Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Control autónomo del Renault Twizy y su modelo 3D en Gazebo

Autor: Bryan Kevin Cóndor Romero
Tutor: Carlos Bordons Alba

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería Electróncia, Robótica y Mecatrónica

Control autónomo del Renault Twizy y su modelo 3D en Gazebo

Autor:
Bryan Kevin Condor Romero

Tutor:
Carlos Bordons Alba
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Grado: Control autónomo del Renault Twizy y su modelo 3D en Gazebo

Autor: Bryan Kevin Cóndor Romero

Tutor: Carlos Bordons Alba

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, agradecer a mis familiares, gracias a quienes he podido llegar hasta aquí. Por todo el trabajo y los sacrificios que han realizado para darme una educación y permitirme estudiar en la universidad.

En segundo lugar, a todos mis compañeros y a quienes estuvieron a mi lado en todos estos años de estudio, tanto en los buenos como en los malos momentos.

Y en último lugar, a todos los profesores que me han impartido clase y a mi tutor de TFG Carlos Bordons Alba por asesorarme en este proyecto.

Bryan Kevin Cóndor Romero

Sevilla, 2019

Resumen

En este Trabajo Fin de Grado se realizará un diseño básico preliminar de un sistema de conducción autónoma para el vehiculo eléctrico Renault Twizy.

Se comenzará hablando sobre la conducción autónoma, sus nociones básicas de manera teórica y las distintas ramas de investigación que hay hoy en día. A continuación, se desarrollará una descripción de todos los componentes hardware que serían necesarios para llevar a cabo este trabajo, además de algunos componentes actuadores que se necesita para adaptar el Renault Twizy a la conducción autónoma en futuros proyectos.

También se dará una breve explicación de los diferentes bloques software que forman el sistema de conducción autónoma como el control longitudinal y lateral del vehículo.

Finalmente se realizará la adaptación parcial del modelo 3D del Renault Twizy a Gazebo.

Abstract

In this Final Degree Project, a basic preliminary design of an autonomous driving system for the Renault Twizy will be carried out.

It will start by talking about autonomous driving, its basic notions and the different branches of research that exist nowdays. Then, a description of all the hardware components that will be needed to carry out this Project Will be developed, in addition to some actuator components that are needed to adapt the Renault Twizy to the autonomous driving in future projects.

A brief explanation of the different software blocks that form the autonomous driving systems such as longitudinal and lateral control Will also be given.

Finally the partial adaptation of the Renault Twizy 3D model to Gazebo Will be made.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación y Glosario

API	Interfaz de programación de aplicaciones
BSD	Berkeley Software Distribution
CCD	Dispositivo de carga acoplada
CES	Consumer Electronics Show
DARPA	Defense Advanced Research Projects Agency
DDT	Dynamic Driving Task
ECU	Unidad de control de motor
ESC	Control de estabilidad electrónico
FOV	Campo de visión
GHG	Gases de efecto invernadero
GPS	Sistema de posicionamiento global
GUI	Interfaz gráfica de usuario
HDOP	Dilución de Precisión para el posicionamiento horizontal
NAVLAB	Nombre de la furgoneta autónoma de la universidad de Carnegie Mellon
ODD	Operational Design Domains
OEM	Original Equipment Manufacturer
OVMS	Open Vehicle Monitoring System
PD	Proportional-Derivative control
RFID	Identificación por radiofrecuencia
ROS	Robot Operating System
RTK	Real Time Kinematic
SAE	Sociedad de Ingenieros Automotrices
ToF	Time of flight camera
XMLRPC	Protocola de llamada a procedimiento remoto que usa XML para codificar datos
:	Tal que
<	Menor que
>	Mayor que
\	Backslash
Pipeline	Técnica para implementar el paralelismo dentro de un solo procesador

1 INTRODUCCIÓN

Hoy en día una de las áreas de investigación que está teniendo un gran crecimiento es la navegación y control autónomo aplicado a cualquier tipo de vehículo. Este crecimiento no solo nos está dirigiendo a un futuro en el que cualquier vehículo o robot pueda ser autónomo, sino que también esta permitiendo que los vehículos sean más seguros, eficientes y mas responsables con el medio ambiente.

Aunque hay muchos campos de interés dentro del control y navegación autónoma, en este trabajo se abordará concretamente el control de un coche eléctrico en un entorno simulado, exactamente un Renault Twizy donado por Renault al departamento de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla.

El reto al que hay que enfrentarnos es el de desarrollar un sistema que pueda implantarse en un vehículo para convertirlo en totalmente autónomo. En este camino la finalidad de este trabajo es presentar una propuesta de control para el Renault Twizy que sirva de introducción a un futuro proyecto que dote de autonomía al coche del departamento. Todo esto enfocado a una conducción en entornos urbanos. Para tal fin, los trabajos enfocados a la conducción autónoma necesitán ser realizados en un entorno simulado que permita evaluar los algoritmos de control antes de llevarlos a una prueba de validación en condiciones reales.

1.1. Objetivo

Este trabajo, por tanto, tiene como objetivo el modelado y control del vehíclo. La simulación se llevará acabo en el entorno de simulación de Gazebo y los controles se implementarán en lenguaje Python, todo unido gracias al entorno de ROS (The Robot Operating System).

Para cumplir el objetivo principal, se plantea el siguiente conjunto de objetivos específicos:

- Desarrollar el modelado del coche Renault Twizy que se implementará en Gazebo para posteriormente llevar a cabo las simulaciones.
- Diseñar los sistemas de control que se implementarán en la simulación, que son un control longitudinal y un control lateral.

2 ESTADO DEL ARTE

2.1. Vehículos eléctricos

Desafortunadamente más del 90% de la demanda energética mundial en transporte se cubre con recursos derivados del petroleo hoy en día, sin embargo, los vehículos eléctricos están teniendo una gran aceptación por parte de las personas y es sin duda el futuro de la automoción, para así dejar de lado los combustibles fósiles y sus derivados.

En algunos casos se piensa que el coche eléctrico tiene pocos años de vida, sin embargo, tiene unos 100 años de antigüedad, siendo Robert Anderson quien montó el primer prototipo de vehículo eléctrico entre los años 1832-1839. Pero si se habla de coche como tal nos tenemos que ir al año 1888, año en el que el ingeniero Andreas Flocken construye el primer coche eléctrico con cuatro ruedas. En los próximos años se fue expandiendo el coche eléctrico, así apareció en el año 1897 taxis eléctricos en ciudades de Estados Unidos como Nueva York. En el 1899 se logra una gran hazaña, alcanzar por primera vez la velocidad de 100 km/h con un coche eléctrico. A principios del siglo XX, en el año 1912 la cifra de vehículos eléctricos en EE.UU alcanzaba las 30.000 unidades.

Sin embargo, la aparición del vehículo de combustión interna en la década de 1930 provocó la desaparición total de los vehículos eléctricos y no fue hasta la aparición de las crisis del petróleo y el problema del medio ambiente cuando se retomó la idea de los vehículos eléctricos.

La expansión de este sector aumentó durante todo el siglo XXI con unas cifras de 50.000 vehículos eléctricos para el año 2011, 180 000 en el 2012, 350 000 en el 2013, 710 000 en el 2014 y así hasta alcanzar la cifra de mas de 3 millones de coches en circulación en el año 2017 y mas de 1 millon vendidos durante el mismo año. Datos estimados por la International Energy Agency.

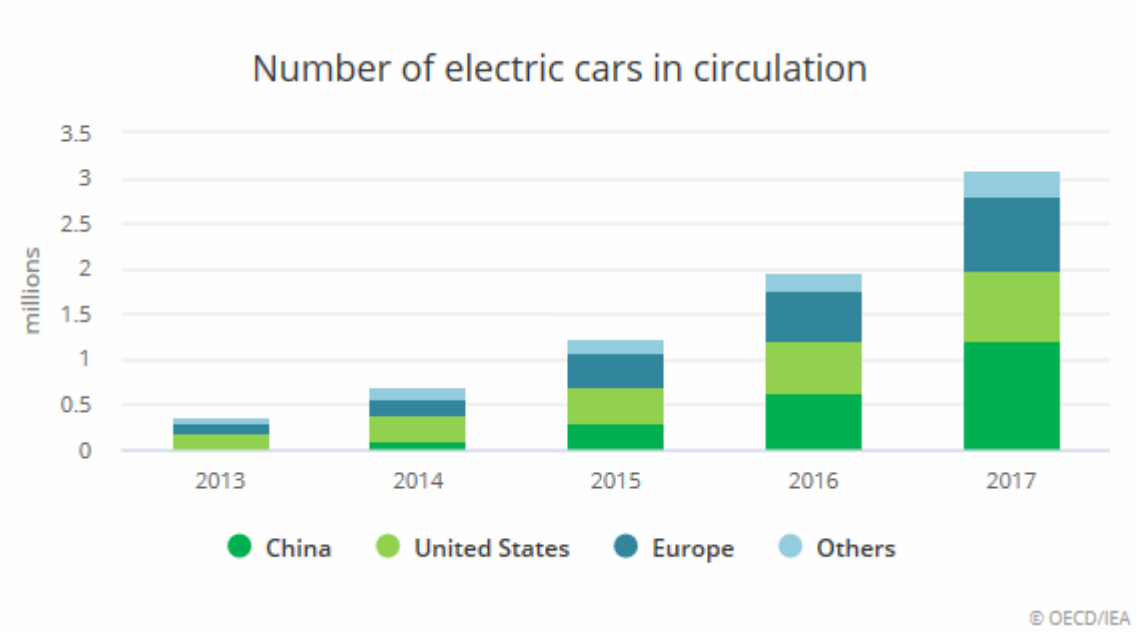

Figura 1 – Numero de coches eléctricos en circulación

Como se puede apreciar en el gráfico de la Figura 1, la tendencia de los coches elécticos tiene un cierto grado exponencial, lo que hace que sea el candidato perfecto para este trabajo.

