Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

ESTUDIO Y DESARROLLO DE SISTEMA DE CONTROL DE EQUILIBRIO PARA PLATAFORMA LEGO-ANYWAY

Alumna: Rodriguez Arechabala, Olatz

Director: Irigoyen Gordo, Eloy

Curso: 2018-2019

Fecha: 22 de julio de 2019

Resumen trilingüe

En este documento se expone la metodología seguida para el diseño de control de equilibrio de una plataforma LEGO-ANYWAY, usando los programas MATLAB y Simulink. Para ello, se ha estudiado el modelo matemático del sistema y se ha diseñado un controlador de tipo LQR. También se ha incluido un análisis de funcionalidad los diferentes sensores que contiene el kit LEGO MINDSTORMS.

La solución propuesta no es del todo satisfactoria pues el robot solo se mantiene en equilibrio unos segundos, pero abre paso a estudios futuros.

Dokumentu honetan LEGO-ANYWAY plataformaren oreka kontrol bat diseinatzeko metodologia aurkezten da, MATLAB eta Simulink programak erabilita. Horretarako, sistemaren eredu matematikoa ikasi da eta LQR motako kontrolagailu bat diseinatu da. Halaber, LEGO MINDSTORMS kitak barne hartzen dituen sentsore ezberdinen erabilgarritasunaren azterketa ere sartu da.

Proposatutako ebazpena ez da guztiz gogobetekoa, izan ere robota segundo batzuk baino ez da mantentzen orekan, baina geroko ikasketetara bidea zabaltzen du.

This document presents the methodology followed for the design of control of balance of a LEGO-ANYWAY platform, using the MATLAB and Simulink programs. For this, it has been studied the mathematical model of the system and it has been designed a LQR type controller. It has also been included an analysis of the functionality of the different sensor that the LEGO MINDSTORMS kit contains.

The proposed solution is not entirely satisfactory because the robot only stays in balance for a few seconds, but opens the way to future studies.

Palabras clave

LEGO MINDSTORMS EV3, MATLAB, Simulink, programación, robot, equilibrio

Índice

Índice de figuras

Índice de tablas

Índice de gráficos

1. Memoria

1.1. Introducción

Este documento recoge los aspectos necesarios para el desarrollo de un sistema de control de equilibrio para la plataforma LEGO-ANYWAY. Se ha estructurado en cuatro partes.

La primera parte, presenta el contexto del trabajo, sus objetivos y beneficios. También se incluyen las prácticas realizadas con diferentes sensores para comprobar su funcionalidad. En este apartado se puede encontrar el diseño detallado del sistema de control. Se explica el sistema matemático del robot y se presenta el controlador escogido y sus resultados.

En la segunda parte, se recogen las conclusiones adquiridas de estos resultados y las capacidades de mejora.

Posteriormente, se explica la metodología seguida en el trabajo. Exponiendo los pasos seguidos con la ayuda de un diagrama de Gantt.

Finalmente, este informe incluye el presupuesto ejecutado en la elaboración de este trabajo. También se incluyen dos anexos, uno con el código elaborado en MATLAB y otro con el montaje del robot utilizado.

1.2. Objetivo y alcance del trabajo

El objetivo principal de este Trabajo Final de Grado es diseñar un control que sea capaz de mantener en equilibrio, sin cambiar de posición, un robot LEGO-ANYWAY mediante el uso de MATLAB y Simulink.

Para ello, se estudiará el funcionamiento de los diferentes sensores que incluye el robot y su comunicación con MATLAB y Simulink.

Se analizará el sistema físico y matemático del robot para comprender su movimiento.

Se elegirá el controlador más adecuado para esta aplicación y se calibrará para que dé los mejores resultados de equilibrio.

1.3. Contexto

1.3.1. Robot de equilibrio sobre 2 ruedas

El robot LEGO-ANYWAY consiste en un robot de equilibrio sobre 2 ruedas, es decir, es un robot que es capaz de mantenerse levantado en equilibrio por sí solo sobre sus dos ruedas. El proceso de equilibrio se denomina típicamente control de estabilidad. Las dos ruedas están situadas debajo de la base y permiten que el chasis del robot mantenga una posición vertical moviéndose en la dirección de inclinación, ya sea hacia adelante o hacia atrás, en un intento de mantener el centro de la masa sobre los ejes de las ruedas.

Figura 1. Segway i2 SE
(Fuente: https://www.ninebotus.com/segway-i2-se/)

Este tipo de robot proporciona un problema desafiante y ha resultado en el desarrollo de muchos diseños útiles e interesantes. Uno de estos robots de dos ruedas que se ha convertido en un éxito comercial es el Segway PT (Personal transporter, transportador personal) de Segway Inc.

El Segway PT fue creado por Dean Kamen y comercializado en 2001. Se trata de un vehículo personal motorizado de dos ruedas con una base donde apoyar los pies y un manillar que, usando el principio de equilibrio mencionado, es capaz de transportar personas sin necesidad de controles por parte del usuario. Este tipo de robots se han vuelto cada vez más populares con los años, siendo muy común verlos en lugares turísticos o como medio de transporte de la policía.

En cuestión de robots a una escala más reducida se pueden encontrar múltiples maquetas a lo largo de los años hechas con diferentes equipos. En esencia, para construir este tipo de robot, se necesitan 2 motores conectados a unas ruedas, un giróscopo para medir el ángulo del robot y alguna estructura para crear el cuerpo. Además, es necesario que incluya o pueda conectarse a algún tipo de sistema de control.

Varias empresas han comercializado sets que permiten construir tales robots. Un ejemplo es el kit ELEGOO Arduino Uno que incluye las piezas necesarias para construir un robot y luego controlarlo mediante Arduino.

El robot usado en este trabajo es un robot construido con el set de robótica LEGO MINDSTORMS. Este set combina la fácil manejabilidad de las piezas LEGO con un sistema de control que permite programar en diversos lenguajes y plataformas, desde la suya propia hasta otras más usadas en programación como LabVIEW o RobotC.

1.3.2. LEGO MINDSTORMS

Este set salió al mercado por primera vez en 1998 como una colaboración entre el MIT y Lego. Hasta hoy en día, ha habido tres generaciones de bloques programables de LEGO MINDSTORMS.

- Robotics Invention System

La primera generación, que en castellano se traduciría como Sistema de Invención robótica, fue lanzada en 1998. Su robot programable se llamaba RCX (Robotic Command eXplorer, explorador de orden robótico). Podía programarse usando el código RCX o ROBOLAB, el cual estaba basado en LabVIEW. El RCX tenía un procesador de 16 MHz con 32K de RAM. Disponía de 3 entradas y tres salidas para sensores. El kit también incluía dos motores, dos sensores táctiles y un sensor de luz.

Figura 2. Robotics Invention System

(Fuente: https://www.bricklink.com/v2/catalog/catalogitem.page?S=3804-1#T=S&O={%22iconly%22:0})

- LEGO MINDSTORMS NXT

Lanzada en 2006, fue una versión mejorada del bloque RCX que contaba con un procesador de 48 MHz con 64 KB de RAM. Esto dotaba al Lego de una mayor capacidad de ejecución de programas. El número de salidas se mantenía y las entradas aumentaron en uno, pero con conectores distintos a los del RCX, lo que impedía su uso. El kit NXT contenía 3 servomotores, un sensor de luz, uno de sonido, uno táctil y uno de distancia. El software NXT era NXT-G, y el kit venía con ROBOLAB.

Para comunicarse con un PC poseía un puerto USB 2.0 y también incluía una interfaz Bluetooth que servía tanto para comunicarse con el computador como para comunicarse con otros robots que se encontraran cerca.

Figura 3. LEGO MINDSTORMS NXT
(Fuente: https://www.fabtolab.com/LEGO-mindstorms-nxt-9797)

- LEGO MINDSTROMS EV3

El más moderno de los bloques, hasta ahora, salió al mercado en 2013. Opera con Linux y tiene un procesador de 300MHz con 64MB de memoria RAM más 16MB de memoria Flash. Este modelo pasa a tener cuatro salidas para motores por lo que cuenta con cuatro entradas para sensores y cuatro salidas para actuadores, con la misma conexión que el modelo NXT, lo que hace que todos los sensores, motores y piezas del modelo anterior sean compatibles con la nueva versión.

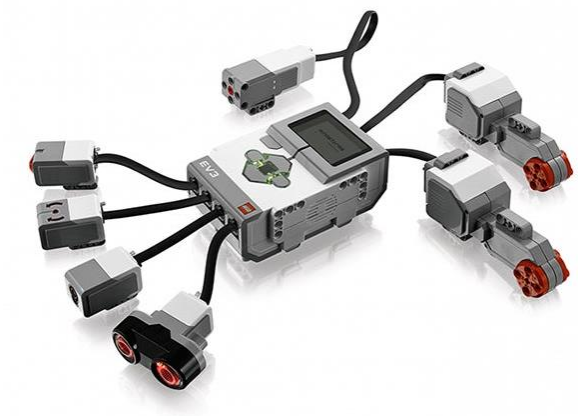


Figura 4. LEGO MINDSTORMS EV3
(Fuente: https://www.myrobotcenter.eu/en/lego-mindstorms-ev3-basis-set-45544)

El set opera una versión de LabVIEW y contiene dos motores grandes, un motor mediano, 2 sensores táctiles, un sensor de color, un sensor giroscópico y un sensor ultrasónico.

En cuanto a conectividad, este modelo cuenta con un puerto mini USB para su conexión directa con un computador y un puerto USB en el que se puede instalar una antena Wi-Fi para establecer la conectividad mediante esta tecnología y conexión Bluetooth.

A continuación, una tabla que compara las características de las diferentes generaciones:

	RCX	NXT	EV3
Procesador	Hitachi H8/3292 10-16 MHz 16 KB ROM 32 KB RAM	Atmel-32 Bit-ARM7 48MHz 256 KB flash 64 KB RAM	ARM 9 300 MHz 16 MB flash 64 MB RAM
Puertos	3 para motores 3 para sensores	3 para motores 4 para sensores	4 para motores 4 para sensores
Comunicación	Puerto IR utilizado para comunicación con el equipo de cómputo y con otro RCX	USB 12 Mbps Bluetooth	Bluetooth v2.1 Wi-Fi mediante el puerto USB
Almacenamiento extra	No	No	Ranura Micro SD

Tabla 1. Comparación entre diferentes generaciones de LEGO MINDSTORMS

1.3.3. Hardware utilizado

En este trabajo el bloque utilizado es el de la última generación y, por ende, el más reciente. El hardware está formado por los siguientes componentes:

- Ladrillo EV3

Sirve como centro de control y eléctrica del motor. Es el cerebro del motor y dota de las características expuestas anteriormente. A él están conectados todos los sensores y motores mediante cables de conexión.

Figura 5. Ladrillo EV3

(Fuente: https://shop.lego.com/en-US/product/EV3-Intelligent-Brick-45500)

- Servomotores:

El robot cuenta con dos servomotores conectados a las ruedas que permiten su movimiento. Estos motores funcionan a 160–170 rpm, con un torque de rotación de 20 Ncm y un torque de rotor bloqueado de 40 Ncm. Tiene un Sensor de rotación incorporado con resolución de 1 grado para un control preciso. Con esta resolución se puede, por ejemplo, determinar la posición del robot.

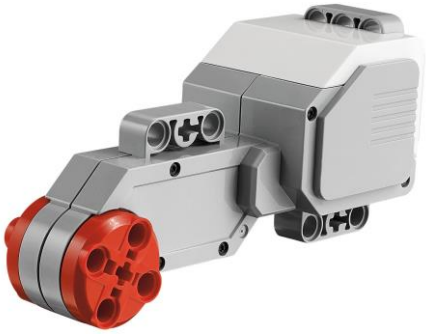


Figura 6. Servomotor grande EV3

(Fuente: https://shop.lego.com/en-US/product/EV3-Large-Servo-Motor-45502)

- Sensor giroscópico

Un giróscopo es un sólido rígido que se mueve manteniendo uno de sus puntos. Se utiliza para medir la orientación en el espacio de algún aparato o vehículo. Su utilización en el sistema mecánico se basa en el efecto conocido como efecto giroscópico. Consiste básicamente en que el movimiento de rotación del sólido tiende a oponerse por inercia a cualquier variación del mismo.

