**Agradecimientos**

Aquí puedes poner lo que estimes oportuno.

**Resumen**

En este TFG se ha implementado una Aplicación Web con la que se podrá visualizar, recoger datos y probar combinaciones sobre un robot móvil con direccionamiento diferencial, cuando trata de seguir un circuito introducido por el usuario.

Para ello, se ha desarrollado la mecánica de este sistema a través de las ecuaciones que relacionan el giro de los motores con la posición y orientación de nuestro vehículo. A continuación, se ha analizado el entorno tecnológico sobre el que se trabajará, decidiendo un entorno Web.

Posteriormente, se han implementado dichas ecuaciones en lenguaje JavaScript y se ha procedido al modelado del objeto y el entorno. Finalmente, se han añadido distintas funcionalidades como la posibilidad de poder guardar los resultados de las pruebas, modificar el vehículo o cargar circuitos de manera dinámica.

# **Introducción**

## **Motivación**

Este trabajo nace motivado por un intento de mejorar una de las herramientas utilizadas en la asignatura “Robótica” de la titulación “Ingeniería Técnica en Informática de Sistemas” impartida en el antiguo Centro de Estudios Superiores Felipe Segundo, que actualmente constituye el Campus de Aranjuez de la Universidad Rey Juan Carlos.

Basado en la siguiente herramienta (MATLAB-Simulink), se pensó en mejorar la herramienta en un entorno Web, que se pudiera utilizar desde cualquier sitio. Además, al realizarlo con un lenguaje de programación tan amplio como JavaScript, en consonancia con WebGL, nos permitía realizar un modelo tridimensional del objeto y que el usuario tuviese diversos métodos para recoger datos.





## **Objetivos**

Vistas las carencias anteriores se podría fijar aquí como subsanarlas (esto fijaría los objetivos y el apartado que tienes actualmente como objetivos te podría valer como texto).

Se pueden poner también aquí los requisitos de la aplicación que se pretende desarrollar. Básicamente sería una lista de los elementos/funcionalidad que debe tener la aplicación. OK, cuando esté terminada la memoria seguro que se pueden agregar más a la lista o retocarla, simplemente es una lista, déjalo para el final.

## **Estado del arte**

Habría que describir aquí las diferentes alternativas existentes para alcanzar dichos objetivos. ¿Hay alguna herramienta para simular este tipo de robots en el mercado? Si las hay, ¿qué ventajas /desventajas ofrecen? Si no las hay, ¿qué alternativas existen para implementar la tuya propia?

## **Estructura de la memoria**

Mostrar la estructura del resto de la memoria mediante un resumen de cada uno de los capítulos de la misma (a razón de un párrafo por capítulo).

# **Robótica móvil**

## **Vehículos con ruedas**

“Los vehículos con ruedas son la solución más simple y eficiente para conseguir la movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas.

Como limitación más significativa cabe mencionar el deslizamiento al impulsar. Es posible que, dependiendo del terreno, aparezcan deslizamientos y vibraciones que desvíen la trayectoria del vehículo. La locomoción mediante ruedas es poco eficiente en terrenos blandos.

Por otra parte, excepto en configuraciones muy especiales, no es posible alterar internamente el margen de estabilidad para adaptarse a la configuración del terreno, lo que limita de forma importante los caminos aceptables del soporte.

Los robots móviles emplean diferentes tipos de locomoción mediante ruedas que les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. La mayor maniobrabilidad se consigue en vehículos omnidireccionales. Un vehículo omnidireccional en el plano es capaz de trasladarse simultánea e independientemente en cada eje del sistema de coordenadas y rotar según el eje perpendicular.”