Otras ventajas que se pueden enumerar de los vehículos eléctricos es que son poco ruidosos, necesitan un menor mantenimiento, no se necesitan filtros contaminantes, el motor eléctrico es igual de potente que un motor de combustión, es controlable (no hay necesidad de tener marchas), es más compacto y tienen mayor eficiencia que los vehículos con motor de combustión. Además, son menos contaminantes como se puede ver en la previsión

de emisión de gases de efecto invernadero para 2030 propuesto por la International Energy Agency. Figura 2.

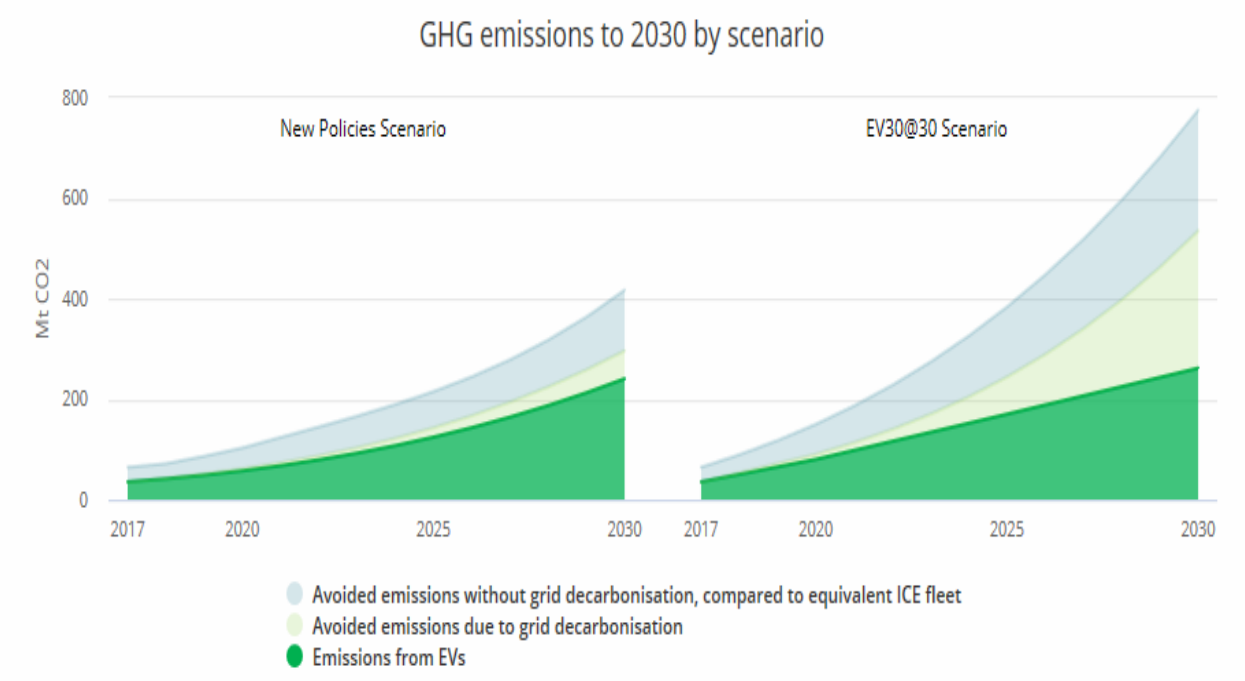

Figura 2 – Emisiones de GHG para 2030

Tambíen hay que enumerar las desventajas de los vehículos eléctricos como es la poca autonomía que se le atribuye, aunque hoy en día ya hay coches que alcanzan los 400 km de autonomía, además de que la red de carga es aun algo reducida, tiempos de cargas algo elevados y sobre todo las baterías contaminantes. Aun con estas desventajas presentes, los coches eléctricos tienen un gran atractivo.

2.1.1 Renault Twizy

El Renault Twizy es el coche escogido para este trabajo debido a que es un gran candidato para poder convertirlo en un coche autónomo debido a las características que nos presenta.

Es un coche pequeño lo que nos facilita el aparcamiento en entornos urbanos donde el espacio es reducido, además su chasis es tubular lo que garantiza una buena protección frente a colisiones. Tambíen cuenta con cuatro frenos de disco que proporciona una seguridad activa total, además de ser muy útil a la hora de diseñar un controlador de estabilidad.

Otro de las características del Renault Twizy es su recarga, que se puede llevar a cabo en cualquier toma eléctrica de 220 V y no requiere de un conector especial, lo que facilita la carga en entornos urbanos. Su tiempo de carga es de unas 3 horas y media y cargándose consume unos 2 000 W, consumo muy similar al de una plancha. También hay que destacar su transmisión reductora que permite mejorar el consumo.

Las características mas técnicas se mostrarán en la Figura 3 y sus dimensiones en la Figura 4.

Homologación	TWIZY 45 Cuadriciclo ligero (L6e)	TWIZY 80 Cuadriciclo pesado (L7e)
TVV		
Nivel de emisión	Cero emisiones: 100% eléctrico	
Número de plazas	2 (1 en Cargo)	
MOTOR		
Tipo motor	3CG - eléctrico asíncrono	
Potencia kW CEE (cv)	4 (5)	13 (17)
Par máx. Nm CEE (m.kg)	33	57
Régimen par máx. (r.p.m.)	de 0 a 2.050 r.p.m.	de 0 a 2.100 r.p.m.
Carburante	Eléctrico	
CAJA DE VELOCIDADES		
Manual - Automática	Automático	
Tipo	Reductor	
Relación de desmultiplicación	1 : 13,4	1 : 9,23
Número de relaciones A.V.	1	
PRESTACIONES		
Velocidad máx. (km / h)	45	80
50 m salida parada (s)	7,5	6,6
0-45 km/h (s)	9,9	6,1
30-60 km/h (s)	5 (hasta 45 km/h)	8,1
CONSUMO CICLO URBANO ECE-15 (en l/100 km y g/km)		
CO_2 (g / km)	0	
Autonomía certificada* ECE-15 (km)	100	90
Autonomía real** (km)	75 a 85	60 a 70
Wh / km	58	63
DIRECCIÓN		
Asistida	cremallera directa	
Ø de giro entre aceras (m)	6,8	
Número de giros del volante	2,8	
TRENES		
Tipo tren delantero	Pseudo-Mc Pherson - Combinado muelle / amortiguador / eje flexible	
Tipo tren trasero	Pseudo-Mc Pherson - Combinado muelle / amortiguador / eje flexible	
Ø barra estabilizadora delantera / trasera (mm)	Delantera y trasera: diámetro 23 mm	
RUEDAS Y NEUMÁTICOS		
Llantas de referencia	33 cm (13")	
Dimensiones neumáticos delanteros	Continental EcoContact 125/80 R13	
Dimensions neumáticos traseros	Continental EcoContact 145/80 R13	
FRENOS		
Tipo de circuito	Circuito simple	
Discos delanteros plenos (Ø en mm)	214	
Discos traseros plenos (Ø en mm)	204	
AERODINÁMICA Y CAPACIDAD		
SCx/Cx	0,64	
Capacidad de energía (kWh)	6,1	
MASAS (kg)		
En vacío en orden de marcha (sin batería)	446 (375)	474 (375)
En vacía en orden de marcha delante	197	206
En vacío en orden de marcha atrás	249	268
Total (MTR)	685	690
Carga útil (CU)	110	115
Masa máx. remolcable frenada	0	
Masa máx. remolcable no frenada	0	

Figura 3 – Datos Técnicos Renault Twizy

VOLUMEN PORTAOBJETOS (dm³)	
Volumen del maletero	31 (180 en Cargo)
Volumen guantera izquierda	3,5
Volumen guantera derecha	5

MEDIDAS (mm)		
C	Voladizo delantero	313
A	Batalla	1686
D	Voladizo trasero	339
B	Longitud total	2338
E	Vía delantera	1094

MEDIDAS (mm)		
F	Vía trasera	1080
G	Anchura delantera Anchura trasera Anchura con retrovisores / con puertas	1237 1232 1381/1396
H	Altura en vacío	1454
H1	Altura en vacío con puertas abiertas mín. / máx.	1818/1980
K	Altura libre en carga	120
L	Reglaje longitudinal asiento delantero	200
P	Altura delantera	908
Q	Altura trasera	843

INFORMACIÓN SOBRE EL RECICLAJE DE LA BATERÍA DE TRACCIÓN DEL VEHÍCULO ELÉCTRICO

- La batería de tu vehículo eléctrico Renault Ión-Litio es reciclable.
- No intentes desmontar la batería, así evitarás cualquier riesgo de shock eléctrico, heridas o quemaduras.
- Sólo un operador habilitado "Z.E." puede realizar esta operación.
- Para cambiar la batería, según los términos del contrato con tu operador de batería, dirígete a tu concesionario Renault habilitado "Z.E.".
- Para cualquier información complementaria, puedes contactar con: RENAULT ESPAÑA COMERCIAL S.A. 902 333 500.

En función del país, este tipo de vehículo denominados "cuadriciclos", están sujetos a una legislación especial. Esta legislación dictamina el tipo de carné necesario para la conducción de Twizy y el tipo de carretera donde se puede circular. En España la versión "Twizy 45", se puede conducir a partir de los 15 años con la licencia de ciclomotor.
Para conducir la versión "Twizy", limitada a 80 km/h, es necesario el carné de conducir tipo B. Para más información, contacta con tu concesionario.

Figura 4 – Dimensiones Renault Twizy

2.2. Navegación Autónoma

Para empezar, se tiene que definir navegación autónoma. Navegación autónoma es la capacidad de ir de un punto a otro evitando obstáculos de forma segura. Este trabajo trata de iniciar la navegación autónoma aplicada a nuestro Renault Twizy, por lo que se explicará de manera breve los antecedentes de la navegación autónoma, aplicaciones y métodos existentes para llevarlo a cabo.