El giróscopo de LEGO MINDSTORMS mide la velocidad angular alrededor de un eje en grados por segundo. Su precisión es de ±3 grados y tiene un output máximo de 440°/s. Además de la velocidad, también indica el sentido de giro por lo que su rango de medida es de ±360°/s.

El robot utilizado dispone de dos sensores giroscópicos para mayor precisión de medida.

Figura 7. Sensor giroscópico EV3

(Fuente: https://shop.lego.com/en-US/product/EV3-Gyro-Sensor-45505)

- Sensor de color

Es un sensor digital que puede detectar el color o la intensidad de la luz que ingresa por la pequeña ventana de la cara del sensor. Este sensor puede utilizarse en tres modos diferentes: Modo color, modo intensidad de la luz reflejada y modo intensidad de la luz ambiental.

En modo color puede distinguir entre siete colores: Los tres primarios (rojo, azul y amarillo) y el secundario verde, además de negro, blanco y marrón. También puede detectar sin color, lo que significa que puede programarse para actuar cuando no se detecta ninguno de los 7 colores, haciendo que haya 8 opciones posibles. Cada color está asignado a un número diferente, del 0 al 7.

Color	Sin color	Negro	Azul	Verde	Amarillo	Rojo	Blanco	Marrón
Número	0	1	2	3	4	5	6	7

Tabla 2. Asignación de colores a números del sensor de color en modo color

En modo intensidad de luz reflejada el sensor es capaz de medir la intensidad de la luz reflejada por las superficies cercanas. El sensor utiliza una escala de 0 (muy oscuro) a 100 (muy luminoso).

En modo intensidad de luz ambiental mide la intensidad de la luz que entra por la ventana desde su entorno.

La razón de muestreo del Sensor de color es 1 kHz/s.

Figura 8. Sensor de color EV3

(Fuente: https://shop.lego.com/en-US/product/EV3-Color-Sensor-45506)

- Sensor de ultrasonidos

Este sensor genera ondas de sonido de alta frecuencia y mediante los ecos de estas en objetos, es capaz de determinar la distancia hasta ellos.

Tiene un rango de medida de 1 a 250 cm, con una precisión de ± 1 cm.

Figura 9. Sensor de ultrasonidos EV3

(Fuente: https://shop.lego.com/en-US/product/EV3-Ultrasonic-Sensor-45504)

- Cables de conexión y piezas de construcción

Para conectar los sensores y motores se utilizan cables de conexión de diferentes tamaños.

Además, para poder hacer un conjunto con todos los elementos y crear el cuerpo del robot se utilizan diferentes piezas de construcción incluidas en el set de LEGO MINDSTORMS EV3.

Figura 10. Kit LEGO MINDSTORMS EV3

(Fuente: https://www.juguetronica.com/lego-mindstorms-ev3)

1.3.4. Software utilizado

LEGO MINDSTORMS permite programar el robot con diversos programas. En primer lugar, tiene su propio lenguaje, con él se puede programar desde el propio ladrillo o desde el ordenador con un programa descargable desde la página de LEGO. Este lenguaje puede ser muy útil para crear programas simples e ir familiarizándose con el robot, pero no es viable para problemas más complejos.

Un programa compatible con LEGO MINDSTORMS es, por ejemplo, LabVIEW, una plataforma para diseñar sistemas con un lenguaje de programación visual gráfico.

En este trabajo se ha utilizado la plataforma MATLAB junto con Simulink.

MATLABes una herramienta de software matemático desarrollado por Mathworks que ofrece un entorno de desarrollo integrado(IDE) con unlenguaje de programaciónpropio (lenguaje M).

Simulink, desarrollado también por MathWorks, es un entorno de programación gráfica para modelar, simular y analizar sistemas dinámicos.

La combinación de ambos resulta muy adecuada para programar un sistema como este ya que con las funciones de MATLAB y estilo gráfico de Simulink se pueden llegar a diseñar programas de una complejidad avanzada de manera sencilla.

MATLAB cuenta con un paquete de apoyo descargable desde la página web de Mathworks llamado *MATLABSupport Package for LEGO MINDSTORMSEV3 Hardware* que incluye diversas funciones con las que se puede controlar los motores e interactuar con los diferentes sensores. Se puede conectar el robot con MATLAB mediante conexión USB, Wi-Fi o Bluetooth.

Simulink dispone de otro paquete de apoyo para LEGO MINDSTORMS EV3 que permite programar y ejecutar algoritmos en tales robots. El paquete incluye una librería de bloques de simulink para configurar y acceder a los sensores, actuadores e interfaces de comunicación del robot.

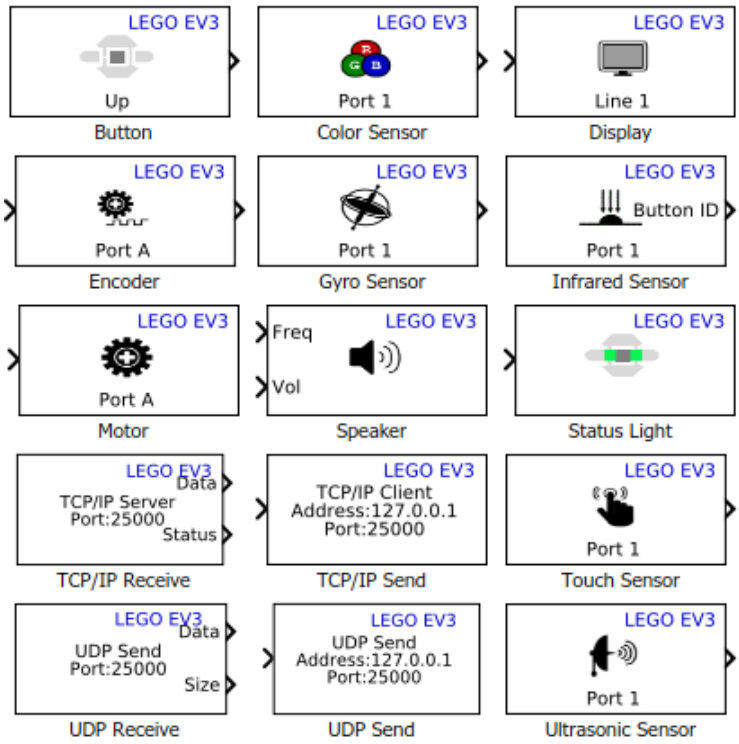


Figura 11. Liberia de Simulink Support Package for LEGO MINDSTORMS EV3 Hardware

Con este paquete, se pueden desarrollar algoritmos en Simulink, simularlos para verificar que funcionan según lo previsto y descargar el algoritmo completado para la ejecución independiente en el dispositivo (mediante conexión USB, Bluetooth, Wi-Fi o Ethernet). También tiene la capacidad de sintonizar parámetros en vivo desde su modelo de Simulink mientras el algoritmo se ejecuta en el bloque EV3.

1.4. Beneficios que aporta el trabajo

Los benéficos que aporta el trabajo se han dividido en dos categorías:

- Beneficios sociales: Este trabajo ayuda a la comprensión del funcionamiento de un robot sobre dos ruedas y lo necesario para mantenerlo en equilibrio. Asimismo, también aporta la base para diseñar futuros proyectos donde es necesario que el robot se encuentre en equilibrio.
- Beneficios personales: Con la realización de este trabajo se adquiere conocimiento de la plataforma LEGO-MINDSTORMS y se amplían los conocimientos de MATLAB y Simulink adquiridos en otras asignaturas del grado. También se han refrescado los conocimientos de mecánica en el estudio matemático del sistema. Además, también se adquiere capacidad de resolución de problemas al hacer frente a los diferentes retos surgidos durante el desarrollo del trabajo.

1.5. Descripción de requerimientos

Para llevar a cabo este proyecto se requiere de conocimientos previos de MATLAB y Simulink, además de conocimientos de mecánica para comprender el sistema físico del robot.

El desarrollo de este trabajo se ha llevado a cabo en dos partes. Primero, se han hecho unas prácticas con los diferentes sensores incluidos en el robot y luego se ha procedido a diseñar un controlador de equilibrio mediante el análisis de la maqueta y distintos tipos de controladores.

1.6. Practicas con sensores

Para poder analizar la funcionalidad de los sensores de LEGO MINDSTORMS se han llevado a cabo distintas prácticas usando los bloques de Simulink del paquete de apoyo de LEGO MINDSTORMS EV3.

1.6.1. Sensor de color

Como ya se ha explicado anteriormente el sensor de color tiene diferentes modos.

Para probar el modo de color se ha utilizado un simple programa de Simulink en el que el sensor lee el color y lo muestra en la pantalla del ladrillo (fig. 12).

Figura 12. Programa de Simulink de lectura de color

En la figura 13, se muestran los colores probados en esta práctica y junto a ellos los colores que el sensor ha reconocido.

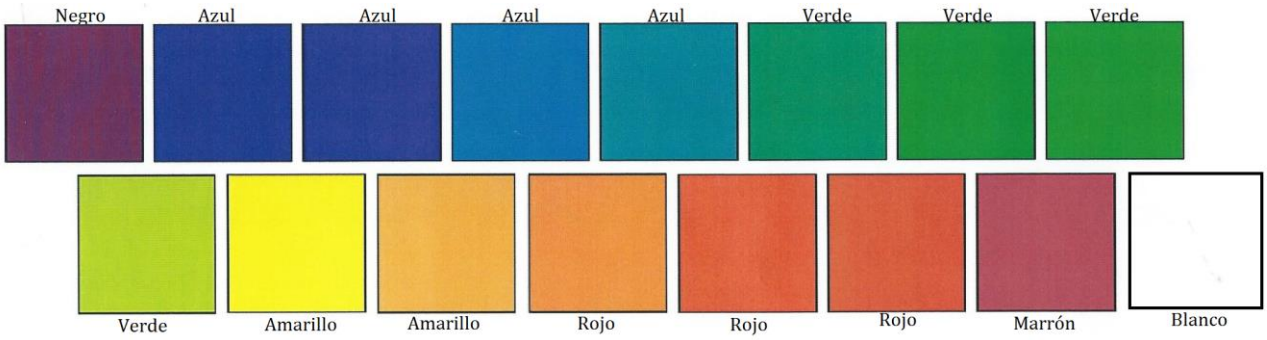


Figura 13. Colores probados con el sensor de color y sus resultados

Como se puede observar es capaz de leer correctamente los colores que tiene en el sistema como el azul o el verde. Sin embargo, los colores más ambiguos pueden dar resultados distintos como las diferentes tonalidades del naranja o el ultimo color rojizo que lo reconoce como marrón.

Usando el mismo programa de Simulink se puede ajustar el bloque de *Color Sensor* para que mida la intensidad de la luz reflejada.

En este modo se puede observar que en el color negro muestra 1 o 2 y mediante se va moviendo hacia el blanco va subiendo hasta llegar hasta 30 aproximadamente.

Ambas pruebas se han hecho con el sensor a aproximadamente 1 cm de distancia de un folio sobre una mesa.

1.6.2. Sensor giroscópico

Ya que el sensor giroscópico mide la velocidad angular y no el ángulo, se ha creado un programa en simulink que deriva este valor y lo muestra en el ladrillo para comprobar como de efectivo es.

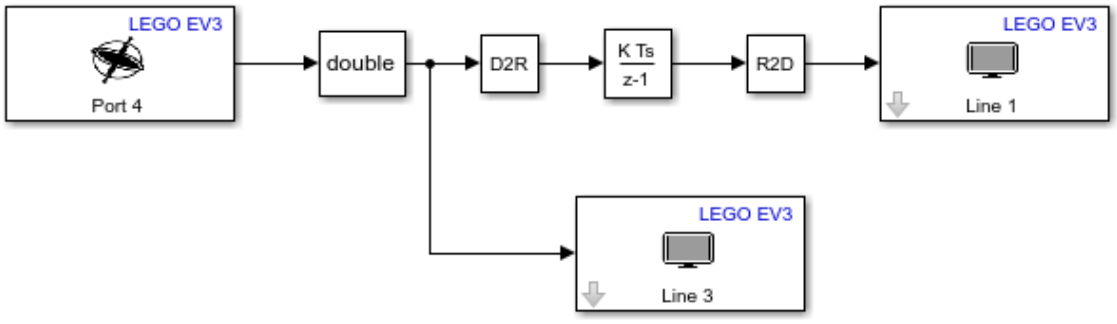


Figura 14. Programa de Simulink de cálculo del ángulo

La prueba realizada consiste en, teniendo el robot depositado sobre la mesa en horizontal, levantarlo hasta una posición aproximada de 90º, es decir, perpendicular a la mesa. El valor que se muestra se encuentra alrededor de los 90º, entre 89 y 93 en distintas pruebas, que, teniendo en cuanta la inexactitud que genera el giro manual, se puede considerar preciso. Sin embargo, se ha observado que algunas veces el valor de la velocidad angular es no nulo desde el principio a pesar de estar en reposo sobre una mesa. Debido a esto el ángulo medido va aumentando, aunque el robot no se mueva. Al hacer la prueba en estas condiciones los resultados no son favorables yendo cada vez a peor con el tiempo. De esta manera, se ve la necesidad de corregir este problema en caso de utilizar este sensor como método de medida de ángulo.