## **Diferentes configuraciones**

El guiado diferencial es tan solo una entre las varias configuraciones existentes para los vehículos con ruedas. Puede haber diferentes configuraciones de vehículos con ruedas:

* Ackerman: Utilizado en vehículos de cuatro ruedas convencionales. Las dos ruedas delanteras son las que giran, siendo la rueda delantera interior la que gira en un ángulo ligeramente superior para evitar deslizamientos.
* Triciclo: Vehículo de tres ruedas, de las cuales la delantera proporciona la dirección y la tracción. Las dos ruedas traseras solamente proporcionan estabilidad, ya que giran libremente.
* Direccionamiento diferencial: Nuestro vehículo usará esta configuración. El direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales, que también poseen la tracción.
* Skid Steer: Se disponen varias ruedas en cada lado del vehículo que actúan de forma simultánea. El movimiento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha.
* Pistas de deslizamiento: Son vehículos tipo oruga en los que tanto la impulsión como el direccionamiento se consiguen mediante pistas de direccionamiento.
* Síncronas: Consiste en la actuación simultánea de todas las ruedas, que giran de forma síncrona. La transmisión se consigue mediante coronas de engranajes o con correas concéntricas.

## **Direccionamiento diferencial**

**Modelo cinemático**

Tal y como se muestra en la Figura 1, supóngase un sistema de referencia {G} y un sistema {L} con origen en el punto de guiado del vehículo y eje ZL en la dirección del eje longitudinal del vehículo.



**Figura 1**. Modelo cinemático de un robot móvil.

Entonces, si el vehículo tiene una velocidad de desplazamiento y de rotación con respecto a {L}, con respecto a {G} la velocidad es:

Así, si se conoce y , se podría calcular la trayectoria del vehículo (cinemática directa).

A la inversa, si se desea que el vehículo siga una determinada trayectoria se podría calcular cual debe ser la velocidad de las ruedas y el giro (cinemática inversa).

**Modelo discreto**

La derivada de una función puede aproximarse por el cociente incremental:

Esto se conoce como derivada discreta hacia adelante, pero también puede aproximarse con el cociente incremental hacia atrás, la aproximación centrada, u otras aproximaciones más complicadas.

El resultado es que, con una ecuación de este tipo, la nueva coordenada x (en t+1) se puede calcular a partir de la anterior (en t) mediante la ecuación:

Aplicando el mismo resultado al resto de coordenadas del modelo cinemático directo, se obtiene:

Por tanto, si se dispone de las coordenadas en cada instante de un determinado horizonte temporal, se puede calcular la nueva posición y orientación del robot en dicho horizonte.

El direccionamiento diferencial viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue con esas mismas ruedas. Adicionalmente, existen una o más ruedas para el soporte. En la figura 2 se muestra una imagen de dicho esquema. Nótese que para especificar la configuración hay que indicar los valores de las tres variables (x, z, ), siendo las variables de control son las velocidades de las ruedas laterales.



**Figura 2**. Locomoción mediante guiado diferencial.

Sean y las velocidades de giro de las ruedas izquierda y derecha, respectivamente. Si el radio de la rueda es , las velocidades lineales correspondientes son y . En este caso, la velocidad lineal y la velocidad angular correspondientes en el modelo vienen dadas por:

Sustituyendo estas expresiones en las obtenidas a partir de la Figura 1, se obtienen las velocidades de las coordenadas del robot en el sistema {G} a partir de la velocidad de giro de cada rueda:

Finalmente, utilizando el modelo discreto, se obtiene:

## **Navegación autónoma**

Con este vehículo, pretendemos realizar una navegación autónoma mediante el uso de unos sensores infrarrojos CNY70, los cuales detectan la claridad del objeto que hay bajo ellos.

Nuestro circuito será de color negro sobre fondo blanco, por lo que mediante estos sensores podemos detectar en qué momento nuestro vehículo se sitúa encima del circuito y frenar una de sus ruedas en consecuencia.

El vehículo avanzará hasta que uno de los sensores detecte un cambio de claridad. En ese momento, la rueda del mismo lado que el sensor que está detectando el cambio se frenará, proporcionando un giro al vehículo en la dirección que sigue el circuito.

En nuestra aplicación la colisión se calculará en cada momento mediante la diferencia de las posiciones de cada sensor con todos los puntos del circuito. Siendo la posición de un sensor (sx, sz) y el punto del circuito con el que calcular la distancia (px, pz) se calcula con la siguiente ecuación:

Cuando la distancia *d* obtenida sea menor que el radio del sensor, entonces se tomará como que el sensor ha detectado colisión con la línea del circuito, frenando la rueda necesaria.