También habría que hablar de los niveles de automatización de la conducción propuesto por el SAE (Society of Automotive Engineers), que ha sido aceptada alrededor del mundo. Estos niveles de automatización van desde el 0 hasta el 5. Figura 5. En los tres primeros niveles que van desde el 0 hasta el 2 se encuentran los coches que se podrían considerar como convencionales. En estos tres niveles lo que tiene que hacer el humano sentado en el asiento de conductor es lo mismo. No importa si estén o no activados las características de ayuda para el conductor, incluso si tus pies están fuera del pedal y no estes dirigiendo el coche, tu tienes que conducir. Además, tienes que supervisar las características de apoyo, tienes que frenar, acelerar o dirigir para manter la seguridad. Algunos ejemplos de las características de apoyo para el conductor para el nivel 1 son los siguientes, freno automático de emergencia, advertencia de punto ciego y de salidad de carril. Para el nivel 2 está el centrado de carril o el control de crucero adaptativo. Para el nivel 3 las dos características del nivel 2 al mismo tiempo.

A partir del nivel 3 hasta el nivel 5 crece gradualmente el grado de autonomía. Los tres niveles comparten lo que el humano tiene que hacer cuando está sentado en el asiento del conductor salvo una excepción en el nivel 3. En todos los niveles, no estás conduciendo cuando las características de conducción automatizadas están activadas incluso si estas sentado en el asiento del conductor. Para el nivel 3 cuando las características de conducción automatizadas lo necesiten tendrás que conducir, y para los niveles 4 y 5 las características de conducción automatizadas no requerirán que conduzcas en ningún momento. Algún ejemplo de estas características para el nivel 3, es el conductor de atascos de tráfico. Para el nivel 4, pedales o dirección de las ruedas pueden o no estar instaladas. Para el nivel 5 lo mismo que para el nivel 4 pero las características pueden conducir donde sea y en cualquier condición.

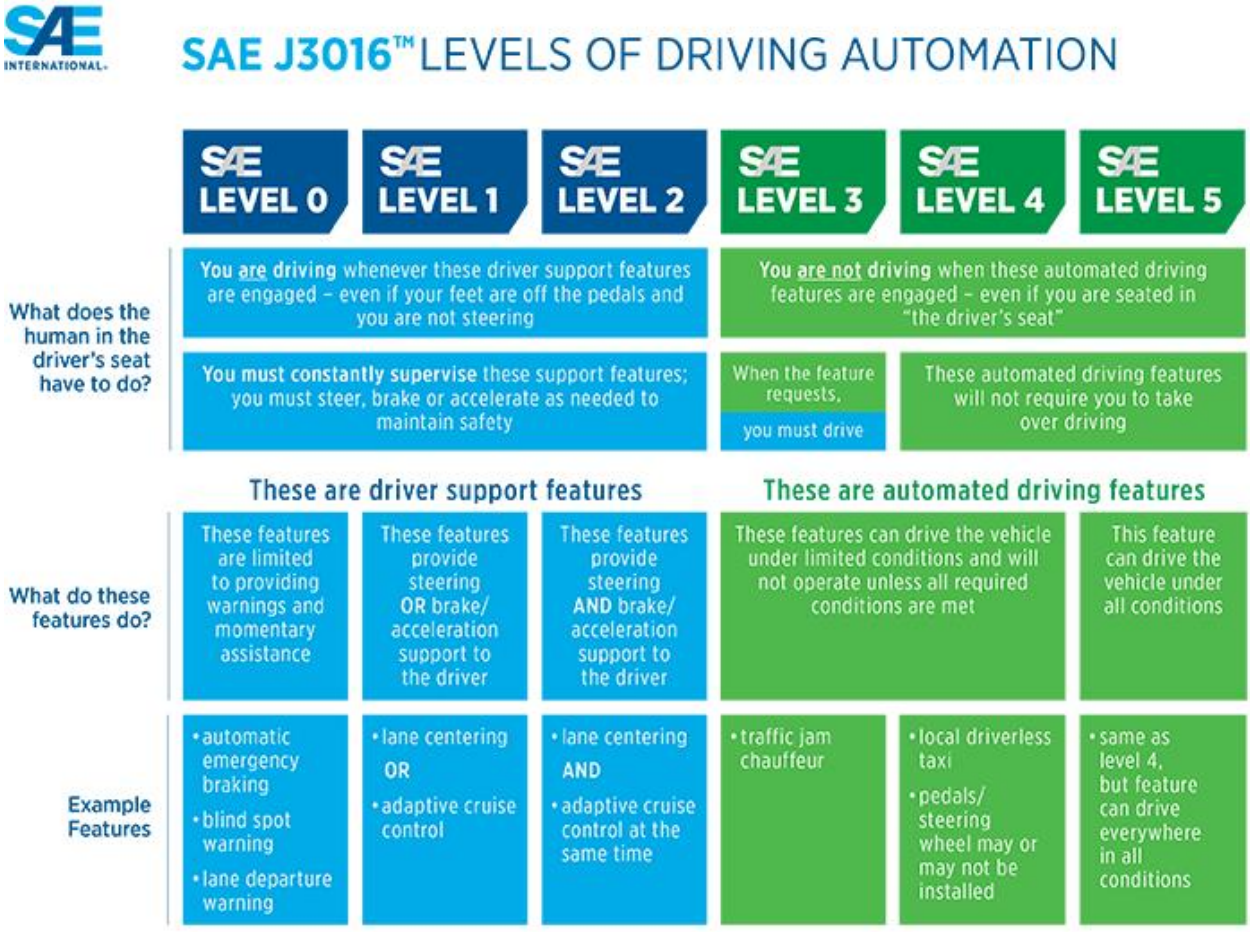

Figura 5 – Niveles de automatización de la conducción

La navegación autónoma es altamente atractiva debido al potencial que tiene para el aumento de la seguridad, productividad y eficiencia del combustible. La seguridad aumentaría debido a la reducción de accidentes provocados por humanos. La productividad se incrementaría debido a que las personas podrían trabajar mientras viajan. La eficience del combustible se beneficiaría debido a la planificación de rutas óptimas, además que los coches totalmente autónomos podrían circular con poco margen de distancia y así también aumentar la eficiencia de las carreteras.

La navegación o conducción autónoma ha sido visionada desde hace mucho tiempo en la ciencia ficción como en la película Minority Report y en el campo de la robótica. Si bien el proyecto PATH (Horowitz y Varaiya 2000), así como Navlab y "Manos libres en América" (Jochem y Pomerleau 1995) fueron los primeros proyectos en el campo de la conducción totalmente autónoma, este tema lleva en nuestras vidas desde hace algo más de tiempo. El primer vehículo con algo de autonomía que se conoce fue presentado en la feria Futurama de 1939, que se trataba de un coche semi eléctrico, pero no encaja dentro de la definición de coche autónomo que se tiene hoy en día. El siguiente vehículo fue presentado por Mercedes-Benz en 1980 y diseñado por Ernst Dickmanns.

Entre los proyectos americanos se puede destacar los primeros vehículos NAVLAB de la universidad de Carnegie Mellon, que sirvieron como plataforma para desarrollar, integrar y testear tecnologías de guiado avanzadas para vehículos autónomos. Estos vehículos podían ser conducidos en redes de carreteras y gracias a ellos se introdujo el concepto de occupancy grids y algoritmos de planificación entre otros. (Elfes 1989; Stentz 1994). Otro programa fue desarrollado dentro del mismo periodo de tiempo como parte de las aplicaciones de defensa, como por ejemplo el programa de vehículos no tripulados experimentales (XUV) de Demo III. Estos vehículos se movían a una velocidad de 32 Km/h en el día y a 16 Km/h durante la noche para 2001 (Schoemaker and Borsntein 1998). Este programa mostró las dificultades encontradas en el uso de camaras de video para detectar el alcance, principalmente usando visión estereo, además de las dificultades para medir la profundidad y las ventajas de usar sensores activos como láseres de escaneo.

En Europa se destaca el proyecto Prometheus (Programa para un tráfico europeo de mayor eficiencia y seguridad sin precedentes 1987-1995). Los primeros coches considerados autónomos fueron el VITA_2 y el VaMP. Ya a principios de 2000, el interés de los coches autónomos iba aumentando en varios países. Un gran ejemplo es el programa alemán PRIMUS que abordaba maniobras no tripuladas en entornos no conocidos. La guía de este vehículo se baso en emplear un Ladar a 3-D para detectar obstáculos y una cámara a color para detectar el camino. Ya durante los siguientes años se trataron dos problemas principales, la robustez, los problemas de eficiencia para la percepción del vehículo (Coue et al. 2006), y el manejo de entornos dinámicos e inciertos (Laugier et al. 2007).

Ya más recientemente uno de los mayores proyectos fue tomado por DARPA (Defense Research Advance Projects Agency) que patrocinaron 3 grandes desafíos. El primero tuvo lugar el 13 de marzo de 2004, 107 equipos fueron registrados y solo 15 de ellos formaron parte de la carrera final. La carrera consistía en recorrer 142 millas fuera de carreteras en menos de 10 horas. Nadie pudo superar siquiera el 5% de la prueba. El segundo desafio tuvo lugar el 8 de octubre de 2005, el desafio era similar y esta vez se registraron 195 equipos, pero solo llegarona la final 23 equipos. De estos 23 equipos solo 5 terminron la carrera con el coche "Stanley" de Stanford a la cabeza con un tiempo por debajo de 7 horas.

El tercer gran desafio, noviembre de 2007, tuvo lugar en la ciudad de california, donde los competidores tenían que seguir las reglas de tráfico de California, además de estar en un entorno urbano donde podrían interactuar con otros agentes, como otros coches autónomos, coches aparcados y coches conducidos por humanos. De los 89 equipos registrados solo 11 llegaron a la final y solo 6 acabaron la carrera, con el coche "Boss" de la universidad de Carnegie Mellon en el primer lugar.

Ya en 2012 Stanford's Dynamic Design Lab en colaboración con Volkswagen Electronics Research Lab crearon Shelley, que usa un GPS diferencial para saber donde está exactamente en lugar del GPS convencional. En 2013 Daimler R&D con el instituto de tecnología de Karlsruhe hicieron un coche capaz de ir desde Mannheim hasta Phorzheim conduciendo de manera autónoma unos 100 km.