Como el robot utilizado dispone de dos sensores giroscópicos también se ha hecho una prueba para comparar que ambos midan lo mismo. Es decir, su resta debería dar 0 en todo momento si se mueven al unísono.

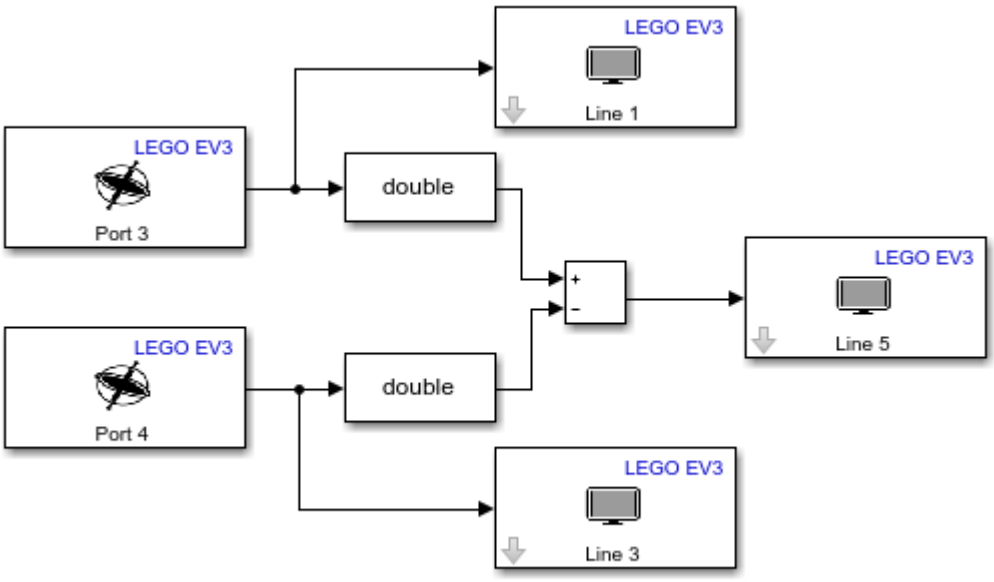


Figura 15. Programa de Simulink de comparación de ángulos

Se puede observar que el valor de la resta oscila entre -1 y 2. Siendo la diferencia más grande cuando se gira a velocidades más altas. Esto es lógico porque todos los sensores van a tener siempre cierto retraso desde que se produce el cambio hasta que se muestra en la pantalla y cada sensor va a tener el suyo. Además, esta prueba solo va a ser concluyente cuando ninguno de los dos sensores comienza con el error explicado anteriormente.

1.6.3. Sensor de ultrasonidos

En esta prueba se han marcado diferentes distancias hasta un obstáculo y se ha medido el valor que recoge el sensor.

Figura 16. Programa de Simulink de lectura de distancia

En la siguiente tabla se recogen las diferentes distancias medidas:

Distancia real	15 cm	20 cm	25 cm
Distancia del sensor	15 cm	19 cm	25 cm

Tabla 3. Distancias medidas con el sensor de ultrasonidos

Aproximadamente, todas las distancias dieron la medida correcta, aunque era bastante difícil comprobar si en el sensor se estaba poniendo en el lugar correcto.

1.6.4. Conclusiones

Habiendo analizado el comportamiento de todos los sensores se puede concluir que todos son bastante fiables.

El sensor de ultrasonidos parece estar correctamente calibrado y es bastante preciso.

El sensor de color funciona con relativa fiabilidad. Si se quiere crear un programa en el que el robot responda de alguna manera determinada ante un color u otro se recomienda no utilizar colores demasiado parecidos o de una tonalidad ambigua para acciones diferentes ya que podría no ser capaz de diferenciarlos correctamente.

Por último, el sensor que más problemas ha dado ha sido el sensor giroscópico. Como ya se ha explicado tiene un fallo recurrente al principio de los programas, haciendo que su medición no sea correcta en ningún momento. La resolución de este problema se explicará posteriormente en el apartado 1.7.2.1.

1.7. Control de equilibrio

1.7.1. Conceptos teóricos

1.7.1.1. Modelo matemático del sistema

Como ya se ha explicado anteriormente, el robot LEGO-ANYWAY consiste en un robot que ha de mantenerse en equilibrio sobre 2 ruedas. Ese sistema se asemeja al de un péndulo invertido en el que una masa está suspendida sobre un carro (fig. 17). Se trata de un sistema no lineal con 2 grados de libertad. En nuestro caso el sistema se complica ya que cada rueda funciona con un motor diferente creando un ángulo de giro distinto en cada motor, haciendo que haya 3 grados de libertad.

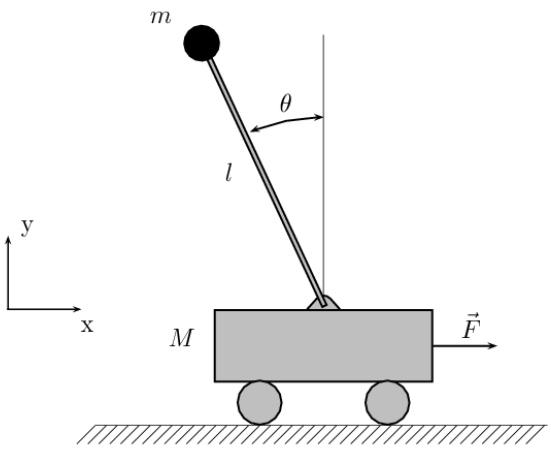


Figura 17. Esquema de pendulo invertido

(Fuente: http://matlabyoctave.appspot.com/invpend/invpend.html)

El robot tiene una forma geométrica compleja, pero se podría decir que se asemeja a un prisma rectangular con 2 discos como ruedas (fig. 18). La masa del prisma se compone de la suma de las masas de todos los elementos del robot exceptuando las ruedas. Su altura es el doble de su centro de gravedad y su anchura es la distancia entre ruedas. Los discos tienen las mismas características que las ruedas, con su masa y su radio.

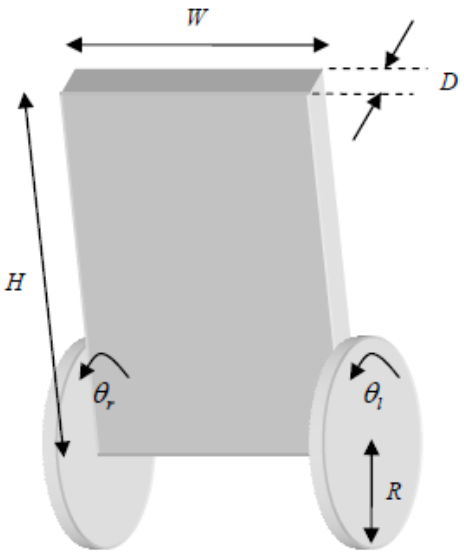


Figura 18. Esquema de prisma rectangular con discos como ruedas

Para simplificar los cálculos, se puede intercambiar el prisma rectangular por una masa puntual situada en la posición del centro de gravedad de la misma conectada a las ruedas mediante una barra sin masa. En la figura 19 se muestra en esquema de este modelo de péndulo invertido.

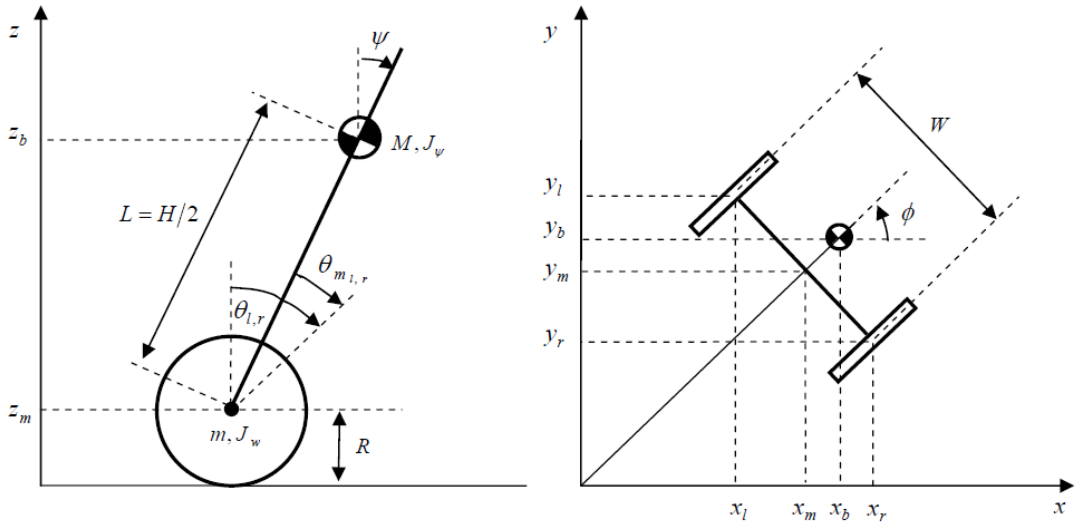


Figura 19. Esquema péndulo invertido sobre 2 ruedas

Siendo este el significado de sus variables:

L: Altura del centro de gravedad del robot

R: Radio de las ruedas

W: distancia entre ruedas

M: Masa del robot sin ruedas

m: Masa de una rueda

ψ: Ángulo del péndulo

$\theta_{m_{l,r}}$: Ángulo del motor (l izquierda/r derecha)

$\theta_{l,r}$: Ángulo de las ruedas

Adaptando estas variables al robot utilizado, ψ va a ser el ángulo obtenido derivando la salida del sensor giroscópico y $\theta_{m_{l,r}}$ es la salida de los encoders de los motores. Y en función de ellas se pueden obtener las variables $\theta_{l,r}$ y ϕ,

$$\theta_{l,r} = \theta_{m_{l,r}} + \psi$$

$$\phi = \frac{R}{W}(\theta_r - \theta_l)$$

Y como la diferencia entre los ángulos θ_r y θ_l ya se recoge en el ángulo ϕ, se pueden agrupar de la siguiente manera:

$$\theta = \frac{\theta_l + \theta_r}{2}$$

Para calcular las ecuaciones diferenciales del sistema se va a utilizar el método de Lagrange.