# **Entorno tecnológico**

## **HTML/CSS/JS**

En este caso, hemos utilizado JavaScript para la parte de la interfaz gráfica porque nos permite incluir librerías que están ya implementadas (Por ejemplo, WebGL), además de ser un lenguaje de programación con una potencia increíble al ser dinámico y orientado a objetos. Nos permite crear objetos y modificarlos sobre la marcha, pudiendo extenderlos de una manera muy sencilla.

Además, JavaScript nos permite utilizar la interfaz gráfica usando la máquina del usuario, por lo que no colapsa un servidor con cálculos tan extensos como los necesarios para todo un sistema gráfico.

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Corresponde con las siglas de *HyperText Markup Language*, es decir, Lenguaje de Marcas de Hipertexto. Estas marcas son mayormente conocidas como etiquetas, las cuales nos permiten ordenar nuestro texto con una estructura básica.

El término CSS corresponde a las siglas en inglés de “Hojas de Estilo en Cascada”. CSS es un lenguaje de diseño gráfico para estructurar y presentar un documento definido por un lenguaje de marcado como puede ser HTML. Utiliza clases que se añaden a las etiquetas HTML, que mediante un conjunto de reglas dan el diseño a la página Web.

JavaScript es un lenguaje de programación interpretado, que se usa principalmente en su forma del lado del cliente (client-side), es decir, que se ejecuta en la máquina del cliente. Actualmente todos los navegadores modernos soportan JavaScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Nos permite ampliar la funcionalidad de las páginas Web con animaciones, cálculos o modificaciones en la estructura.

En nuestra aplicación tendremos botones con diversas funcionalidades como cambiar la cámara, añadir un circuito o guardar los cambios modificados. Éstos funcionan mediante un método que escuchará los eventos de click en ellos, es decir, el método solo se ejecutará en el momento en el que se haga click sobre un botón.

Tendremos varios campos de texto numéricos que nos permitirán introducir los valores que deseemos para modificar la estructura de nuestro vehículo.

También podemos utilizar el teclado para usar la aplicación Web. Podremos acelerar o frenar el vehículo con las teclas *W* o *S*, respectivamente y podremos frenar solamente una de las dos ruedas delanteras con las teclas *A* o *D*.

La información recogida la podremos encontrar en unas etiquetas y una lista que se sitúan en la parte inferior de nuestra página Web.

## **WebGL**

En la Web podemos encontrar varias API’s disponibles para la visualización de gráficos por computador. Un ejemplo son las siguientes API’s:

* Chart.js: Api que permite visualizar conjuntos de datos mediante el uso de gráficos de tablas o mapas de puntos, con animaciones y preparado para utilizarse incluso en móviles.
* ArcGIS: Api diseñada para visualizar mapas en las aplicaciones Web y poder utilizarlos para visualizar datos en ellos.

Para visualizar gráficos renderizados por ordenador tenemos la API WebGL. Es una API basada en OpenGL ES 2.0 que permite llevar a cabo la representación 2D y 3D en un elemento canvas HTML. La representación se realiza mediante código JavaScript donde se describen tanto los objetos que vamos a representar como los cambios físicos que se van a aplicar en ellos (Translación, Rotación y Escalado).

Además del código JavaScript, se utiliza un código en lenguaje GLSL para implementar los shaders, que es un código de sombreado que permite simular la iluminación de focos, luces e iluminación ambiente al impactar sobre distintos objetos. Este código es ejecutado para cada vértice enviado a través de la API y para cada píxel rasterizado en pantalla. La implementación de WebGL compila estas instrucciones shader en código GPU, por lo que aumenta la velocidad de renderizado cuanta más potencia gráfica tengas.

La API de WebGL ha sido implementada por el consorcio de tecnología sin ánimo de lucro Kronos Group, los cuales consolidaron el WebGL Working Group. La versión 1.0 de las especificaciones WebGL fue lanzada en marzo de 2011. En 2013 comenzaron el desarrollo de las especificaciones de WebGL 2.0, las cuales están basadas en OpenGL ES 3.0. Concluyeron en enero de 2017.