Para 2014 Induct Technology lanza Navia, el primer coche autónomo disponible para venta comercial. Estaba limitado a unos 20 Km/h y destinado a transportar personas por centros urbanos peatonales, aeropuertos, parques temáticos, campus universitarios, entre otros. A finales de este mismo año Tesla Motors anuncia su primera versión de su coche autopilotado, que contaba con control de dirección, frenado y velocidad basados en señales de reconocimiento de imagen. Su sistema también contaba con aparcamiento automático y poder ser actualizado.

Para 2015 varios estados de EE. UU, como Nevada, Florida, California, Virginia, Michigan y Whashington, permitieron el testeo de coches autónomos en sus calles. Para abril de este mismo año un coche diseñado por Delphi Automotive se convirtio en el primer vehículo capaz de recorrer de costa a costa Norte América, con un 99% de la distancia computada en su ordenador.

Para 2017 nadie había sido capaz de llegar al nivel 3 de conducción autónoma, y fue en este año cuando Audi lanzó su A8 con una velocidad máxima autónoma de 60 Km/h. Además, Audi fue el primero en usar escáneres láser además de cámaras y sensores ultrasónicos para su sistema.

Para noviembre de 2017, Waymo anunció que había estado haciendo pruebas con su coche no tripulado, sin la presencia de un conductor de seguridad, pero con un supervisor a bordo. En octubre de este mismo año Waymo había estado viajando de manera autónoma por mas de 16 000 000 Km convirtiéndose así en uno de los lideres en el sector. De esta manera, para noviembre de 2018, Waymo fue el primero en comercializar un servicio de taxi totalmente autónomo en los Estados Unidos.

Y así llegamos a la actualidad, en la que hay una gran diversidad de compañías involucradas en el sector de los coches autónomos comerciales. Como, por ejemplo:

- Aurora es una compañía que colabora con Hyundai, Byton y Volkswagen. Su coche autónomo, VW e-Golfs, están en las carreteras de Palo Alto, San Francisco y Pittsburgh.
- Drive.ai esta ofreciendo un servicio de conducción autónoma en Texas, exactamente en Arlington.
- Ford está trabajando con Argo AI para sus propios coches autónomos, compañía que posee algunos coches autónomos en las calles de Detroit, Michigan, Miami, California y Pittsburgh para el testeo de su tecnología.
- Aptiv empezó con el servicio de sus coches autónomos durante el CES (Consumer Electronics Show) de 2018. Y a finales de mayo de 2019 anunció que había completado 50 000 viajes en Sin City, Las Vegas, además la puntuación media de los consumidores es de 4.97 sobre 5.
- Por su parte General Motors ha adquirido a la compañía de coches autónomos Cruise desde 2016, y su tercera generación de coches están en las calles de San Francisco, Scottsdale, Arizona y Orion, para ser testeados.
- Tesla por su parte cuenta con un autopiloto que según la descripción que nos proporcionan ellos mismos, sería capaz de que el coche gire, acelere y frene automáticamente, además de autoaparcamiento tanto en paralelo como en perpendicular y que el coche se dirija hacia el propietario dentro de la zona de aparcamiento.
- Yandex una empresa rusa que lanzó su proyecto de coche autónomo en 2017, ofreciendo un servicio de transporte autónomo desde la segunda mitad de 2018 en las ciudades de Innopolos y Skolkovo. Además, hoy en día esta testeando sus coches autónomos en las ciudades de Moscú y Tel Aviv, demostrando al mundo lo que su coche puede hacer en este último CES de 2019.
- Volvo también esta dentro del sector autónomo junto con la compañia Veonner. Recientemente lograron obtener permiso para que sus coches vuelvan a poder ser testeados en las calles de Suecia. También hay que resaltar que Volvo también esta desarrollando su propio camión autónomo, enfocados tanto para transporte como industriales, como el Volvo FMX capaz de trabajar bajo tierra.
- Waymo por su parte, es considerada una de las más avanzadas compañías en el tema de coches autónomos, de hecho, tiene sus propios coches autónomos disponibles comercialmente.

Como se puede observar los coches autónomos están llegando a nuestras vidas, y a pesar de sus inconvenientes y accidentes como el de Tesla Autopilot en China, los de Waymo en los años 2015 y 2016, el de Navya con su bus autónomo con pasajeros dentro y el peor de todos, el segundo accidente de Uber que desafortunadamente produjo la muerte de un peatón. Todos estos casos plantearon muchas cuestiones legales y se creó mucha controversia alrededor de los coches autónomos, que aún hoy en día generan debates.

2.2.1 Arquitectura del Software

Para entender mejor el proceso de automatización del vehículo, primero se tiene que introducir algunos términos muy importantes definidos en SAE J3016:

- Dynamic Driving Task (DDT) – funciones operativas y tácticas en tiempo real requeridas para operar un vehículo, excluyendo funciones estratégicas como la programación de viajes o la planificación de rutas.
- Driving automation system – sistemas de hardware y software que sean capaces colectivamente de realizar parte o la totalidad del DDT de forma sostenida. Driving automation systems suelen estar compuestos por funciones espedíficas de diseño denominadas características (features).
- Operational Design Domains (ODD) – Las condiciones específicas bajo las cuales un determinado sistema de automatización de conducción o característica está diseñado para funcionar.
- DDT feature – una funcionalidad específica de diseño en un nivel específico de automatización de conducción con un ODD particular.

Además de las limitaciones hardware, un coche totalmente autónomo implica la automatización del DDT en todos los ODDs, desarrollando un Sistema de automatización de conducción. Recursivamente este Sistema de automatización de conducción está compuesto por características específicas de diseño. Luego un coche totalmente autónomo se considera como el desarrollo, implementación y la organización de suficientes características de DDT para satifacer todas las condiciones (ODD) en las que el ser humano pueda operar un vehículo.

Para la SAE un coche totalmente autónomo debe tener lo siguietne:

- Control del movimiento lateral del vehículo mediante steering.
- Control longitudinal del vehículo mediante aceleración y deceleración.
- Monitorización del entorno de conducción mediante detección de objetos y eventos, reconocimiento, clasificación y preparación de respuesta.
- Ejecución de la respuesta de objetos y eventos.
- Planificación de maniobras.
- Mayor visibilidad a través de la iluminación, señalización y gesticulación, etc.

Además, un coche totalmente autónomo tiene que ser capaz de planificar la ruta óptima entre dos puntos proporcionados por el usuario. También hay que decir que una arquitectura para un coche totalmente autónomo tiene que ser reactivo y deliberativo, es decir, tiene que ser capaz de planificar para el futuro y reaccionar en el menor tiempo posible a eventos inesperados. En este sentido vamos a ver unos componentes funcionales para lograr esta arquitectura.

- Abstracciones de sensors: Proporcionan interfaces de software para los sensors de hardware, adaptadores y la funcionalidad de conversión necesaria para interpretar los datos. Distinguimos dos clases de sensors y de abstracciones. Sensores que supervisan el estado interno del vehículo (mediciones inerciales, velocidad, etc.). Sensores que supervisan el entorno externo según los requisitos del SAE. El monitoreo Ambiental puede estar basado en tecnologías RADAR, LIDAR y de cámaras. En caso de una conducción cooperativa, la comunicación con los otros participantes del tráfico se realiza mediante vehicle-to-everything (V2X). El posicionamiento global del vehículo se lleva a cabo con el GPS, además el GPS es necesario para trazar rutas globales. Todas las abstracciones relativas al estado interno del vehículo se agrupan en un componente funcional, porque la elección es específica para cada OEM.
- Fusión de sensores: Varios sensores del entorno generan una gran cantidad de datos con diferentes resoluciones que algunas veces se corrompen debido a la variedad de condiciones de ruido y desorden que cambian continuamente debido a los cambios temporales en el medio ambiente. La fusión de sensores combina datos de diferentes fuentes para incrementar la precisión de las medidas. Los componentes funcionales se eligen con respecto a los requisitos del SAE para la detección,

reconocimiento y clasificación de objetos y eventos. Se distinguen entre objetos dinámicos y estáticos (p.ej. una barrera y peatones) y objetos de carretera (p.ej. semáforos). Mediante esta fusión de sensores, se puede usar un procesamiento pipeline reuniendo información de varios sensores para clasificar un objeto, determinar su velocidad, y añadir otras propiedades a sus descripciones.

- Modelo mundial: El modelo mundial representa la imagen completa del entorno exterior percibido por el vehículo, junto con su estado interno. Se usa conjuntamente los datos que vienen de la fusión de sensores y datos almacenados (p.ej. mapas) para crear una completa representación del mundo. El modelo mundial actúa como un buffer entre el procesamiento de sensores y generación de comportamiento. EL modelado mundial almacena y utiliza información histórica (de bucles de procesamiento pasados) y proporciona interfaces para consultar y filtrar su contenido para otros componentes. Estas interfaces, denominadas data sinks, filtran el contenido o agrupan datos para diferentes consumidores con el fin de revelar diferentes perspectivas.
- Generación de comportamiento: La generación de comportamiento es la clase cognitiva más alta de las funciones en la arquitectura. Aquí el sistema genera predicciones sobre el entorno y el comportamiento del vehículo. Según el objetivo del vehículo, se desarrolla varias opciones de comportamiento (generación de comportamiento) y se selecciona el mejor de todos (selección de comportamiento). Un objetivo de un vehículo es el de alcanzar un destino sin ningún accidente. Cuando el destino cambia (a través de una interfaz maquina-humano (HMI)), el componente de enrutamiento global cambiará el objetivo y desencadenará una nueva generación de comportamiento. Estos componentes corresponden a las funciones estratégicas de SAE.