Se empieza por adquirir las coordenadas de cada elemento:

- Carro (suma de las 2 ruedas): $(x_m, y_m, z_m) = (\int \dot{x}_m dt\,, \int \dot{y}_m dt\,, R)$

  Donde las velocidades $\dot{x}_m$ y $\dot{y}_m$ son: $(\dot{x}_m, \dot{y}_m) = (R\dot{\theta}\cos\phi\,, R\dot{\theta}\sin\phi)$

- Rueda izquierda: $(x_l, y_l, z_l) = (x_m - \frac{W}{2}\sin\phi\,, y_m + \frac{W}{2}\cos\phi\,, z_m)$

- Rueda derecha: $(x_r, y_r, z_r) = (x_m + \frac{W}{2}\sin\phi\,, y_m - \frac{W}{2}\cos\phi\,, z_m)$

- Masa puntual: $(x_b, y_b, z_b) = (x_m + L\sin\psi\cos\phi\,, y_m + L\sin\psi\sin\phi\,, z_m)$

Con estas coordenadas se pueden calcular las energías cinéticas traslacional (T_1) y rotacional (T_2) y la energía potencial (V).

$$T_1 = \frac{1}{2}Mv_G^2 = \frac{1}{2}m\left(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2\right) + \frac{1}{2}m(\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2) + \frac{1}{2}M\left(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2\right) \quad (1)$$

$$\begin{aligned} T_2 &= \frac{1}{2}\langle\omega\rangle^T[J_G]\langle\omega\rangle \\ &= \frac{1}{2}J_W\dot{\theta_l}^2 + \frac{1}{2}J_W\dot{\theta_r}^2 + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_l - \dot{\psi}\right)^2 \\ &+ \frac{1}{2}n^2J_m\left(\dot{\theta}_r - \dot{\psi}\right)^2 \end{aligned} \quad (2)$$

$$V = mgH_G = mgz_l + mgz_r + Mgz_b \quad (3)$$

La función lagrangiana es la función resultante de diferencia entre las energías cinética y potencial, por lo tanto, sustituyendo las coordenadas se obtiene esta ecuación:

$$\begin{aligned} L &= T_1 + T_2 - V \\ &= \frac{1}{2}m\left[\left(R\dot{\theta}\cos\phi - \frac{w}{2}\dot{\phi}\cos\phi\right)^2 + \left(R\dot{\theta}\sin\phi - \frac{w}{2}\dot{\phi}\sin\phi\right)^2\right] \\ &+ \frac{1}{2}m\left[\left(R\dot{\theta}\cos\phi + \frac{w}{2}\dot{\phi}\cos\phi\right)^2 + \left(R\dot{\theta}\sin\phi + \frac{w}{2}\dot{\phi}\sin\phi\right)^2\right] \\ &+ \frac{1}{2}M\left[\left(R\dot{\theta}\cos\phi + L\dot{\psi}\cos\psi\cos\phi - L\dot{\psi}\sin\psi\sin\phi\right)^2\right. \\ &\left.+ \left(R\dot{\theta}\sin\phi + L\dot{\psi}\cos\psi\sin\phi - L\dot{\psi}\sin\psi\cos\phi\right)^2\right] + J_W\dot{\theta}^2 \\ &+ \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 + n^2J_m\left(\dot{\theta} - \dot{\psi}\right)^2 - 2mgR \\ &- Mg(R + L\cos\psi) \end{aligned} \quad (4)$$

La ecuación de Lagrange tiene la siguiente forma para sistemas mixtos de fuerzas conservativas y no conservativas:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = F_{q_j} \qquad j = 1,2,\dots,n \quad (5)$$

Donde q_j son las coordenadas generalizadas, en este caso : $\begin{cases}\theta \\ \psi \\ \phi\end{cases}$

Haciendo todos los cálculos pertinentes, estas son las tres fuerzas que salen:

$$\begin{aligned} F_\theta &= [(2m + M)R^2 + 2J_W + 2n^2J_m]\ddot{\theta} + (MLR\cos\psi - 2n^2J_m)\ddot{\psi} \\ &- MLR\dot{\psi}^2\sin\psi \end{aligned} \quad (6)$$

$$\begin{aligned} F_\psi &= (MLR\cos\psi - 2n^2J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\sin\psi \\ &- ML^2\dot{\phi}^2\sin\psi\cos\psi \end{aligned} \quad (7)$$

$$F_\phi = \left[\frac{1}{2}\mathrm{mW}^2 + \mathrm{J}_\Phi + \frac{w^2}{2R}(J_w + n^2 J_m) + ML^2 \sin^2\psi\right]\ddot{\phi} + 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi \quad (8)$$

Estas ecuaciones se pueden linealizar considerando el límite de los ángulos $\cos\alpha \to 1$ y $\sin\alpha \to \alpha$, y anulando cualquier coordenada que sea de segundo o mayor orden. Al final estas son las ecuaciones finales,

$$F_\theta = [(2m + M)R^2 + 2J_W + 2n^2 J_m]\ddot{\theta} + (MLR\cos\psi - 2n^2 J_m)\ddot{\psi} \quad (9)$$

$$F_\psi = (MLR - 2n^2 J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} - \ddot{M}gL\psi \quad (10)$$

$$F_\phi = \left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}(J_W + n^2 J_m)\right]\ddot{\phi} \quad (11)$$

Estas fuerzas representan las fuerzas generalizadas que sufre el robot. Si se analizan esas fuerzas teniendo en cuenta el torque y la fricción viscosa del motor, las fuerzas generalizadas son:

$$F_l = nK_t i_l + f_m(\dot{\psi} - \dot{\theta}_l) - f_w\dot{\theta}_l \quad (12)$$

$$F_r = nK_t i_r + f_m(\dot{\psi} - \dot{\theta}_r) - f_w\dot{\theta}_r \quad (13)$$

$$F_\psi = -nK_t i_l - nK_t i_r + f_m(\dot{\psi} - \dot{\theta}_l) - f_m(\dot{\psi} - \dot{\theta}_r) \quad (14)$$

Donde $i_{r,l}$ son las intensidades de los motores derecho e izquierdo respectivamente.

Estas ecuaciones se pueden relacionar con las fuerzas de las coordenadas generalizadas de la siguiente manera:

$$(F_\theta, F_\psi, F_\phi) = \left(F_l + F_r\ ,\ F_\psi\ , \frac{W}{2R}(F_r - F_l)\right) \quad (15)$$

Como el control de los motores está basado en PWM (voltaje) y no en intensidad, es necesario que las ecuaciones obtenidas estén en función del voltaje. Se puede relacionar la intensidad y el voltaje de la siguiente manera:

$$v_{l,r} = R_m i_{l,r} + L_m \dot{i}_{l,r} - K_b(\dot{\psi} - \dot{\theta}_{l,r}) \quad (16)$$

Si se considera la inductancia del motor (L_m) insignificante y que se aproxima a 0, la intensidad finalmente es:

$$i_{l,r} = \frac{v_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r})}{R_m} \quad (17)$$

De esta manera ya se pueden representar las fuerzas en función de la tensión de los motores:

$$F_\theta = \alpha(v_l + v_r) + 2\beta\dot{\psi} - 2(\beta + f_w)\dot{\theta} \quad (18)$$

$$F_\psi = -\alpha(v_l + v_r) - 2\beta\dot{\psi} - 2\beta\dot{\theta} \quad (19)$$

$$F_\phi = \frac{W}{2R}\alpha(v_r - v_l) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \quad (20)$$

Donde α y β son:

$$\alpha = \frac{nK_t}{R_m} \quad , \quad \beta = \frac{nK_tK_b}{R_m} + f_m$$

Igualando las ecuaciones (9,10,11) y (18,19,20) se obtienen las ecuaciones diferenciales. Se puede observar que tanto las ecuaciones (9) y (18) como (10) y (19) están solo en función de θ y ψ y (11) y (20) en función de ϕ. Por lo tanto, se pueden agrupar en 2 sistemas diferentes:

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + G\begin{bmatrix}\theta\\ \psi\end{bmatrix} = H\begin{bmatrix}v_l\\ v_r\end{bmatrix} \quad (21)$$

$$E = \begin{bmatrix}(2m+M)R^2 + 2J_w + 2n^2J_m & MLR - 2n^2J_m\\ MLR - 2n^2J_m & ML^2 + J_\psi + 2n^2J_m\end{bmatrix}$$

$$F = \begin{bmatrix}2(\beta + f_w) & -2\beta\\ -2\beta & 2\beta\end{bmatrix}$$

$$G = \begin{bmatrix}0 & 0\\ 0 & -MgL\end{bmatrix}$$

$$H = \begin{bmatrix}\alpha & \alpha\\ -\alpha & -\alpha\end{bmatrix}$$

$$I\ddot{\phi} + J\dot{\phi} = K(v_r - v_l) \tag{22}$$

$$I = \frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}(J_W + n^2 J_m)$$

$$J = \frac{W^2}{2R^2}(\beta + f_w)$$

$$K = \frac{W}{2R}\alpha$$

Definiendo las variables x_1 y x_2 como estados y u como salida,

$$x_1 = [\theta \quad \psi \quad \dot{\theta} \quad \dot{\psi}]^T \quad , \quad x_2 = [\phi \quad \dot{\phi}]^T \quad , \quad u = [v_l \quad v_r]^T$$

Se obtienen 2 sistemas de espacios de estados:

$$\begin{cases} \dot{x}_1 = A_1 x_1 + B_1 u \\ y_1 = C_1 x_1 + D_1 u \end{cases} \tag{23}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -E^{-1}G & & -E^{-1}F & \end{bmatrix} \quad , \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ E^{-1}H & \end{bmatrix} \quad , \quad C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad , \quad D_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{cases} \dot{x}_1 = A_1 x_1 + B_1 u \\ y_2 = C_2 x_2 + D_2 u \end{cases} \tag{24}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -J^{-1}I \end{bmatrix} \quad , \quad B_2 = \begin{bmatrix} 0 & 0 \\ -K^{-1}I & K^{-1}I \end{bmatrix} \quad , \quad C_2 = [1 \quad 0] \quad , \quad D_2 = [0 \quad 0]$$

También se puede unir todo en un solo sistema.

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{25}$$

$$x = [\theta \quad \psi \quad \dot{\theta} \quad \dot{\psi} \quad \phi \quad \dot{\phi}]^T$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -E^{-1}G & & -E^{-1}F & & 0 & 0 \\ & & & & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -J^{-1}I \end{bmatrix} \quad , \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ & E^{-1}H \\ 0 & 0 \\ -K^{-1}I & K^{-1}I \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad , \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Los parámetros específicos del robot utilizado se pueden encontrar en el *Anexo I: Código de MATLAB*.

1.7.1.2. Controlador LQR

El regulador linear cuadrático (de sus siglas en inglés, LQR) es un método de teoría de control moderno que usa el enfoque de estados de espacio para analizar dicho sistema, como esa es la manera en la que tenemos definido nuestro sistema resulta apropiada su utilización. La idea es establecer una retroalimentación completa mediante una combinación lineal de las variables de estado.

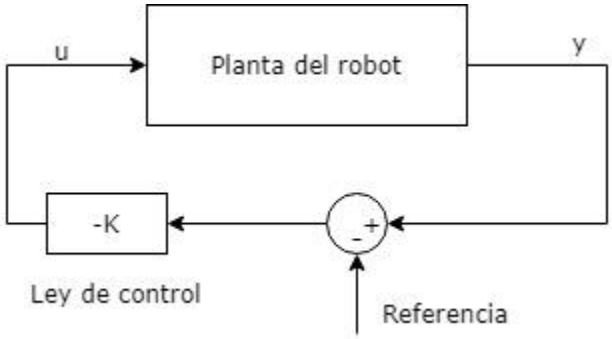


Figura 20. Esquema de controlador LQR

Esta técnica se basa en elegir una ley de control $u = \Phi(x)$ que minimice la función cuadrática de costo, que se puede representar con la siguiente ecuación,

$$J = \int_0^\infty [x(t)^T Q x(t) + u(t)^T R u(t)]\, dt \qquad (26)$$

Donde Q y R son las matrices de peso reales, simétricas y definidas positivas. A través de Q se puede elegir el peso que tiene cada estado en la función J y R indica el peso que se le quiere dar a la energía asociada con la señal de control.

La ley de control que minimiza la ecuación J siempre va a tener la forma de $u = -Kx$. Esa K va a ser definida por la siguiente ecuación:

$$K = R^{-1}B^T P \qquad (27)$$

Donde P es la única solución definida positiva de la matriz Ecuación Algebraica de Riccati:

$$A^T P + PA - PBR^{-1} B^T P + Q = 0 \quad (28)$$

MATLAB incluye una función `lqr` que minimiza la ecuación (26) con los parámetros de entrada A, B, Q y R y devuelve las marices K y P. Donde A y B son las matrices de las ecuaciones de estado y Q y R las matrices de peso. Estas últimas han de modificarse manualmente hasta conseguir los resultados deseados. Generalmente, la matriz R consiste en una matriz de identidad de las dimensiones del número de salidas. La matriz R se suele calcular haciendo $Q = C'C$, con la C de las ecuaciones de estado, y se van tuneando los valores distintos a 0 para encontrar el controlador óptimo.

Si se quiere discretizar el sistema y calcular la matriz K, MATLAB también dispone de la función `lqrd`. Añadiendo el tiempo de muestreo discretiza la planta continua (A, B) y las matrices de peso continuas (R, Q) usando un retenedor de orden cero (Zero-Order Hold, ZOH). Después calcula la matriz K del sistema discreto. Esta es la función que se utiliza en este trabajo.

1.7.2. Simulación e implementación

1.7.2.1. Implementación en MATLAB y Simulink

Controlador con 4 estados

Se han diseñado 2 controladores de tipo LQR diferentes.