Para usar WebGL hay que utilizar un elemento canvas. Este elemento lo inicializaremos desde javascript, utilizando una función específica de WebGL (*canvas.getContext(“webgl”)*). El valor resultante de esta función se lo asignaremos a una variable global (*“gl”* en nuestro caso).

La variable *“gl”* será aquella que utilizaremos para todas las funciones que conlleven un pintado o renderizado de elementos. Con ella podremos determinar los shaders que vamos a utilizar y asignar toda la configuración necesaria. También podremos crear buffers donde guardaremos los objetos completos que usaremos para formar los escenarios, por ejemplo, un cubo, un cilindro o una esfera.

## **GitHub**

En un principio, utilizamos Dropbox para almacenar y tener el código subido a la nube. Esto no nos era suficiente, ya que no podíamos llevar un control exhaustivo del código, ni tener diferentes versiones de la aplicación.

Debido a esto, comenzamos a utilizar Github, que es un sistema de control de versiones que está en la nube (nos permite tener el código en todo momento). Mediante Github, podemos tener el registro de cambios que realizamos en nuestro código.

Además de llevar el registro de cambios, podemos utilizar diferentes ramas para experimentar con GitHub, por ejemplo, para cambiar el sistema de shaders.

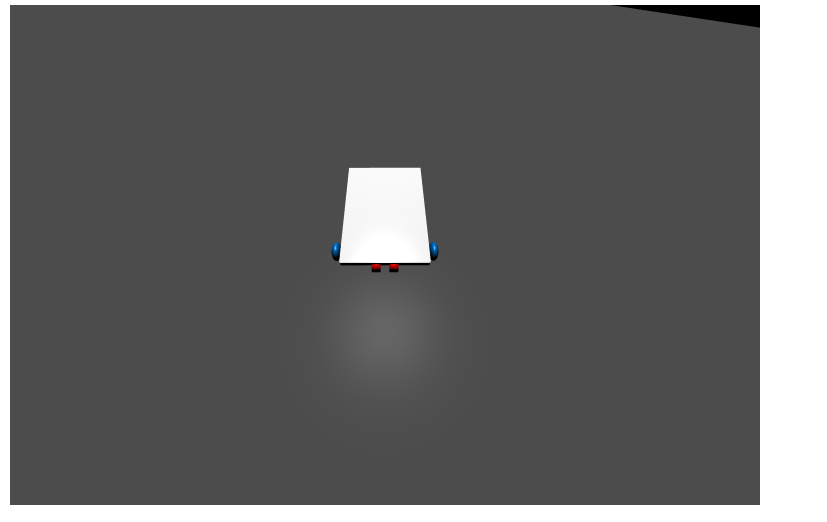
# **Descripción de la aplicación**

En la aplicación podemos observar tres zonas importantes: Viewport, campos de entrada de datos y la zona de información

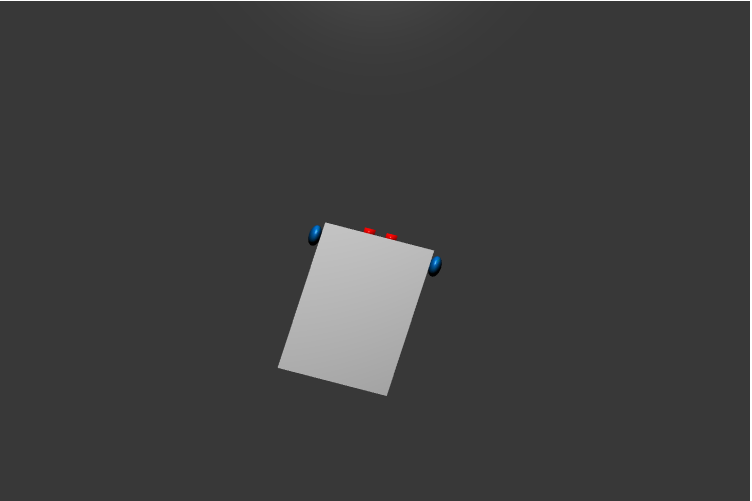
## **Zona izquierda: Viewport**

En la izquierda podemos observar el canvas, donde podemos observar a nuestro robot. En un inicio el robot estará sobre una superficie gris, para poder ver bien el movimiento y las iluminaciones sobre el coche.