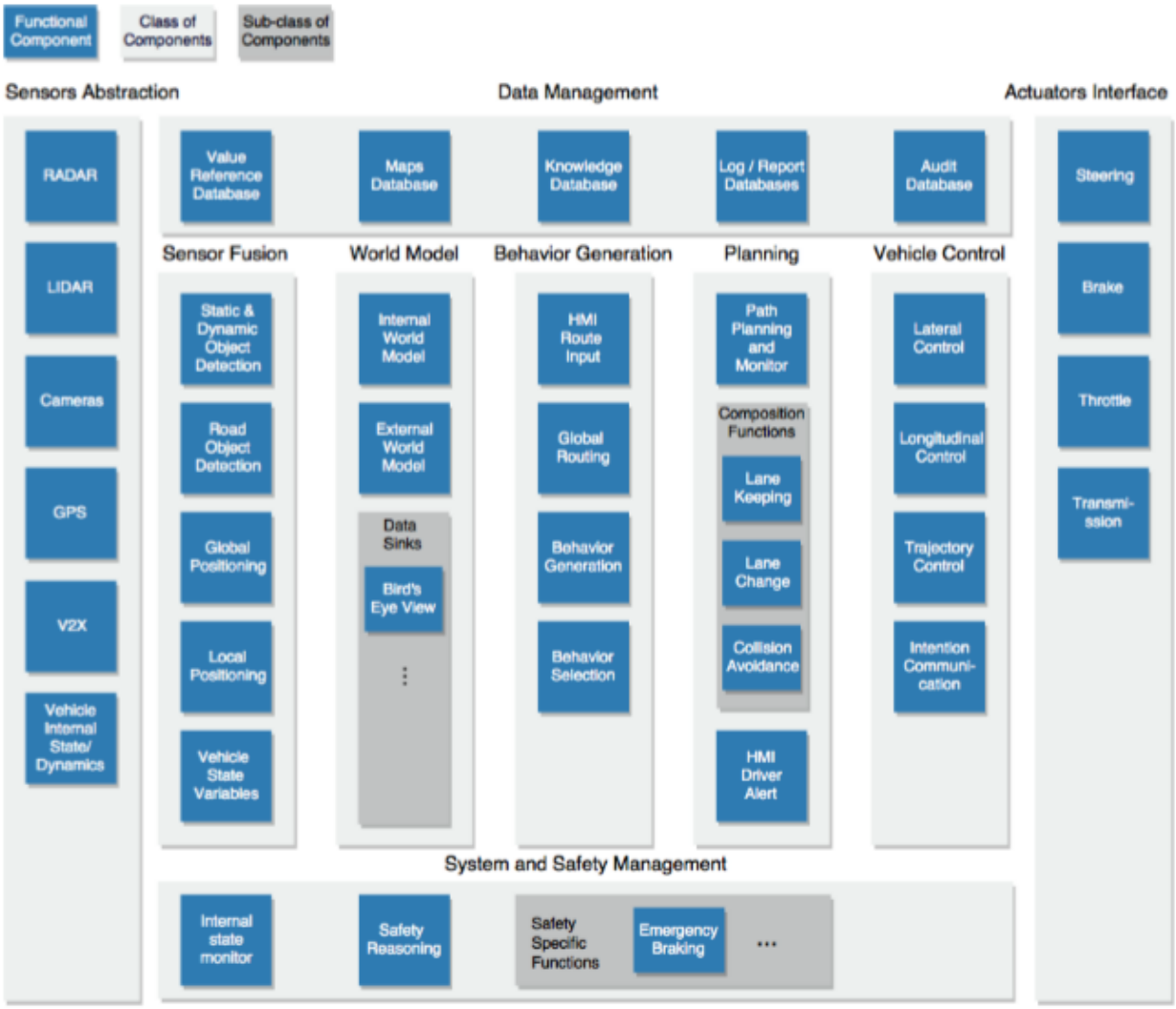

Figura 6 – Arquitertura functional propuesta

- Planificación: La planificación determina cada maniobra que un coche autónomo tiene que ejecutar para satisfacer un determinado comportamiento. El componente de planificación y monitorización de rutas genera una trayectoria libre de obstáculos y compone el plan de implementación de la trayectoria a partir de las funciones de composición desplegadas en el vehículo. El planificación y monitorización de la ruta actúa como un orquestador que decide que funciones son necesarias para completar la trayectoria y coordinarlas hasta que el objetivo sea cumplido o un nuevo objetivo llegue.
- Control del vehículo: Este módulo es esencial para guiar al coche a lo largo de la trayectoria planificada. La tarea de control general se divide en el control longitudinal y control lateral, según los requerimientos del SAE para una función operativa. El bloque de control de trayectorias toma una trayectoria (generada en el nivel de monitorización y planificación de rutas) como entrada y controla el módulo lateral y longitudinal. Por ejemplo, el control lateral computa el deseado angulo de dirección (steering angle) dado las propiedades dinámicas del coche y la trayectoria objetivo, y el control longitudinal recibe el estado longitudinal deseado y decide que acción tomar, si frenar, acelerar, etc.
- Interfaces del actuador: Este módulo transforma la información proveniente de la capa de control del vehículo en comandos del actuador.
- Gestión de datos: Los coches autónomos manejan una cantidad gigantesca de datos. Sin olvidar decir que muchos de estos datos requieres un procesamiento en tiempo real. La localización global requiere un almacenamiento de mapas interno; los algoritmos de decisión inteligentes y reconocimiento de patrones requieren modelos entrenados (base de datos de conocimiento); el informe interno de estado requiere mecanismos avanzados de registro (base de datos de registro). Las bases de datos de informes de registro también se utilizan para almacenar los datos necesarios para ajustar y mejorar los algoritmos inteligentes.
- Gestión de sistemas y seguridad: Este bloque maneja los mecanismos de seguridad funcional (detección y dentión de fallos) y las preocupaciones de seguridad del tráfico. La Figura 6 solo muestra los componentes que detectan fallas de funcionamiento y activan el comportamiento de seguridad, pero no mecanismos de redundancia. A medida que aumenta el nivel de automatización, es necesario tomar decisiones complejas de seguridad. Comenzando con el nivel 3, el vehículo se convierte totalmente responsable de la seguridad del tráfico. Aunque aún no está claro como se estandarizará e implementará el razonamiento de seguridad en los vehículos futuros.

La industria automotriz tiene altas restricciones de seguridad functional impuestas por el cumplimiento obligatorio de la norma ISO 26262. El objetivo de la seguridad functional es evitar cualquier lesión o mal funcionamiento de los components en respuesta a insumos, hardware o cambios ambientales. Además, hasta el momento no está claro cómo se comportarán los vehículos autónomos en caso de que no se pueda evitar un accidente y qué riesgos minimizar. Sin embargo, es esperado que las normas de seguridad futuras incluyan especificaciones para el comportamiento de seguridad.

2.3. ROS

El entorno de simulación que se empleará en este trabajo será ROS. Su origen se remonta hasta el 2007, cuando el Laboratorio de Robótica e Inteligencia Artificial de la Universidad de Stanford inicio un proyecto que daría soporte software para su robot Robot Stair. Desde 2008 el proyecto fue llevado por la empresa Willow Garage hasta 2013, cuando le ceden a Open Source Robotics Foundation los derechos. Hoy en día ROS es un software libre bajo términos de licencia BSD, es decir, que existe una libertad para su uso comercial e investigador.

ROS ha sido diseñado pensando en el código abierto, de manera que los usuarios sean capaces de elegir la configuración de herramientas y librerias que interactúen con el centro de ROS para que los usuarios puedan cambiar sus pilas (stacks) de fostware para que se ajusten a su robot y a su área de aplicación. ROS es un conjunto de herramientas rico, un amplio conjunto de capacidades por paquetes.

Además, ROS posee un carácter distributivo, es decir, que existe una gran comunidad que contribuyen al desarrollo de ros mediante los paquetes, hoy en día hay miles de paquetes públicos. Estos paquetes van desde una calidad industrial hasta pruebas de conceptos básicas.

ROS no solo te permite desarrollar tus propios proyectos, sino también la oportunidad de colaborar y contactar con diferentes expertos en la materia en todo el globo.

ROS cuenta con muchas aplicaciones como:

- Percepción
- Identificación de objetos
- Reconocimiento y segmentación
- Seguimiento de objetos
- Percepción de profundidad mediante el uso de cámaras (Visión estéreo)
- Movimientos
- Robots móviles
- Control y planificación
- Automatizacion
- Etc

2.3.1 Conceptos básicos

ROS tiene varios conceptos básicos. **Roscore** es uno de ellos, es en realidad el bloque mas importante de toda la estructura de ROS, ya que si quieres trabajar con ROS obligatoriamente tienes que estar este nodo en funcionamiento. Luego es el programa que se encargará de controlar la comunicación entre todas las partes de la aplicación. Cuando lanzamos roscore en nuestro sistema, en realidad se está ejecutando tres procesos diferentes:

- Parameter Server: Es una base de datos compartida entre nodos que permite el acceso comunitario a información estática o semiestática. Datos que no cambian con frecuencia y a los que se accedera con poca frecuencia son los datos que normalmente quedan almacenados en el parameter server. El parameter server es implementado usando XMLRPC, lo que significa que su API es accesible a través de bibliotecas XMLRPC normales.
- Master: Es quien proporciona servicios de denominación y registro al resto de los nodos del sistema ROS. La función del máster es la de permitir que los nodos ROS individuales se puedan localizar entre si y una vez que se hayan comunicado puedan comunicarse entre si peer-to-peer.
- Rosout: Es el nombre del mecanismo de informes de registro de la consola en ROS. Comprende varios componentes. El nodo *rosout* para suscribir, registrar y volver a publicar los mensajes. El topic */rosout*. El topic */rosout_agg* para suscribirse a un feed agregado. *Rosgraph_msgs/Log*, que define campos estándar, así como niveles de verbosidad. *Clientes API* para facilitar el uso fácil del mecanismo de informes rosout. Y las herramientas *GUI*, como *rqt_console*, para ver los mensajes de registro de la consola.