Primero, se agruparon ambas ruedas como un sistema único, es decir, se asumió que la diferencia de giro de los motores era insignificante haciendo un sistema de 2 grados de libertad. Por lo tanto, ϕ es 0 y la única ecuación de estado que se aplica es la ecuación (23). Como ambas ruedas se consideran un mismo conjunto, solo habría necesidad de una salida v ($v = v_l = v_r$) y la matriz B se simplificaría cogiendo solo una de sus columnas, ya que ambas tienen los mismos valores.

En este caso, la matriz de peso R tendría unas dimensiones de 1x1, es decir,

$$R = 1$$

La matriz Q seria de 4x4, con estos valores iniciales,

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Haciendo múltiples pruebas con distintos calibrados sin llegar a conseguir el equilibrio, se podía observar una tendencia a girar hacia un lado constantemente. Por lo tanto, se comprobó que la suposición de $\phi = 0$ es incorrecta y es necesario considerar cada rueda por separado para conseguir el equilibrio sin que el robot cambie su posición.

Controlador con 6 estados

Considerando los 3 grados de libertad, se utiliza la ecuación (25). Substituyendo los parámetros del robot utilizado en las matrices de estado se consiguen estos resultados:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -807.1 & -458.8 & 458.8 & 0 & 0 \\ 0 & 162 & 62.44 & -62.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -257.8 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 445.9 & 445.9 \\ -60.69 & -60.69 \\ 0 & 0 \\ -98.41 & 98.41 \end{bmatrix}$$

Y estos serían los R y Q iniciales,

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Planta en Simulink

En la figura 21 se muestra la planta diseñada en Simulink.

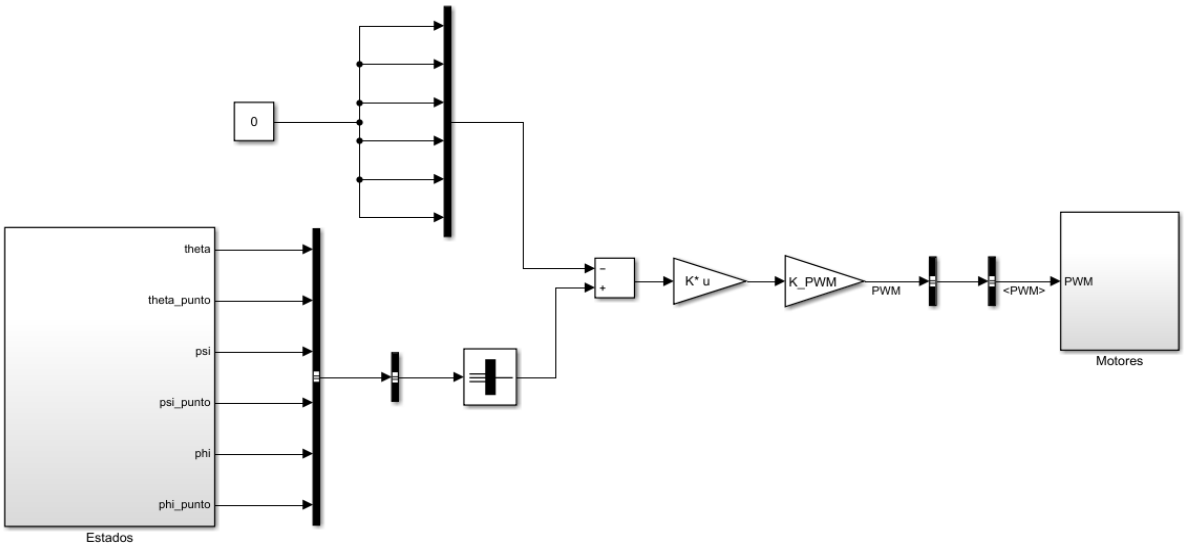


Figura 21. Planta de Simulink del controlador

Como se puede observar, primero, los estados, que han sido obtenidos mediante los sensores, se comparan con la referencia que, como en este caso buscamos que el robot se mantenga en equilibrio sobre un punto, va a ser nula. Luego, se multiplican por la ley de control K. Antes de enviar esa información a los motores se ha de multiplicar los valores obtenidos por una constante K_PWM que simboliza la conversión de volts a PWM (Modulación por ancho de pulsos), ya que los motores de LEGO MINDSTORM funcionan con esta medida.

Offset del giróscopo

Como ya se ha comentado en las practicas con sensores, el sensor giroscópico puede comenzar el programa con un fallo prestablecido, a este fallo se le conoce como *offset*. Para solucionar este problema es necesario calcular el valor inicial del sensor y restarlo posteriormente durante el programa de equilibrio. Para ello se han usado los bloques de Simulink *clock* y *enable*.

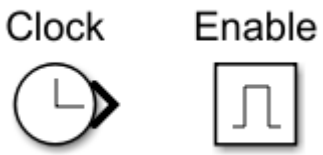


Figura 22. Bloques de Simulink clock y enable

El bloque *clock* (del inglés reloj) genera el tiempo de simulación real y el bloque *enable* (habilitar) hace que un subsistema se active solo cuando le llega una señal positiva y es capaz de mantener los valores obtenidos cuando la señal se anula. Se ha diseñado la planta de manera que en los primeros 2 segundos mida el offset del giróscopo y hasta el segundo 10 no empiece a calcular el ángulo en el que se encuentra el robot. De manera que el usuario ha de dejar el robot en reposo durante los 2 primeros segundos y tiene 8 segundos para ponerlo de pie en, aproximadamente, la posición donde se encuentra el equilibrio. Para que sea más fácil saber cuándo los 2 segundos han finalizado se ha implementado que el ladrillo muestre una luz naranja al principio que se apaga cuando la lectura ha terminado.

Reset de los encoders

Se ha observado, además, que los encoders de los motores no se resetean al inicio de cada programa. Por lo tanto, se ha añadido un reset en el primer segundo del programa. Los bloques del paquete de apoyo de Simulink tienen la opción de implementar un reset con una señal externa. Usando el bloque *clock* de nuevo y un *switch* se activa el reset hasta el primer segundo y se desactiva el resto del tiempo.

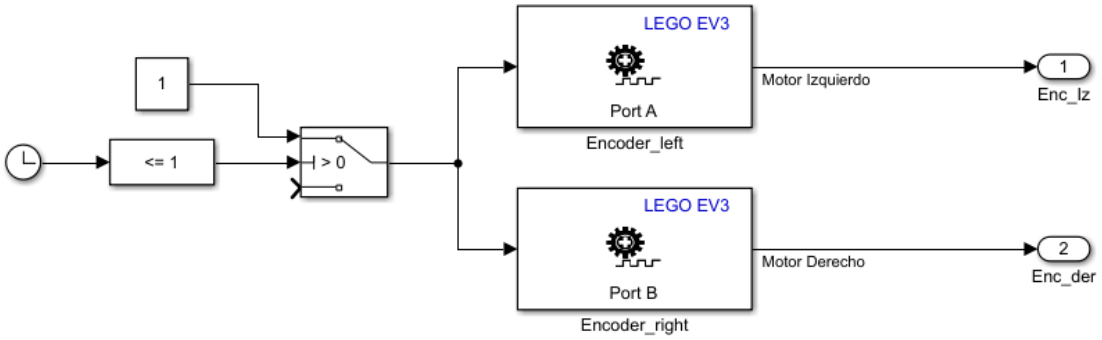


Figura 23. Subsistema de encoders

1.7.2.2. Calibrado del controlador

Una vez calculado y diseñado el programa de equilibrio para el robot LEGO-ANYWAY lo único que falta es calibrar el controlador para encontrar el diseño optimo que mantenga en equilibrio el robot de la forma más estable.

Como ya se ha explicado anteriormente el controlador se calibra cambiando los valores no nulos de las matrices de peso Q y R. Sin embargo, al usar la función `lqrd` en MATLAB el tiempo de muestreo (Ts) también va a influir en el resultado final. La lógica dicta que cuanto más pequeño sea el tiempo de muestreo mejor ya que actualizará más rápido la información recibida y podrá actuar en acorde. Sin embargo, un tiempo de muestreo demasiado bajo puede causar que el robot no tenga tiempo suficiente de hacer los cálculos necesarios y que se produzca un retraso. Después de realizar múltiples pruebas con el robot se ha concluido que el tiempo a utilizar sea $T_s = 0{,}03$.

La matriz R solo influye en las salidas y se puede comprobar que su variación no afecta apenas al resultado final, por lo tanto, se ha decidido mantenerlo en 1. En conclusión, los únicos valores que se cambiaran serán Q_{11}, Q_{22} y Q_{55}.

En primer lugar, se observa cuáles son los valores y resultados obtenidos cuando esos valores son igual a 1. La ley de control K va a tener los siguientes valores:

$$K = \begin{bmatrix} -0.5637 & -63.2355 & -1.0996 & -8.1909 & -0.7014 & -0.0027 \\ -0.5637 & -63.2355 & -1.0996 & -8.1909 & 0.7014 & 0.0027 \end{bmatrix}$$

Antes de probar el controlador en el robot se ha analizado su respuesta teórica mediante gráficos. En el siguiente gráfico se muestra la respuesta que obtendríamos de los tres ángulos en cuando los voltajes de las ruedas tienen ambos una entrada de escalón unitario.

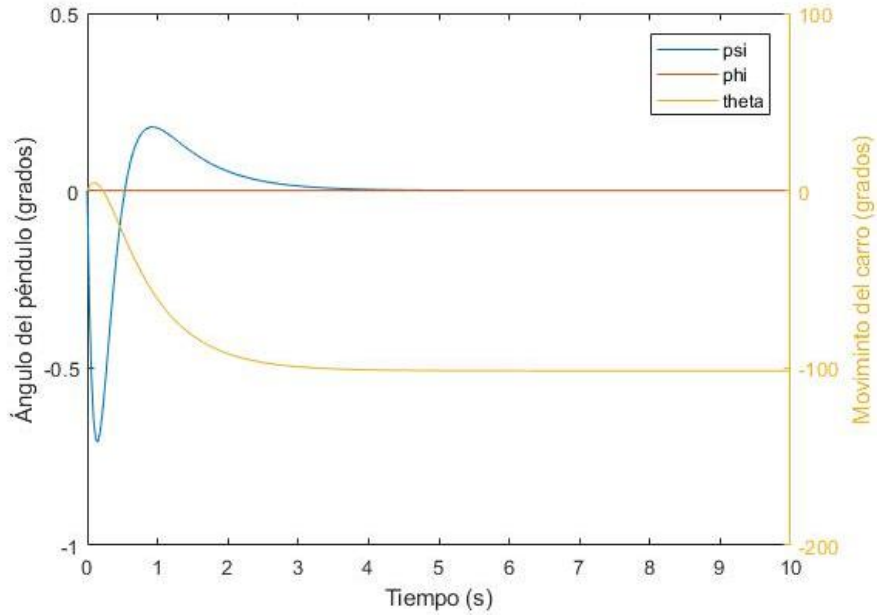


Gráfico 1. Respuesta teórica para $Q_{11}=Q_{22}= Q_{55}=1$

El gráfico muestra los ángulos ψ y ϕ en la escala de la izquierda en grados y el ángulo θ en la escala de la derecha, también en grados, en función del tiempo. Se puede observar en el gráfico que, tras una oscilación hasta aproximadamente -0,75º, el ángulo ψ se estabiliza en unos 3 segundos. Es decir, en tres segundos el péndulo robot debería ser capaz de volver a su estado original. En el caso del ángulo θ, se ve que el carro se mueve unos 100º para lograr estabilidad. Este gráfico no es satisfactorio ya que muestra demasiada oscilación teniendo en cuenta que en la realidad la entrada no va a ser una constante continua, sino que va a cambiar con el tiempo. Además, se ha considerado el voltaje de ambos motores como el mismo haciendo que no haya ninguna perturbación en el ángulo ϕ. Si se cambia un poco la entrada y en vez de un escalón unitario para ambos motores se baja un motor al 0,9 este sería el resultado que se conseguiría (Gráfico 2). Con una pequeña diferencia ya se crea un giro de unos 4º.