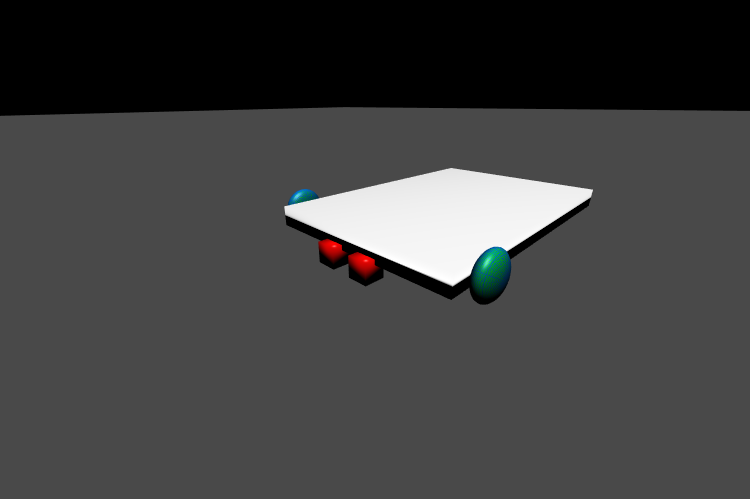
Al comienzo, la cámara estará en centrada en el coche en cualquier modo: “Automática”, “Ortográfica” o “Siguiendo” (Ilustraciones 1, 2 y 3).