Además de roscore tenemos los **nodos**, que son los procesos ejecutables que están incluidos dentro de los paquetes, lo mas normal es que se utilicen varios nodos en un mismo programa. El tener nodos nos permite aplicar técnicas de escalabilidad, lo que implica que es mas fácil realizar e implementar programas. También tenemos los **tópicos**, que son los nombres que identifican el contenido de un mensaje y estos se enrutan de dos formas, un publicador (publisher) y otro suscriptor (suscriber). Puede haber varios editores y suscriptores concurrentes a un mismo tópico, y un único nodo puede publicar y/o suscribirse a múltiples tópicos. Para que la comunicación pueda establecerse con éxito es necesario que tanto el publicador como el suscriptor utilicen el mismo tipo de mensaje y el nombre del topic sea el mismo.

Los **mensajes** es lo que se envían los nodos cuando se están comunicando. Un mensaje es una estructura de datos, que pueden incluir estructuras arbitrariamente anidadas y matrices (como las estructuras de C). Y por último están los **servicios**. Es la arquitectura encargada de realizar la comunicación entre nodos, usan dos tipos de mensajes, una para la solicitud y otro para la respuesta proporcionada por el otro nodo a esa petición.

2.4. Gazebo

Gazebo es un simulador de robotica 3D de código abierto que está integrado perfectamente con ROS. Gazebo nos permite virtualizar nuestro robot o vehículo en nuestro caso, así como el entorno sobre el que va a funcionar permitiendo llevar a cabo pruebas simuladas en un entorno controlado antes de ponerlo en práctica en la vida real. Con esto nos ahorramos dinero y tiempo a la hora de crear un código o programa para nuestro robot, o simplemente testear el comportamiento que tendría nuestro vehículo en cirtas circunstancias.

Gazebo puede usar varios motores físicos de alto rendimiento, como ODE, Bullet, etc. Normalmente se usa ODE que proporciona una representación realista de entornos que incluyen iluminación, sobras y texturas de alta calidad. También puede modelar sensores que ven el entorno simulado.

ROS y Gazebo tienen un amplio abanico de sensores de diferentes marcas y con funciones muy diferentes. Cada uno de estos tiene un apartado en la pagina web ROS o Gazebo. En caso de que no exista documentación sobre el sensor que necesitas, puedes generarla tu mismo o pedir ayuda a la comunidad que siempre está activa. Algunos de los sensores que se pueden encontrar son los siguientes:

- Sensores 2D
  - Hokuyo Scanning range Finder
  - Una gran gama de los sensores SICK lasers
  - Leuze rotoScan laser rangefinder driver (ROD-4, RS4)
  - TeraRanger Multiflex
  - TeraRanger Hub Evo & Tower
  - Neato XV-11 Laser Driver
  - RPLIDAR 360 laser scanner Driver (python)
  - NaviRadar
- Sensores 3D
  - Argos3D P100 ToF camera
  - Basler ToF ES camera
  - DUO3D™ stereo camera
  - Forecast 3DLaser with SICK LIDAR
  - Intel® RealSense
  - SICK lasers
  - Sentis ToF M100 camera
  - Velodyne 3D LIDAR
  - Livox 3D LIDAR
- Reconocimiento de audio
  - Baidu_speech
  - Hark
  - Rospeex
  - Pocketsphinx
- Camaras
  - Allied Vision Tech GigE cameras

  - IEEE 1394 Digital Camera
  - Camera_aravis
  - Canon_gphoto
  - GeViCAM stereo camera
  - Prosilica Camera
  - WGE100 camera
- Sensor de fuerza, par y tacto
  - Wacoh-Tech DynPick
  - Leptrino force/torque sensor
  - Nano 17 6-axis force/torque sensors
  - Schunk LWA 3 Force Torque Controller base don ATI Mini 45
- GPS/IMU
  - Applanix Position and Orientation System for land Vehicles
  - Bosh Sensortec BMA 180 3-axis accelerometer
  - DUO3D™ IMU sensor
  - Microstrain 3DM-GX3-45
  - Oxford technical solutions GPS/IMU products
  - Razor´s IMU 9 DOF
  - Swiftnav Piksi RTK-GPS
  - Xsens MTx/MTi/MTi-G devices
  - RT Corp. USB Output 9-axis IMU sensor module
- Fuentes de alimentación
  - Carnetix CNX-P2140 DC-DC power supply
  - Mini-Box M4-ATX power supply
- Interfaces de sensores
  - ArbotiX RoboController
  - Arduino Sensor Interface Module
  - Phidgets Interface
  - Lego NXT Sensors
  - Roboard_sensors
  - Rosserial_arduino
  - Sensoray 626 analog and digital I/O
  - Shadow RoNeX
- Sensores de velocidad
  - OmniPreSense Radar Sensor
  - Sick RMS3xx Radar Sensor

Estos son solo algunos ejemplos, existen también sesores de RFID, de captura de movimiento o ambientales, y más.

Como se puede apreciar existe una gran variedad de sensores que se puede usar en ROS/Gazebo y es por ello por lo que se ha elegido este entorno para las simulaciones. Pero también porque Gazebo ha sido usado como entorno de simulación para diferentes retos y competiciones tecnológicas. Como:

- DARPA Robotics Challenge
- NASA Space Robotics Challenge
- Toyota Prius Challenge
- Agile Robotics for Industrial Automation Competition (ARIAC)
- DARPA Service Academy Swarm Challenge (SASC)
- DARPA Subterranean Challenge o Virtual RobotX Competition (VRX)

Todos fueron competicones muy importante a nivel mundial, y el hecho que Gazebo haya sido uno de los simuladores para las pruebas, nos dice que es un simulador de alta importancia para el desarrollo de tecnologías. Es por eso que se ha elegido Gazebo y ROS para desarrollar este trabajo, además es una buena oportunidad para entrar en contacto con estos programas.

2.5. Python

Python es:

- Interpretado: Es decir, se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución
- Dinámico: Las operaciones realizadas en tiempo de compilación pueden realizarse en tiempo de ejecución.
- Multiplataforma: Está disponible para Windows, Linux/UNIX, MacOS X, y otros, luego no habría problemas si se cambia de plataforma de trabajo, o se trabaja con variasa a la vez.
- Multiparadigma: Permite crear programas usando más de un estilo de programación, como programación functional, imperativa y orientada a objetos.

Además, python tiene una sintaxis legible lo que hace que la curva de aprendizaje sea rápida, además es gratuito y funciona muy bien con ROS.

Python también cuenta con varias librerías, tipos de datos y funciones incorporadas en el propio lenguaje lo que facilita la programación y no empezar desde cero.

Python puede ser aplicado en diversos campos de la tecnología, como machine learning, desarrollo de software, visión artificial, navegación autónoma, inteligencia artificial, etc. Python está en todas partes y se está convirtiendo en el lenguaje de programación más usado en el mundo según muestran las estadísticas, e irá creciendo en los próximos años según la página Stack overflow figura 7.

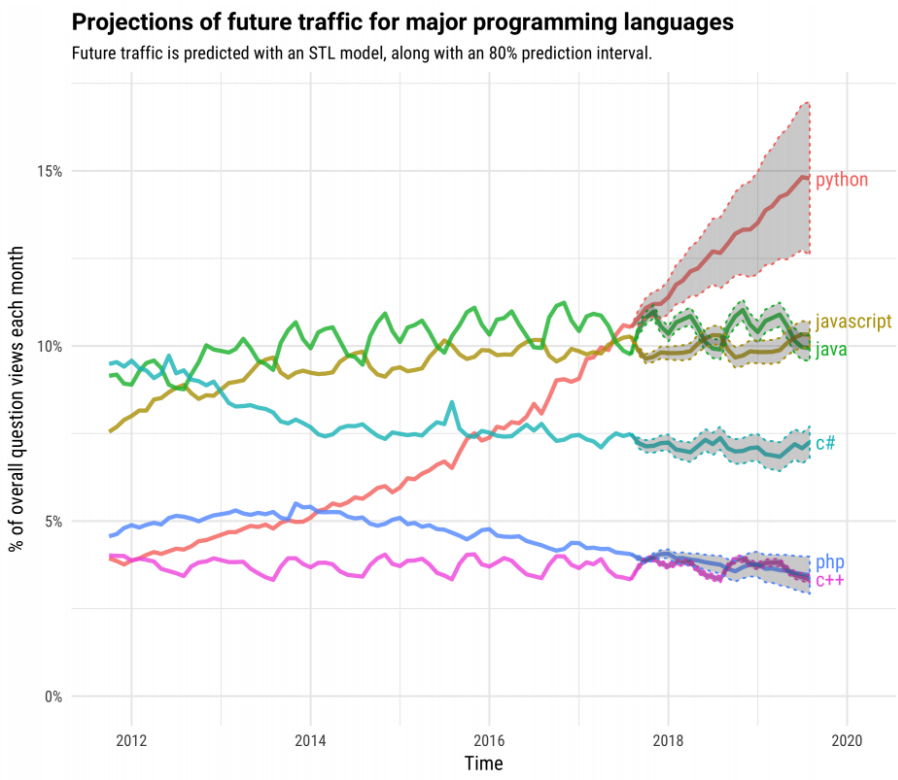

Figura 7 – Proyección de la popularidad de los lenguajes de programación

También hay que mencionar que Python está siendo usado por grandes compañías tecnológicas hoy en día como uber, Paypal, Google, Reddit, entre otros.

Por todo esto y que es un lenguaje de programación nuevo para mi, se ha escogido Python como lenguaje de programación.

3 DINÁMICA DEL VEHÍCULO

La dinámica de vehículos estudia el comportamiento dinámico de los vehículos terrestres. Es una parte de la ingeniería basada en la mecánica clásica, pero también puden verse involucradas otras áreas tales como la física del estado sólido, mecánica de fluidos, ingeniería eléctrica, comunicación, teoría de control, etc. El estudio dinámico de un vehículo se divide en dos análisis en general, el primero de ellos es la dinámica lateral, que estudia las fuerzas laterales de un vehículo que son responsables de que el vehículo pueda girar en una curva. El segundo es la dinámica longitudinal que estudia las fuerzas de propulsión y resistencia al movimiento.