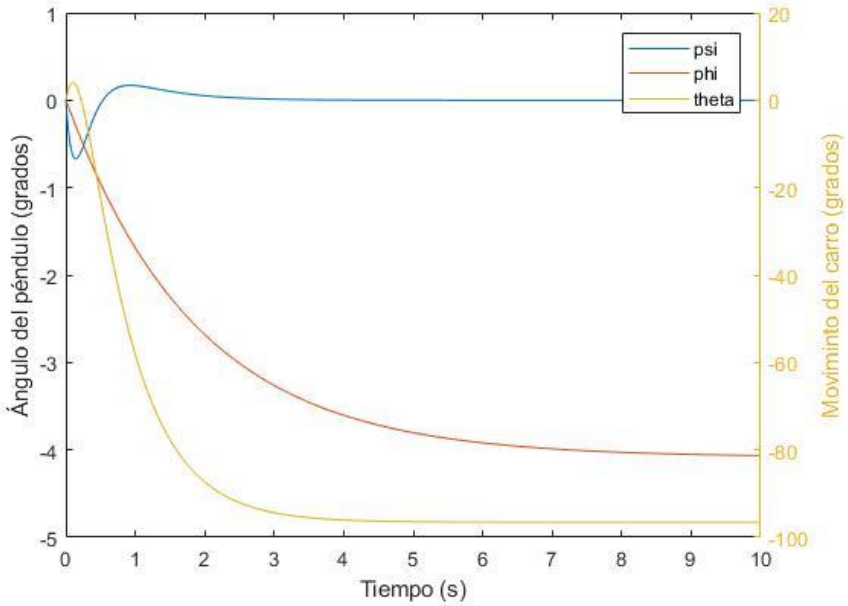


Gráfico 2. Respuesta teórica para Q_{11}=Q_{22}= Q_{55}=1 con diferencia entre motores

A pesar de saber que el resultado no sería favorable se ha simulado este controlador en el robot. El resultado ha sido que el LEGO-ANYWAY respondía correctamente intentando moverse hacia adelante y hacia atrás cuando el ángulo se caía hacia un lado o el otro, pero no tenía suficiente fuerza para mantenerse en pie. Por desgracia no se han podido obtener gráficos en tiempo real del comportamiento del robot ya que para ello es necesario tener una conexión Wi-Fi o de Bluetooth entre el robot y el ordenador. Por ello, la única manera de analizar la respuesta del robot ha sido visualmente.

Cada valor de Q corresponde a una coordenada. El elemento Q_{11} controla el ángulo de las ruedas, el elemento Q_{22} el ángulo del robot y el elemento Q_{55} el giro alrededor de sí mismo.

Cuanto más se aumenten los valores de la matriz Q mejores resultados se obtienen en los gráficos, pero en la realidad llega un punto en el que los esfuerzos del control requeridos son demasiado grandes. Para hacer una prueba se han llevado los valores a números excesivamente altos. Y se puede comprobar cómo aunque teóricamente ψ y ϕ no se mueven más de 0,4º y θ se estabiliza en menos de 4º, en la realidad en cuanto el robot supera unos pocos grados las ruedas responden con una velocidad muy alta llevándolo a la caída.

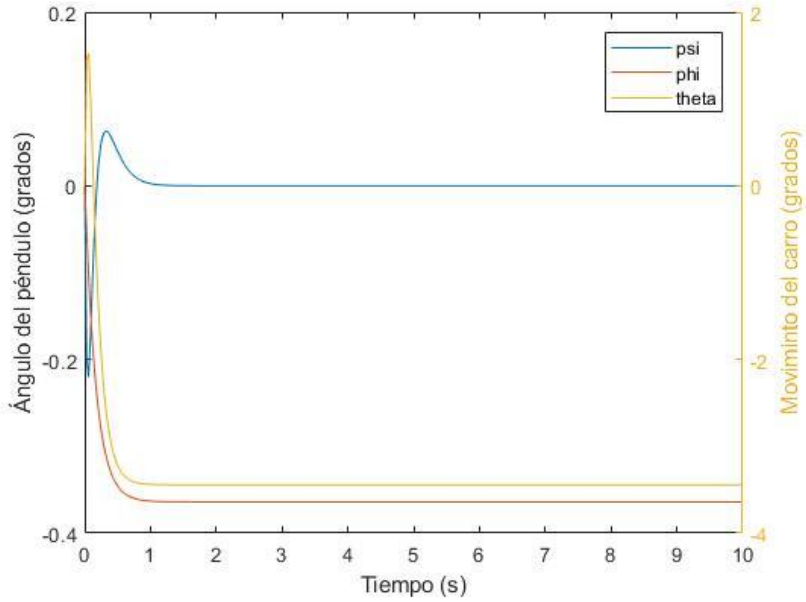


Gráfico 3. Respuesta teórica para Q_{11}=5000, Q_{22}=10000, Q_{55}=300

Después de hacer unas cuantas pruebas, el controlador con los mejores resultados encontrado ha sido el que corresponde a los siguientes valores:

$$Q_{11} = 200, Q_{22} = 2000, Q_{55} = 8$$

$$K = \begin{bmatrix} -6.2917 & -152.7792 & -2.7498 & -20.4050 & -1.9551 & -0.0076 \\ -6.2917 & -152.7792 & -2.7498 & -20.4050 & 1.9551 & 0.0076 \end{bmatrix}$$

Y a continuación está el gráfico obtenidos con ellos.

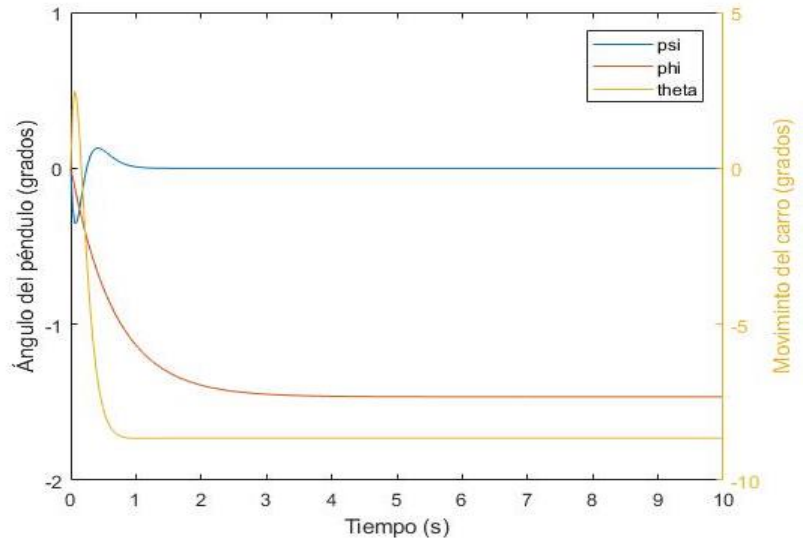


Gráfico 4. Respuesta teórica para Q_{11}=200, Q_{22}=2000, Q_{55}=8

Los resultados conseguidos con este controlador no son del todo satisfactorios. El robot es capaz de mantenerse en equilibrio durante un par de segundos, pero en cuanto el ángulo ψ aumenta ya no es capaz de recuperarse.

1.7.3. Conclusión

En este apartado se ha analizado el modelo matemático del sistema y se ha escogido el controlador LQR como el más adecuado para él. Con todo, no se ha encontrado un controlador que fuera capaz de mantener el robot en equilibrio más de unos pocos segundos.

Esto se puede deber a que no se disponía de una conexión Wi-Fi o Bluetooth para poder visualizar en el ordenador el comportamiento de cada ángulo mediante gráficas. En caso de tener tal conexión se puede hacer una mejor comparativa entre diferentes matrices K y observar en que influye cada cambio en el control.

2. Conclusiones

En este proyecto se ha diseñado un sistema de control de equilibrio para el robot LEGO-ANYWAY. Para ello, se han obtenido las ecuaciones que corresponden con su sistema físico, se ha creado una planta en Simulink que controla el robot y se ha diseñado un controlador de tipo LQR. Sin embargo, no se ha llegado a conseguir el objetivo principal que era que el robot se mantuviera en equilibrio por sí solo durante un tiempo prolongado.

De todas formas, este trabajo puede ser usado como base para encontrar un controlador de mayor precisión y de ahí crear otro tipo de programas.

En este documento también se explican la precisión y el comportamiento de los sensores que pueden ser usados en programas como seguimiento de línea o esquivar obstáculos.

3. Metodología seguida en el desarrollo del trabajo

3.1. Descripción de tareas

A continuación, se muestran las tareas en las que se ha dividido la realización de este trabajo y su duración. Las tareas principales se muestran en un color más oscuro y debajo se encuentran los subapartados en las que consisten.

El trabajo no ha sido continuo ya que no se ha trabajado en el proyecto en épocas de exámenes ni cuando la carga laboral por aparte de otras asignaturas ha sido elevada.

Nombre de la tarea	Fecha de inicio	Fecha final	Duración (días)
Estudio y familiarización de la plataforma LEGO-MINDSTORMS	21/02/2019	17/05/2019	86
Búsqueda de información	21/02/2019	27/03/2019	35
Estudio de los paquetes de soporte de MATLAB y Simulink	28/03/2019	15/04/2019	19
Estudio del comportamiento de los sensores	19/04/2019	17/05/2019	29
Estudio del sistema físico y matemático del robot	29/05/2019	09/06/2019	12
Cálculo de ecuaciones de movimiento	29/05/2019	08/06/2019	11
Obtención de parámetros	09/06/2019	09/06/2019	1
Desarrollo del control de equilibrio	10/06/2019	12/07/2019	33
Búsqueda y estudio de soluciones similares	10/06/2019	15/06/2019	6
Diseño de la planta de Simulink	16/06/2019	22/06/2019	7
Diseño controlador LQR	23/06/2019	30/06/2019	8
Calibrado del controlador	01/07/2019	12/07/2019	12
Redacción de la documentación	13/07/2019	22/07/2019	10
Redacción de la memoria	13/07/2019	22/07/2019	10

Tabla 4. Lista de tareas y su duración

3.2. Diagrama de Gantt

Este diagrama muestra las tareas realizadas en el tiempo. Se han mantenido los colores de la tabla 4 para facilitar su comprensión.

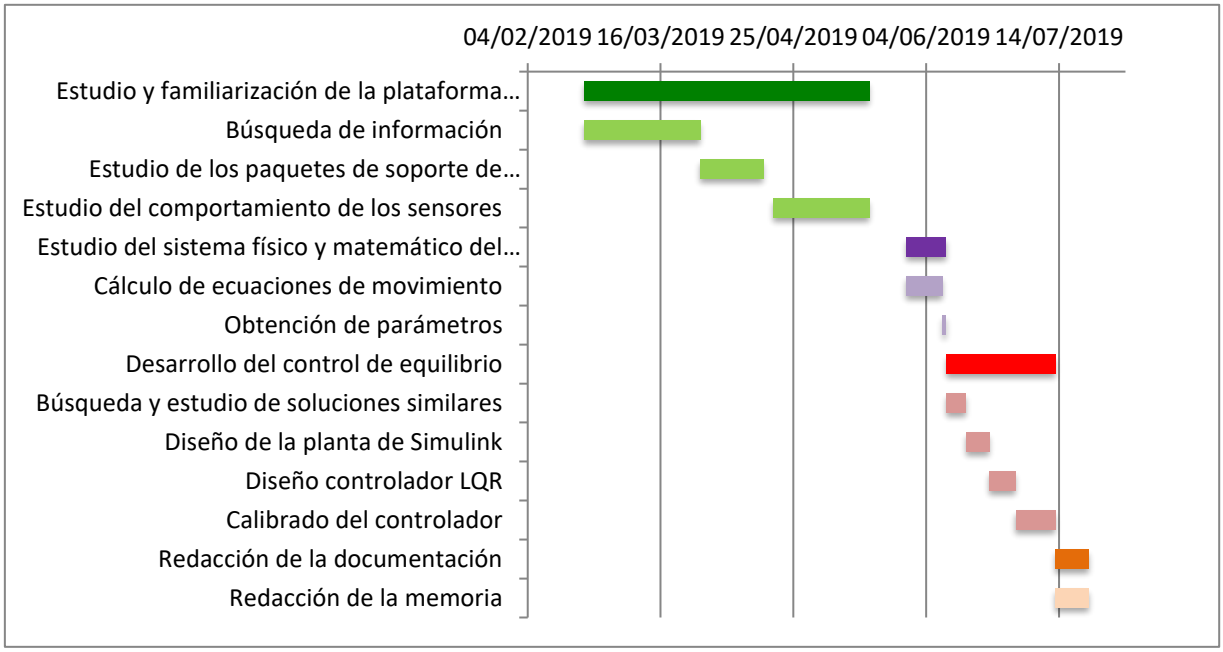


Figura 24. Diagrama de Gantt

4. Aspectos económicos

En este apartado se recogen los diferentes gastos que ha implicado la realización del trabajo y el presupuesto necesario para llevarlo a cabo. Se ha dividido en tres apartados:

- Horas internas:

  Este gasto define el coste debido a las horas dedicadas tanto por el alumno como por el supervisor del proyecto.

