,100)/100.0 ;
138         dir=dxf ;
139         dy=(inByte[6]-48)*10+(inByte[7]-48) ;
140         dyf=map(dy,10,99,-100,100)/100.0 ;
141       }
142     }
143   }
144
145   Serial.print("Vel=") ; Serial.print(vel) ; Serial.print("\t") ;
146   Serial.print("Dir=") ; Serial.print(dir) ; Serial.print("\t") ;
147
```

```
148     // PD ESTABILIDAD
149     K_prop = 5.05;
150     K_der = 0.22;
151     float Km = 1.0;
152     float Kt = 6.0;
153 
154     AngRef = 0;
155     Error = AngRef - rollAngle;
156     Der_error = rollAngle / T_muest_seg;
157     acc = K_prop * Error + K_der * Der_error;
158 
159     acc_1 = acc + Kt*vel + Km*dir ;
160     acc_2 = acc + Kt*vel - Km*dir ;
161 
162 
163   // DIRECCION MOTORES
164   if (acc_1 > 0.00) {digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);}
165   if (acc_2 > 0.00) {digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);}
166 
167   if (acc_1 < 0.00) {digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);}
168   if (acc_2 < 0.00) {digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);}
169 
170   // ACCION DE CONTROL MOTORES
171   if(acc_1 >= MAX_ACC) {acc_1 = MAX_ACC;}
172   if(acc_2 >= MAX_ACC) {acc_2 = MAX_ACC;}
173   if(acc_1 <= MIN_ACC) {acc_1 = MIN_ACC;}
174   if(acc_2 <= MIN_ACC) {acc_2 = MIN_ACC;}
175 
176   analogWrite(EN_A,map(abs(acc_1)*10,0,MAX_ACC*10,700,4095));
177   analogWrite(EN_B,map(abs(acc_2)*10,0,MAX_ACC*10,700,4095));
178 
179   Serial.print("rollAngle=") ; Serial.print(rollAngle) ; Serial.print("\t
      ") ;
180   Serial.print("acc_1=") ; Serial.print(acc_1) ; Serial.print("\t") ;
181   Serial.print("acc_2=") ; Serial.print(acc_2) ; Serial.println("\t") ;
182 
183   // FUNCIONAMIENTO LED
184   time2=millis() ;
185   if (time2-time1>T_muest) {digitalWrite(LED,HIGH) ;}
186   else {digitalWrite(LED,LOW) ;}
187   while (time2-time1<T_muest) {time2=millis() ;}
188   time1=millis() ;
189 
190 }
191 
192 // ============================================================
193 // ||                                                        ||
194 // ||                      FUNCIONES                         ||
195 // ||                                                        ||
196 // ============================================================
197 
198 
199 
```

```
200 // ========================================================
201 //              FUNCION CALIBRADO ACELEROMTERO
202 // ========================================================
203 void Calibrado() {
204   digitalWrite(LED, HIGH);
205   Serial.print("Valores iniciales");
206   float angleSum = 0.0;
207   int cc = 0;
208   for (int ii = 1; ii <= 1000; ii++) {
209     Muestra();
210 
211     if(ii>=200) {
212       angleSum += rollAngle;
213       delay(10);
214       Serial.print(rollAngle) ; Serial.print("\t") ; Serial.println(ii) ;
215       if (cc>=100) {Serial.print(".") ;
216       digitalWrite(LED,HIGH) ; delay(50) ; digitalWrite(LED,LOW) ; delay
    (50) ;
217       cc=0 ;} else {cc++ ;}
218     }
219   }
220 
221   initialRoll = angleSum / 800.0;
222   Serial.print("Alabeo inicial: "); Serial.println(initialRoll);
223   delay(1000) ;
224 
225 }
226 
227 // ========================================================
228 //               FUNCION MUESTRA ACELEROMETERO
229 // ========================================================
230 void Muestra() {
231   int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;
232   float Acc, Gy;
233 
234   // Leer los valores del Acelerometro de la IMU
235   Wire.beginTransmission(MPU);
236   Wire.write(0x3B);
237   Wire.endTransmission(false);
238   Wire.requestFrom(MPU, 6, true);
239   AcX = Wire.read() << 8 | Wire.read();
240   AcY = Wire.read() << 8 | Wire.read();
241   AcZ = Wire.read() << 8 | Wire.read();
242 
243   Acc = atan(-1 * (AcX / ACCEL_RATIO) / sqrt(pow((AcY / ACCEL_RATIO), 2)
    + pow((AcZ / ACCEL_RATIO), 2))) * RAD_TO_DEG;
244 
245   // Leer valores IMU
246   Wire.beginTransmission(MPU);
247   Wire.write(0x43);
248   Wire.endTransmission(false);
249   Wire.requestFrom(MPU, 4, true);
250 
```

```
251
252   GyX = Wire.read() << 8 | Wire.read();
253   GyY = Wire.read() << 8 | Wire.read();
254
255   // Calcular angulo IMU
256   Gy = GyY / GIRO_RATIO;
257
258   // Filtro complementario
259   rollAngle = (0.98 * (angR_prev + Gy * T_muest_seg) + 0.02 * Acc) ;
260   rollAngle = rollAngle - initialRoll ;
261   angR_prev = rollAngle + initialRoll;
262 }
```

Anexo C

En este proyecto, se añade un anexo que detalla los objetivos específicos del proyecto, destacando especialmente el compromiso de esta labor. El objetivo principal es contribuir al progreso de al menos cinco de los **Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas**. Esta iniciativa se enmarca en la aspiración de generar un impacto positivo en áreas de importancia global, en consonancia con la agenda de desarrollo sostenible de la ONU.