3.1. Dinámica longitudinal

La dinámica longitudinal abarca el análisis del comportamiento del vehículo circulando en línea recta o en curvas de gran radio. El objetivo es establecer la ecuación fundamental del movimiento longitudinal (ejeX) que, a partir de la potencia del vehículo, permitirá predecir sus prestaciones: pendiente, velocidad y aceleración máxima. Estas prestaciones están limitadas por la adherencia neumático-calzada y el conjunto motor-sistema de transmisión.

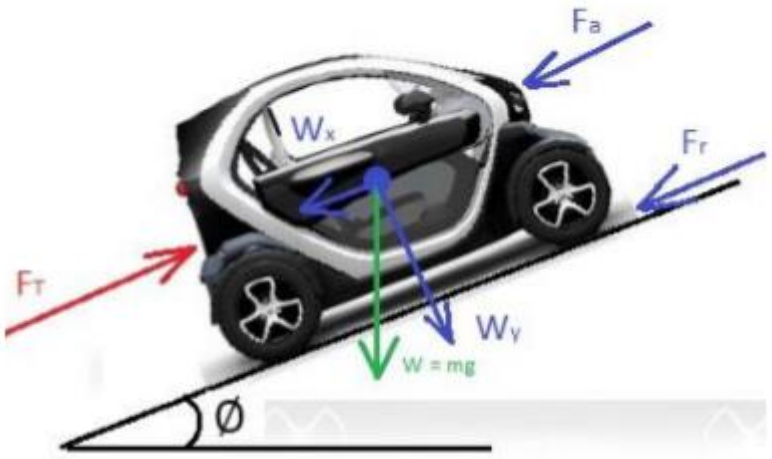

Figura 8 – Acción de fuerzas longitudinales

Para modelar el movimiento longitufinal de un vehículo se analiza las fuerzas que actuan sobre su centro de gravedad y se realiza un equilibrio de esfuerzos según la segunda ley de Newton (Ecuación 3.1).

$$\sum F_x = m.a_x = Fuerza\ tractora - Fuerzas\ resistentes \tag{3.1}$$

La fuerza tractora es generada por el motor del vehiculo y transmitida por los neumáticos al entorno. Por otra parte, las fuerzas resistentes son las que se oponen al movimiento del vehículo, en este sentido se tiene:

- Resistencia aerodinámica al avance: La resitencia aerodinámica de la circulación del aire por el interior y el flujo exterior del vehículo. La resistencia aerodinámica puedes ser calculada mediante la ecuación (3.2).

$$F_{xa} = 0.5.\rho.C_x.A_f.V^2 \tag{3.2}$$

Donde:

ρ Es la densidad del aire con un valor de 1.225 Kg/m^3 a 25ºC y 1 atmosfera

C_x Es el coeficiente aerodinámico

A_f Es el area frontal en metros caudrados

V Es la velocidad en m/s

El área frontal de un coche se puede calcular de la siguiente manera, $A_f = f.base.altura$, donde f es una variable que puede estar entre 0.8 y 0.85,

Y el coeficiente aerodinámico depende de cada coche, además normalmente es proporcionado por el fabricante.

- Resistencia gravitaroria: La resistencia a la rodadura se define por la deformación del neumático sobre la superficie debido a la carga que soporta (Ecuación 3.3).

$$R_r = R_{rd} + R_{rt} = (f_0 + f_v.V^n).P.\cos(\theta) = f_r.P.\cos(\theta) \quad (3.3)$$

Donde:

P Es el peso del vehículo

f_0 , f_v Son parámetros que dependen de la presión de inflado de los neumáticos

n Es un factor con valores cercanos a 2 o 2,5

V Va en Km/h

θ Ángulo de inclinación de la carretera

f_r Magnitud adimensional que expresa la fuerza opuesta al movimiento por cada unidad de carga soportada.

Si se supone que el coche rodará por hormigón y la presión de inflado será próxima a 179 kPa entonces f_r se puede aproximar a:

$$f_r = 0{,}01.\left(1 + \frac{V}{160}\right) \quad \rightarrow \quad 0{,}01 \leq f_r \geq 0{,}02$$

En muchos casos se puede despreciar el efecto de la velocidad y tomar un valor medio de:

Tipo de Vehículo	Superficie		
	Asfalto	Dureza media	Arena
Turismos	0.015	0.08	0.3
Camiones	0.012	0.06	0.25
Tractores	0.02	0.04	0.2

Figura 9 – valores de f_r aproximados según el vehículo y el asfalto

- Resistencia gravitatoria: Es fuerza solo toma importancia cuando el coche está circulando por una carretera con pendiente y viene determinado por la ecuación (3.4).

$$R_g = P.sen(\theta) \quad (3.4)$$

Si el angulo θ es menor que cero, la fuerza será negativa (propulsora).

3.2. Dinámica lateral

Antes de hacer el análisis dinámico lateral, como nuestro vehículo circulará a velocidades bajas y considerando aceleraciones laterales bajas y curvas amplias, se pueden hacer una serie de suposiciones:

- Comportamiento lineal de los neumáticos en deriva.
- Ángulos de dirección iguales en ambas ruedas directrices.
- Efectos dinámicos como deformación de las suspensiones, transferencia de carga, variaciones de los ángulos de dirección, etc. Despreciables.
- Además, el modelo del comportamiento direcciónal puede simplificarse aún más si se considera que los ángulos de deriva de los neumaticos en un mismo eje son iguales, lo que a su vez impone que las fuerzas laterales soportadas por los neumáticos de un mismo eje también sean iguales.

Luego, el comportamiento de los neumáticos de un mismo eje puede representarse como uno equivalente situado en el centro de cada eje que soporta la fuerza lateral de ambos con el mismo ángulo de deriva.

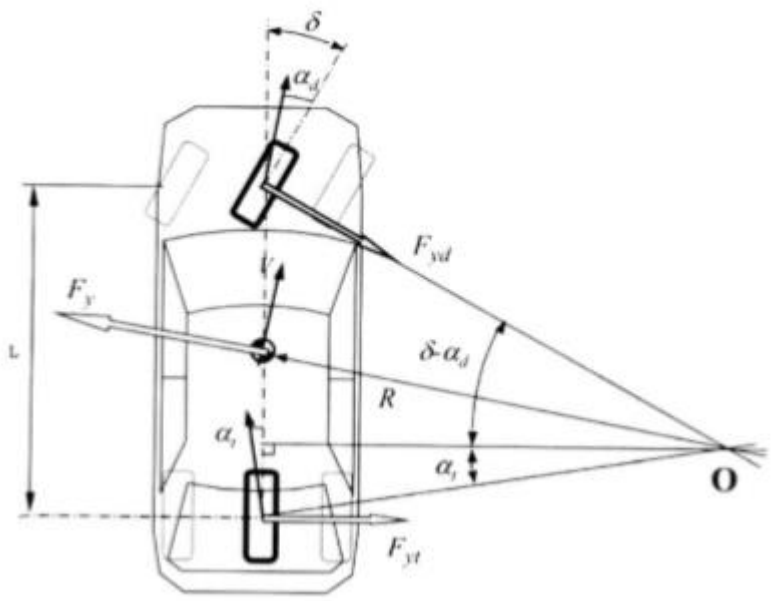

Figura 10 – Modelo simplificado para el análisis del comportamiento direccional

De la representación gráfica se puede deducir que

$$\delta - \alpha_d + \alpha_t = \frac{L}{R}$$

De donde el ángulo de dirección

$$\delta = \alpha_d - \alpha_t + \frac{L}{R}$$

Como los ángulos de dirección son pequeños, se puede suponer que las fuerzas laterales que soporta cada eje serán función de los pesos sobre cada uno de ellos y de la aceleración lateral a la que están sometidos.

$$F_{yd} = m_d.\frac{V^2}{R} = \frac{P_d}{g}.\frac{V^2}{R} \qquad\qquad F_{yt} = m_t.\frac{V^2}{R} = \frac{P_t}{g}.\frac{V^2}{R}$$

Teniendo en cuenta que $\alpha = F_y/C_\alpha$, los ángulos de deriva en cada eje serán

$$\alpha_d = \frac{F_{yd}}{C_{\alpha d}} = \frac{P_d}{C_{\alpha d}} . \frac{V^2}{g.R} \tag{3.5}$$

$$\alpha_t = \frac{F_{yt}}{C_{\alpha t}} = \frac{P_t}{C_{\alpha t}} . \frac{V^2}{g.R} \tag{3.6}$$

Con lo que el ángulo de guiado de las ruedas directrices se puede escribir como

$$\delta = \frac{L}{R} + \left(\frac{P_d}{C_{\alpha d}} - \frac{P_t}{C_{\alpha t}}\right) . \frac{V^2}{g.R}$$

Y si se define el coeficiente de viraje K_v como $K_v = \frac{P_d}{C_{\alpha d}} - \frac{P_t}{C_{\alpha t}}$, entonces

$$\delta = \frac{L}{R} + K_v . \frac{V^2}{g.R} \tag{3.7}$$

Donde K_v es igual a 0 si el vehículo es neutro, mayor que cero si es subvirador y mayor que cero si es sobrevirador.

4 SENSORES Y ACTUADORES

En este apartado se propondrán diferentes sensores que son necesarios para que un coche sea autónomo. Y también se hablará de los diferentes actuadores que hace falta incluir en el Renault Twizy para dotar al coche de autonomía y poder controlarlo. Este apartado también va enfocado para poder realizar un presupuesto de los componentes necesarios a la hora de convertir el Renault Twizy en autónomo.

4.1. Sensores

Para que un vehículo funcione apropiadamente sin la intervención del ser humano, se necesitan una serie de sensores que sean capaces de captar e interpretar el entorno del vehículo.

- Laser 2D:

Este sensor nos permitirá detectar objetos hasta los 8 metros de distancia tanto por la parte delantera, como de la trasera del vehículo. Este laser nos permitirá supervisar una gran superficie, pero nos dejara algún punto ciego que serán cubiertos por las camaras ToF.