- Gastos:

  Aquí se incluye el coste del material utilizado para implementar este proyecto.

- Amortizaciones:

  Las amortizaciones son esos gastos que no se han realizado específicamente para este trabajo. Por ello, solo se considera en el presupuesto el porcentaje del coste total que se ha usado comparado con su vida útil.

En la tabla 5 se muestra el coste incluido en cada apartado y su suma total. Se ha añadido un %5 de los gastos al precio final por imprevistos que puedan ocurrir. Al final, el presupuesto calculado para llevar a cabo este trabajo ha sido de 4.842,39 €.

Horas internas					4.200,00 €
Concepto			Coste por hora (€/h)	Horas (h)	Coste total (€)
Director del TFG			60	20	1.200,00 €
Autor del TFG			20	150	3.000,00 €
Gastos					399,99 €
Concepto					Coste total (€)
Kit LEGO MINDSTOMS EV3					399,99 €
Amortizaciones					11,81 €
Concepto	Coste (€)	Horas útiles (h)	Coste por hora (€/h)	Horas utilizadas (h)	Coste total (€)
Ordenador	850	12000	0,07	150	10,63 €
Licencia MATLAB estudiante	69	8760	0,01	110	0,87 €
Licencia Microsoft office	69	8760	0,01	40	0,32 €
SUBTOTAL (€)					4.611,80 €
Imprevistos (%5)					230,59 €
TOTAL (€)					4.842,39 €

Tabla 5. Presupuesto del trabajo

5. Bibliografía

[1] Anderson, D.P., *nBot Balancing Robot*. Consulta: 15/07/2019. http://www.geology.smu.edu/~dpa-www/robo/nbot/

[2] Watters, A., *LEGO MINDSTORMS: A History of Educational Robots.* Consulta: 16/07/2019. http://hackeducation.com/2015/04/10/mindstorms

[3] *LEGO MINDSTORMS EV3 user-guide*. Disponible: https://education.lego.com/en-us/support/mindstorms-ev3/user-guides

[4] Rollins, M., 2014, *Beginning LEGO MINDSTORMS EV3,* Apress

[5] *LEGO MINDSTORMS EV3 Support from MATLAB*. Consulta: 15/07/2019. https://es.mathworks.com/hardware-support/lego-mindstorms-ev3-matlab.html

[6] *LEGO MINDSTORMS EV3 Support from Simulink*. Consulta: 15/07/2019. https://es.mathworks.com/hardware-support/lego-mindstorms-ev3-simulink.html?s_tid=srchtitle

[7] Segway, *About Us*. Consulta: 16/07/2019. https://store.segway.com/about-us

[8] Yamamoto, Y., *NXTway-GS (Self-Balancing Two-Wheeled Robot) Controller Design*. Consulta: 10/07/2019. Disponible: https://es.mathworks.com/matlabcentral/fileexchange/19147-nxtway-gs-self-balancing-two-wheeled-robot-controller-design

[9] Bilbao, A., Amezua, E., Altuzarra, O., 2008, *Mecánica aplicada: Dinámica,* Editorial Síntesis

[10] Sorazu Irigoyen, R., 2017, *Herramienta docente basada en MATLAB y LEGO para laboratorios de sensorización avanzada*

ANEXO I: Código de MATLAB

```
%% Parámetros físicos
m = 0.016;                          % peso rueda [kg]
R = 0.0216;                         % radio rueda [m]
Jw = m * R^2 / 2;                   % momento de inercia rueda
[kgm^2]
M = 0.617;                            % peso cuerpo [kg]
W = 0.11;                           % anchura cuerpo [m]
D = 0.08;                           % profundidad cuerpo [m]
h = 0.25;                           % altura cuerpo [m]
L = h / 2;                          % distancia G a eje ruedas [m]
Jpsi = M * L^2 / 3;                 % momento de inercia PITCH
[kgm^2]
Jphi = M * (W^2 + D^2) / 12;        % momento de inercia YAW [kgm^2]
fm = 0.0022;                        % coeficiete fricción
cuerpo/motor DC
fw = 0;                             % coeficiente de fricción
rueda/suelo
pi=3.141592654;                     % pi
%% Parámetros motores
Jm = 1e-5;                          % momento de inercia del motor
DC[kgm^2]
Rm = 6.69;                          % resistencia motor DC [Ohm]
Kb = 0.468;                         % constante motor de corriente
continua EMF [V*s/rad]
Kt = 0.317;                         % constante torque motor
DC[Nm/A]
n = 1;                              % relación de transmisión
K_PWM = 8.087;                      % coeficiente volts a PWM [1/V]
Ts = 0.03;                        % Tiempo de muestreo del sistema

%% Variables auxiliares
alpha = n * Kt / Rm;
beta = (n * Kt * Kb / Rm) + fm;
E = [(2*m+M)*R^2 + 2*Jw + 2*n^2*Jm   M*L*R - 2*n^2*Jm;
      M*L*R - 2*n^2*Jm               M*L^2 + Jpsi + 2*n^2*Jm];
F = 2*[beta+fw   -beta;
       -beta     beta];
G = [0   0;
     0  -M*g*L];
H = [alpha   alpha;
    -alpha  -alpha];
I = m * W^2 / 2 + Jphi + (Jw + n^2 * Jm) * W^2 / (2 * R^2);
J = (W^2 / (2 * R^2)) * (beta + fw);
K = alpha * W / (2 * R);
```

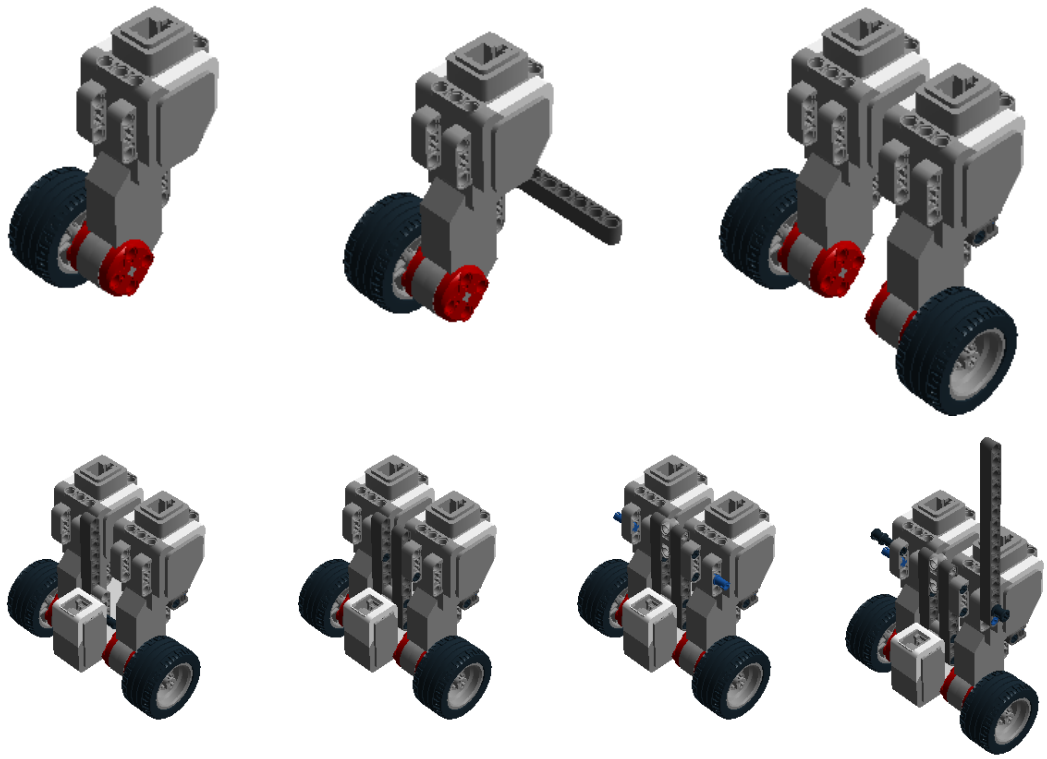
```
%% Matrices de espacio de estado

A1 = [0 0   1 0;
      0 0   0 1;
     -E\G  -E\F];
B1 = [0 0;
      0 0;
      E\H];
C1=[1 0 0 0; 0 1 0 0];
D1=[0 0; 0 0];

A2 = [0   1
      0  -J/I];
B2 = [0    0
     -K/I  K/I];
C2 = [1 0];
D2 = [0 0];
%% Modelos de espacio de estado
s1 = ss(A1, B1, C1,D1);
s1.StateName = {'theta', 'psi', 'theta_dot', 'psi_dot'};
s1.InputName = {'Vl', 'Vr'};
s1.OutputName = {'theta', 'psi'};
s2 = ss(A2,B2,C2,D2);
s2.StateName = {'phi', 'phi_dot'};
s2.InputName = {'Vl', 'Vr'};
s2.OutputName = {'phi'};
%% Añadir los estados phi y phi_punto
s3 = append(s1, s2);
s3.B(end, [1 2]) = s3.B(end, [3 4]);
s3(:, [3 4]) = [];
% Controlador de 4 estados
QQ = s1.C'*s1.C;
RR = 1;
QQ(1,1) = 100;
QQ(2,2) = 1000;

K = lqr(s1.A, s1.B(:,1), QQ, RR)

Ac = (s1.A - s1.B(:,1)* K);
Bc = s1.B(:,1);
Cc = s1.C;
Dc = s1.D(:,1);

states = {'theta', 'psi', 'theta_dot', 'psi_dot'};
inputs = {'v'};
outputs = {'theta'; 'psi'};
```

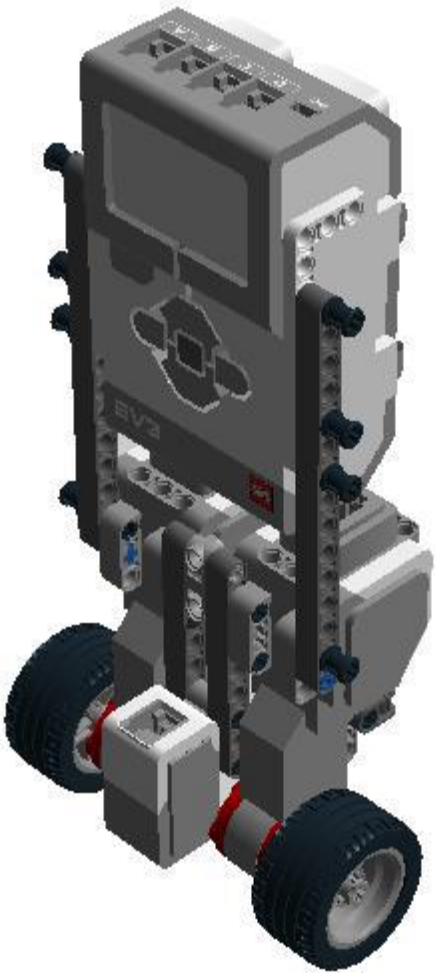
```
sys_cl =
ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname
',outputs);

figure

t = 0:Ts:10;
r =1*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
y=y.*180./pi;
[AX,H1,H2] = plotyy(t,y(:,2),t,y(:,1),'plot');
set(get(AX(1),'Ylabel'),'String','Ángulo del péndulo (grados)')
set(get(AX(2),'Ylabel'),'String','Moviminto del carro (grados)')
legend ('psi', 'theta')
title('Respuesta de escalón con control LQR 4 estados')
% Controlador de 6 estados
QQ = s3.C'*s3.C;
RR = eye(2);
QQ(1,1) = 200;
QQ(2,2) = 2000;
QQ(5,5) = 8;

K = lqrd(s3.A, s3.B, QQ, RR,Ts)

Ac = (s3.A - s3.B*K);
Bc = [s3.B];
Cc = [s3.C];
Dc = [s3.D];

states = {'theta', 'psi', 'theta_dot', 'psi_dot','phi',
'phi_dot'};
inputs = {'vl','vr'};
outputs = {'theta'; 'psi';'phi'};

sys_cl =
ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname
',outputs);

figure(1)
t = 0:Ts:10;
r =[1*ones(1,length(t));1*ones(1,length(t))];  % vl y vr iguales
%r =[1*ones(1,length(t));0.9*ones(1,length(t))];  % vl y vr no
iguales
[y,t,x]=lsim(sys_cl,r,t);
```