Figura 1: Objetivos de Desarrollo Sostenible (ONU).[1]

[1] `https://www.un.org/sustainabledevelopment/es/`

Los **Objetivos de Desarrollo Sostenible (ODS)** son una serie de 17 objetivos globales establecidos por las Naciones Unidas en 2015 para abordar los mayores desafíos que enfrenta la humanidad. Estos objetivos apuntan a poner fin a la pobreza, proteger el planeta y garantizar la paz y la prosperidad para todos para 2030. Los Objetivos de Desarrollo Sostenible cubren diversas áreas como educación, salud, igualdad de género, trabajo decente y crecimiento económico, promoviendo el desarrollo sostenible e integrando esfuerzos sociales, económicos y ambientales. (Naciones Unidas, 2015)

La **Agenda 2030** es un plan de acción global adoptado por los estados miembros de las Naciones Unidas en 2015 para proporcionar un marco para lograr los Objetivos de Desarrollo Sostenible. Este programa presenta una visión integrada para el desarrollo sostenible, reconociendo la interdependencia de los objetivos y la necesidad de un enfoque colaborativo en todos los sectores de la sociedad. La Agenda 2030 para el Desarrollo Sostenible enfatiza la importancia de no dejar a nadie atrás, priorizando la inclusión y el avance de los grupos más vulnerables y rezagados.(Ministerio de Derechos Sociales, Consumo y Agenda 2030, 2024)

ODS 4: Educación de Calidad

Este objetivo se centra en garantizar una educación inclusiva, igualitaria y de calidad, así como en promover oportunidades de aprendizaje durante toda la vida para todos. Eso significa educación igualitaria para todos, sin discriminación. Con ello se pretende eliminar las brechas y desigualdades de género en el acceso a la educación, garantizando que todas las personas, independientemente de su nivel socioeconómico, tengan la oportunidad de adquirir los conocimientos y habilidades necesarios para una participación plena y legítima en la sociedad.

ODS 9: Industria, Innovación e Infraestructura

El objetivo número 9 promueve la construcción de infraestructuras resilientes, la industrialización inclusiva y sostenible, y el fomento de la innovación. Este objetivo reconoce que el desarrollo económico y el avance tecnológico son necesarios para resolver problemas globales y mejorar la calidad de vida. La atención se centra en desarrollar una infraestructura confiable y sostenible, promover una industrialización que beneficie a todos y promover la innovación como motor del desarrollo.

ODS 11: Ciudades y Comunidades Sostenibles

Este objetivo busca hacer que las ciudades y los asentamientos humanos sean inclusivos, seguros, resilientes y sostenibles. Con el rápido desarrollo de las ciudades, este objetivo responde a la necesidad de una mejor planificación y gestión urbana para crear espacios que garanticen una alta calidad de vida a las personas. Promueve el acceso a viviendas asequibles y servicios básicos, transporte público sostenible, reduce el impacto ambiental de las ciudades y protege el patrimonio cultural y natural.

	Nada apreciable	Poco apreciable	Efecto apreciable	Notablemente apreciable	Muy apreciable
1. Fin de la pobreza	X				
2. Hambre cero	X				
3. Salud y bienestar	X				
4. Educación de calidad					X
5. Igualdad de género		X			
6. Agua limpia y saneamiento	X				
7. Energía asequible y no contaminante	X				
8. Trabajo decente y crecimiento económico			X		
9. Industria, innovación e infraestructura				X	
10. Reducción de las desigualdades	X				
11. Ciudades y comunidades sostenibles				X	
12. Producción y consumo responsables		X			
13. Acción por el clima	X				
14. Vida submarina	X				
15. Vida de ecosistemas terrestres	X				
16. Paz, justicia e instituciones sólidas	X				
17. Alianzas para lograr los objetivos	X				

Tabla C.1: Impacto de los ODS en el proyecto

Referencias

A3 Association for Advancing Automation (1979). A3 association for advancing automation. Automate. https://www.automate.org/.