1. Cámara Automática

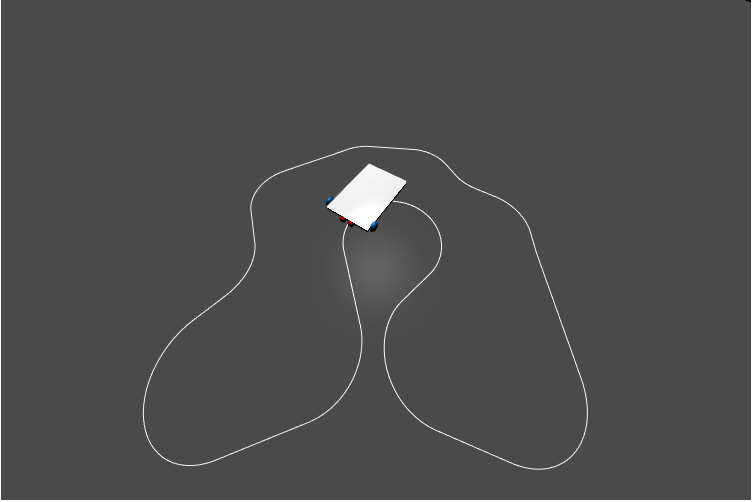


. Cámara Ortográfica

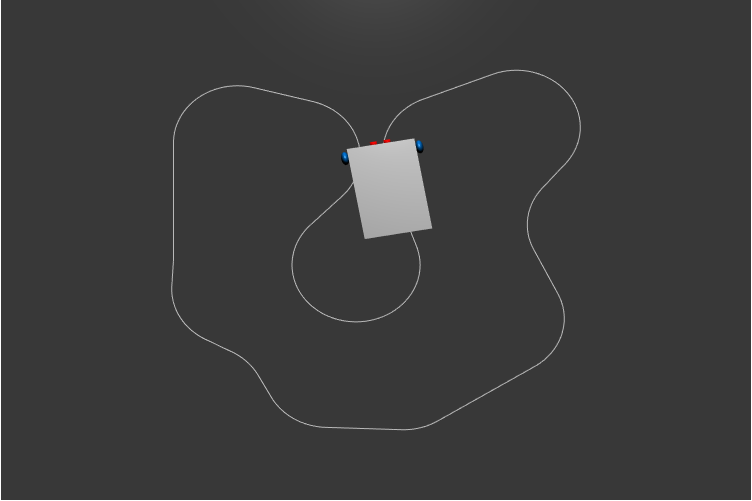


. Cámara Siguiendo

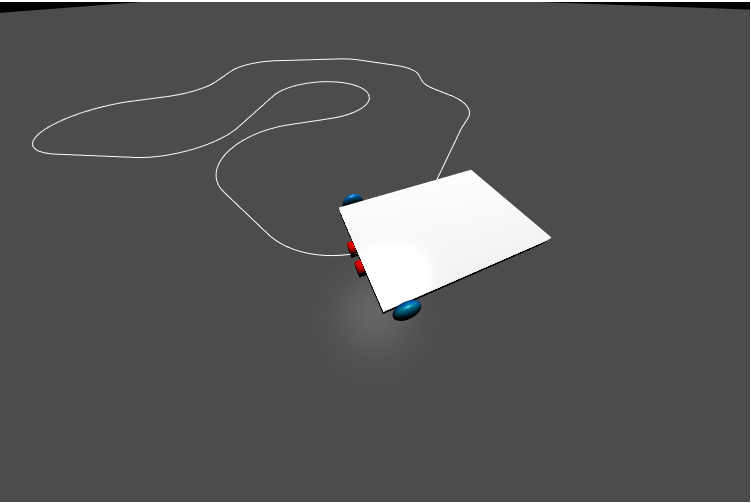
En el momento en el que añadamos un circuito los modos de cámara “Automática” y “Ortográfica” estarán centradas en el punto central del circuito (Ilustraciones 4, 5 y 6).



. Cámara automática centrada



. Cámara Ortográfica Centrada



. Cámara Siguiendo con circuito

Podemos realizar varias acciones con el teclado y ratón sobre el canvas:

* Si hacemos click izquierdo sobre el canvas y arrastramos con el ratón podemos girar la cámara dependiendo del modo que tengamos activado.
* Podemos aumentar y disminuir la velocidad con las teclas “*W*” (Aumentar velocidad) y “*S*” (Disminuir velocidad).
* Podemos girar el robot manualmente con las teclas “*D*” (Derecha) y “*A*” (Izquierda).
* También podemos hacer zoom con la rueda del ratón.

Descripción de la parte izquierda de la aplicación (viewport), explicando cómo se ubican los diferentes elementos en la escena (centrado el circuito en el origen y posición de la cámara en sus diferentes vistas de acuerdo con los dos modelos de proyección).

Si finalmente conseguimos el control de la cámara mediante el ratón (posición y zoom) la descripción de dicho sistema iría aquí.

Finalmente, dicha zona también se aprovecha para mostrar todos los parámetros geométricos del robot sobre una imagen del mismo.

## **Zona derecha: Campos de entrada datos**

En nuestra zona derecha podemos observar distintos campos editables. Éstos campos nos determinarán

Descripción de la parte derecha de la aplicación detallando cada uno de los campos con los que se recogen los datos de entrada, se activan/desactivan funcionalidades, muestran información y se ejecuta la aplicación.

## **Casos de uso**

Se pueden mostrar las diferentes trayectorias de robots diferentes sobre un mismo circuito. En dichas trayectorias se podrían medir longitud, tiempo, etc… dejando abierta la posibilidad de desarrollar otra aplicación que ajuste los parámetros del robot que minimizan alguna de dichas medidas.

# **Conclusiones**

Enumerar las conclusiones del trabajo. Especificar si tras la realización del proyecto habría la posibilidad de llevar a cabo alguna mejora/ampliación como trabajo futuro.

# **Apéndice**

Si se quiere poner algún apéndice se pueden poner aquí diagramas de secuencia, clases, etc… que muestren la implementación de la herramienta desarrollada.

# **Bibliografía**

Los libros ya te los paso yo y lo completas con otras fuentes que hayas consultado tu por tu cuenta.

* Robótica. Manipuladores y robots móviles. *Aníbal Ollero Baturome*
* <https://es.wikipedia.org/wiki/JavaScript>
* <https://es.wikipedia.org/wiki/HTML>
* <https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada>
* <https://www.chartjs.org/>
* <https://developers.arcgis.com/javascript/3/>
* <https://es.wikipedia.org/wiki/WebGL>

**Formato y estilos:**

Encabezado y pie de página.

Títulos de las secciones justificados a la izquierda y sin subrayado.

Intentar hacer figuras propias en la medida de lo posible para que no parezca un corta-pega.

Numeración de las ecuaciones.

Buscar logos del software utilizado.