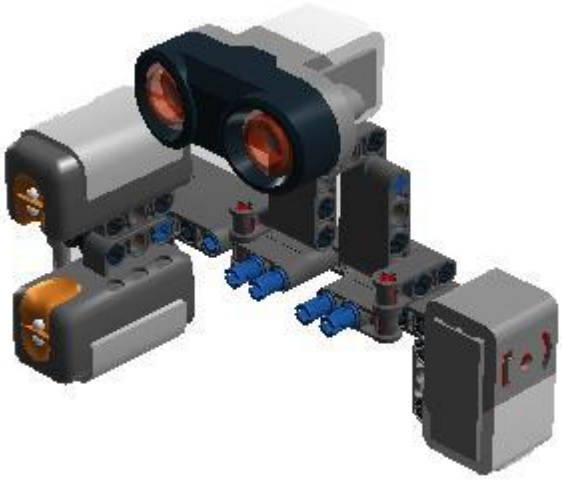
```
y=y.*180./pi;
[AX,H1,H2] = plotyy(t,[y(:,2) y(:,3)],t,y(:,1));
set(get(AX(1),'Ylabel'),'String','Ángulo del péndulo (grados)')
set(get(AX(2),'Ylabel'),'String','Moviminto del carro (grados)')
xlabel('Tiempo (s)')
title('Respuesta de escalón con control LQR 6 estados')
legend ('psi','phi','theta')
```

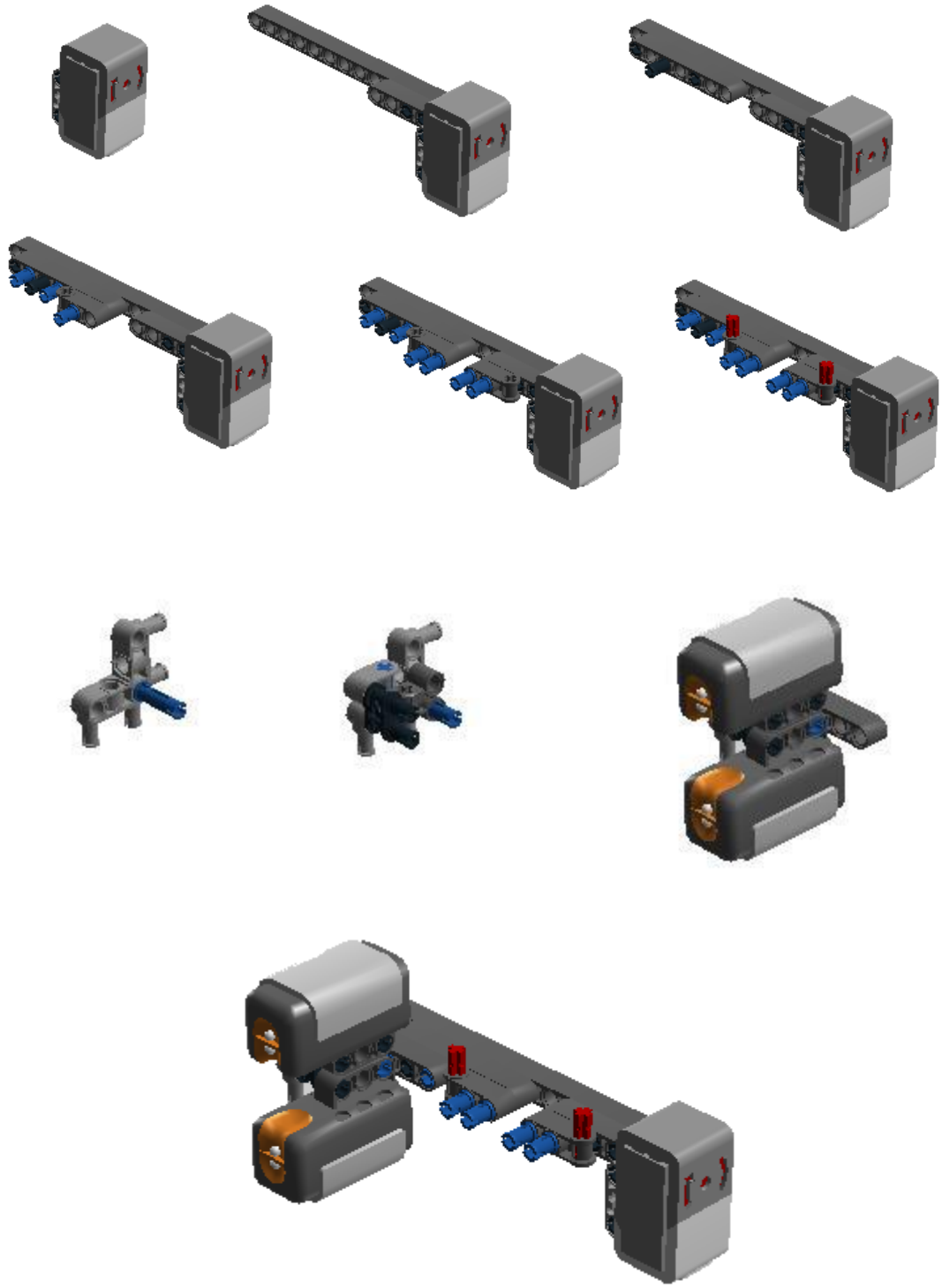
ANEXO II: Montaje del Robot Utilizado

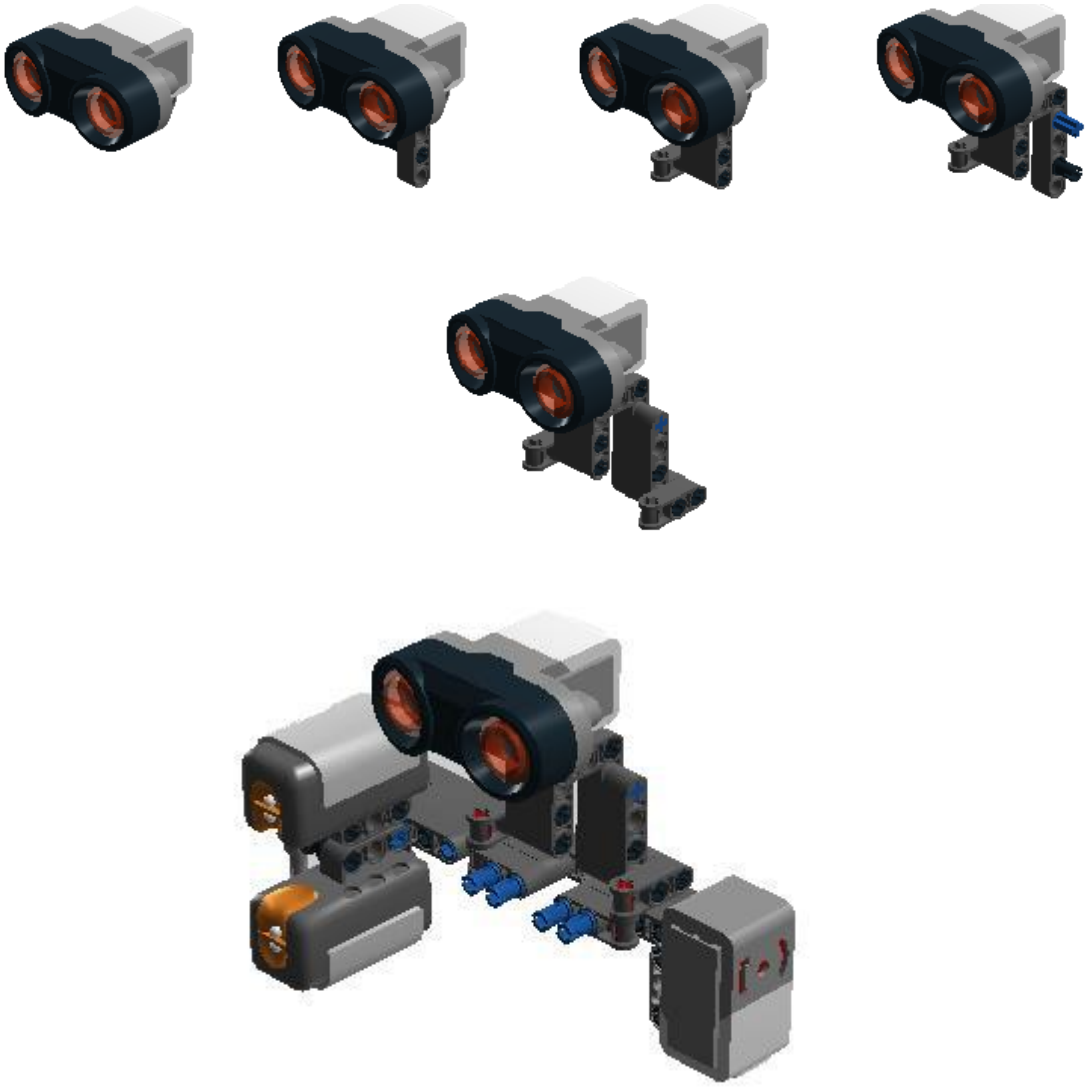
Se ha utilizado el montaje explicado en el Trabajo de Fin de Grado de Rubén Sorazu Irigoyen, *Herramienta docente basada en MATLAB y LEGO para laboratorios de sensorización avanzada* [10], con la sola diferencia de que en el soporte que sujeta el sensor giroscópico se ha sustituido la pieza de la izquierda por otro sensor giroscópico en horizontal. No obstante, ese sensor no se ha usado en el control de equilibrio. A continuación, se muestran imágenes de como montar el robot:




EV3







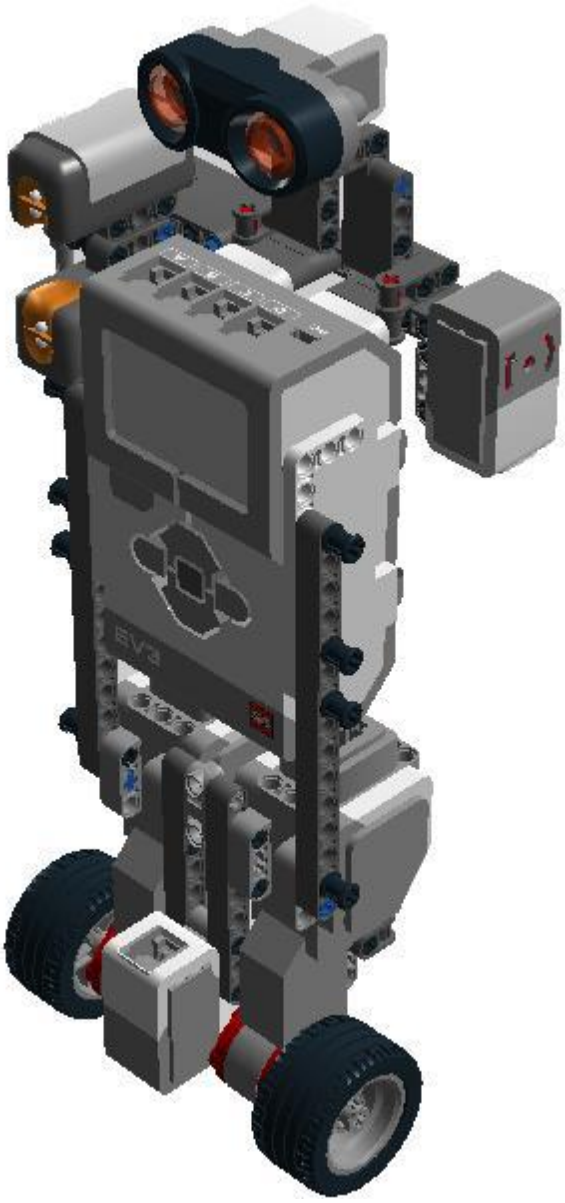


Figura 25. Robot LEGO-ANYWAY utilizado