AENOR (2024). Aenor: la marca que crea confianza entre personas y empresas. https://www.aenor.com/.

Agencia Estatal Boletín Oficial del Estado (2024). Boe.es - agencia estatal boletín oficial del estado. https://www.boe.es/.

Aula21 (2023). Servomotores: Héroes silenciosos de la tecnología moderna. https://www.cursosaula21.com/que-es-un-servomotor/.

Bohórquez, G. R. B. (2003). Modelamiento cinemático y odométrico de robots móviles. *UAB*.

Bolton, W. (2017). *Mecatrónica*. Alpha Editorial.

Garibay Pascual, Jonathan Ruiz (2006). Robótica: Estado del arte. *Universidad de Deuston. Número. Fecha*, 54.

I+D Electrónica (2024). Arduino Due. I+D Electrónica - Didácticas Electrónica. https://www.didacticaselectronicas.com/index.php/sistemas-de-desarrollo/arduino/arduino-2/a000062-detail.

Mathworks (2024). Simscape multibody — blocks. https://es.mathworks.com/help/sm/referencelist.html?type=block&listtype=cat&category=index&blocktype=all&capability=&startrelease=&endrelease=&s_tid=CRUX_topnav.

Ministerio de Derechos Sociales, Consumo y Agenda 2030 (2024). Conoce la agenda. https://www.mdsocialesa2030.gob.es/agenda2030/conoce_la_agenda.htm.

Naciones Unidas (2015). Objetivos de desarrollo sostenible. https://www.undp.org/es/sustainable-development-goals.

Naylamp Mechatronics (2024a). Driver puente h l298n 2a. Naylamp Mechatronics - Perú. https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html.

Naylamp Mechatronics (2024b). Módulo mpu6050: Acelerómetro, giroscopio i2c. Naylamp Mechatronics - Perú. https://naylampmechatronics.com/sensores-posicion-inerciales-gps/33-modulo-mpu6050-acelerometro-giroscopio-i2c.html.

Pololu (2024a). Aluminum mounting hub for 1/4? shaft m3 pololu - fijación de la rueda. TME - Elektroniikka komponentit. https://www.tme.eu/es/details/pololu-1994/accesorios-para-robotica-y-rc/pololu/aluminum-mounting-hub-for-1-4-shaft-m3/.

Pololu (2024b). Pololu - 30:1 metal gearmotor 37dx52l mm 12v (piñón helicoidal). Pololu. https://www.pololu.com/product/4742/specs.

Pololu (2024c). Stamped aluminum l-bracket pair for 37d pololu - montaje. TME - Elektroniikka komponentit. https://www.tme.eu/es/details/pololu-1084/accesorios-para-micromotores/pololu/stamped-aluminum-l-bracket-pair-for-37d/.

Samaniego, J. F. (2023). El futuro de la robótica. *UOC*. https://www.uoc.edu/es/news/2023/150-theker-robotica.

SBG Systems (2024). Imu - unidad de medición inercial. SBG Systems. https://www.sbg-systems.com/es/unidad-de-medicion-inercial-sensor-imu/.

Solorobotica (2014). Motores de corriente continua. https://solorobotica.blogspot.com/2011/08/motores-de-corriente-continua.html. Consultado en el sitio web *Solo Robótica*.

stepperonline (2024). ¿qué es un motor paso a paso? https://www.omc-stepperonline.com/es/support/que-es-un-motor-paso-a-paso. Consultado en el sitio web *OMC Stepper Online*.

Sundin, C. (2012). Modelamiento cinematico.pmd. CUT, Department of Signals and Systems. https://odr.chalmers.se/server/api/core/bitstreams/3e7cc529-0f70-4785-b99d-13db701904fb/content.

Universidad Complutense de Madrid (2023). 76-2013-07-11-05_lissajous_figures.pdf. https://www.ucm.es/data/cont/docs/76-2013-07-11-05_Lissajous_figures.pdf.

Vitaliti, J. A. (2023). Sistemas de control - estabilización mecánica del péndulo invertido. https://www.youtube.com/watch?v=akvYwt0bMq4.

Víctor Esteban Falconi Loja (2015). Diseño e implementación en pcb de un robot autobalanceado mediante arduino con módulo inalámbrico.