Figura 11 – SICK LASER TIM551

- Camaras ToF:

Estas camaras nos servirán como suplemento a los laseres 2D de nuestro coche y supervisarán los laterales del vehículo. Si bien son poco usadas en los coches autónomos, es una buena opción para detectar objetos a corta distancia. ToFs permiten detectar objetos con un alto número de cuadros por segundo, son poco afectados por las condiciones de luz y sombras, y sobre todo tienen un costo aceptable.

Figura 12 – ToF Sentis3D-M420

Estos dos sensores se encargarán de la percepción de corto alcance. Todo objeto detectado en este aro de corto alcance se podría suponer como un riesgo y daño inherente para las acciones del vehículo, por lo que estos sensores son ordenados por la unidad de procesamiento en tiempo real.

- LIDAR 3D:

Este dispositivo permite conocer en tiempo real la posición precisa de millones de puntos, y la distancia entre cada punto y el foco LIDAR. Esto nos permite detectar objetos y saber exactamente la distancia a la que se encuentra cada uno de ellos con gran precisión. De esta manera se pueden prever el movimiento de otros vehículos o peatones cuya trayectoria se cruzará con la del coche.

El LIDAR es cuestionado e incluso algunos casos es tachado de innecesario. Como ejemplo esta TESLA, que durante el Tesla Autonomy Day del 22 de abril de 2019, explicó porque no es necesario el LIDAR. Pero, por otra parte, Waymo, considerado el líder de los coches autónomos, recientemente ha desarrollado su propio LIDAR, que es el que se incluirá en esta selección de sensores.

Figura 13 – Laser Bear Honeycomb

Este LIDAR 3D nos permitirá establecer trayectorias (cambios de carril, evitación de obstáculos, reducción de la velocidad dependiendo de las condiciones de tráfico, etc.), seguimieno de objetos, clasificación de objetos y predicción del comportamiento de otros conductores.

- Camara:

También es indispensable tener una cámara a color colocada en el techo del vehículo. Esta cámara será de gran utilidad, ya que, servirá para detectar la carretera, carriles, líneas viales y señales de tráfico. Además, la visión es capaz de clasificar objetos, como distinguir entre un coche y un peatón o una rueda.

Figura 14 – Prosilica GT1290

- Localización:

Para la localización del vehículo se usaría un sistema basado en una unidad RTK (Real-Time-Kinematic) y una IMU (Inertial Measurement Unit). La navegación por satélite RTK consiste en una técnica usada para mejorar la posición de los datos obtenidos de los sistemas basados en satélites como GPS, Galileo, etc. Este sistema RTK puede proporcionar una precisión por debajo de los 2 cm.

Figura – 15 – EMILID RTK GNSS Receiver

El sistema de localización cancelaría el posicionamiento RTK cuando el numero de satélites detectados es menor que cuatro o cuando el HDOP (Dilución de Precisión para el posicionamiento horizontal) es mayor que cinco. En estos casos entraría en juego el IMU (Inertial measurement unit) para llevar a cabo la tarea de localización del vehículo.

Figura 16 – NAV440CA-200-2

4.2. Actuadores

Además de añadir los componentes descritos en el subapartado anterior, también es necesario realizar unas modificaciones en el sistema de frenado, sistema de dirección y el acelerador del vehículo. Pero, también es necesario llevar a cabo estas modificaciones sin que se vea afectado la conducción manual del vehículo, ya que, será necesario de un piloto de seguridad en las pruebas que se haga en el desarrollo autónomo del Reanult Twizy en futuros proyectos.

Una de las modificaciones que se haría sería la del sistema de dirección. La solución es un problema mecánico básico, que consiste en intervenir en la columna de la dirección. Un conjunto mecánico motor-reductor podría ser ensamblado a la columna de dirección del vehículo mediante engranajes, permitiendo que el eje gire. El eje original no ha sido modificado, luego se podría tener tanto una conducción manual y autónoma. El sistema de dirección implementado estaría compuesto por un motor Maxon EC60fl, con un reductor GP52C, el Encoder MILE de 4096 cpt con 2 canales y un controlador electrónico EPOS4.

La siguiente modificación se realizaría en el freno. El mecanismo implementado sería parecido al mecanismo de la dirección. Se ha implementado un conjunto mecánico con un motor que tiene un engranaje acoplado a una leva que presionaría el pedal del freno con un control de fuerza. El sistema de frenado estaría formado por un motor Maxon EC60fl, con un reductor GP52C, el Encoder MILE de 4096 cpt con 2 canales y un controlador electrónico EPOS4.

Antes de empezar con la última modificación del sistema, hay que mencionar un programa de código abierto llamado OVMS (Open vehicle Monitoring System), que nos serviría de intermediario para acceder a todos los niveles de la ECU del motor, de la que se hablará a continuación.

Y por ultimo, vendría la modificación del acelerador. En esta parte también se podría optar por una solución mecánica parecida propuesta en las modificaciones tanto de la dirección como del freno. Sin embargo, hay una solución a nivel de software que se podría llevar a cabo. La ECU que controla el motor del Renault Twizy es producida por una compañía llamada Sevcon, que fabrica controladores de motor y componentes de sistemas para vehículos eléctricos. Un modelo instalado en el Twizy es el controlador Sevcon Gen4. Por defecto el Devcon Gen4 convierte el voltaje de la posición del pedal del acelerador en un valor numérico escalado que controla la potencia de salida del motor eléctrico, luego si se accediera a esos parámetros del controlador, se podría modificar la función de conversión para alterar el comportamiento del vehículo. Bastaría con desarrollar un módulo software que emulara el funcionamiento.

Además, existen dos interruptores booleanos que describen el estado de los cambios del vehículo como se aprecia en la tabla 1. De forma predeterminada estos valores se establecen mediante unos interruptores físicos junto al volante

Action	*Forward*	*Reverse*
Neutral (No torque)	0	0
Negative Torque	0	1
Positive Torque	1	0
Unauthorized	1	1

Tabla 1 – Gear State

Todo este sistema montado, tanto para el acelerador como para los cambios del vehículo, funcionarían en paralelo junto con el sistema original, luego un control manual y automático del acelerador del vehículo es posible.

5 DISEÑO

En este capítulo se discutirá el modelo en 3D del vehículo que se implementará en las simulaciones. Así como el diseño de los diferentes controles, control longitudinal, control lateral.

5.1. Modelado 3D

El modelo en 3D del Renault Twizy en Gazebo se ha desarrollado en XML, en un archivo de extensión ".URDF", a través del editor de texto "*Atom*". Para la parte visual del vehículo, se ha usado un modelo 3D de la comunidad.

El modelo en 3D del Renault Twizy que se usara para la parte visual implementada en gazebo se llevo a cabo gracias a un modelo en 3D obtenido en la web *GRABCAD ACOMMUNITY*. El archivo obtenido es de extensión .3dm y Gazebo solo soporta archivos con extensión .STL o .DAE. Luego lo primero que se tiene que hacer es transformar el archivo 3dm a STL, y para ello se usa el programa *CAS Exchanger*.

Ya tendríamos nuestro modelo 3D del Renault Twizy, pero para que se pueda mover se deberá tener por separado las ruedas del chasis del vehículo y así poder decirle al programa cuales son nuestras ruedas motrices y directrices. Para poder llevar esto acabo se tendrá que hacer uso de otro programa *Atuodesk Netfabb*. Tanto este programa como el anterior mencionado tienen una versión de prueba gratuita.

Una vez realizado todos los cortes, tenemos que crear un archivo URDF. En este archivo URDF tenemos que definir el eje de giro de cada una de las ruedas (joints), también se debe limitar el par que se puede ejercer en las ruedas motrices que son las traseras, y definir que las ruedas directrices también se puedan desplazar longitudinalmente.

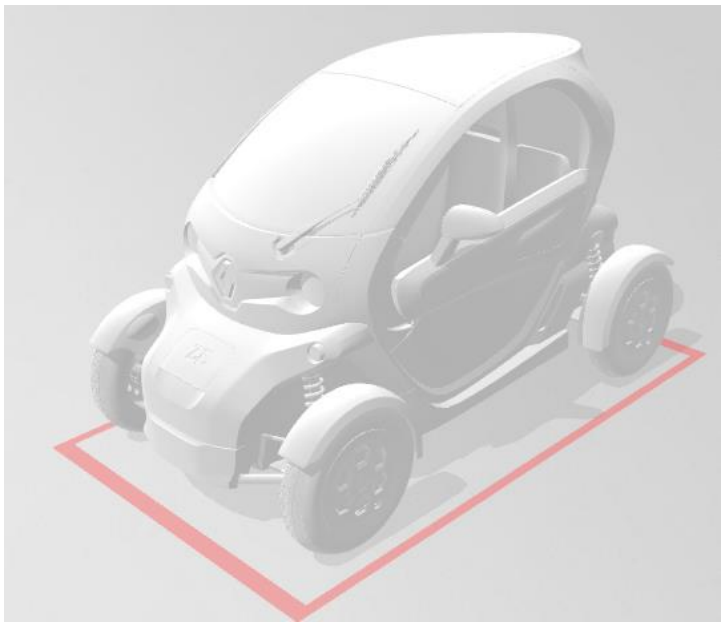

Figura 17 – Modelo 3D Renault Twizy

También se tiene que poner el peso del vehículo y el peso de las ruedas, así como sus momentos de inercia respectivos. Los momentos de inercia lo obtenemos gracias a un programa llamado *Meshlab*, que nos permite calcular rápidamente los momentos de inercia de un objeto en 3D.

Para hacer que las ruedas directrices giren respesto a "z", también hay que definir que así lo hagan, imponiéndoles que tengan un ángulo de giro de ± 0.7 radianes y que la velocidad de giro no sea brusca.

Lamentablemente al tratar de diseñar que las ruedas directrices puedan girar respecto a su eje "z", se ha tenido numerosos fallos que se explicarán en el apartado 7 "Simulaciones en Gazebo".

Para que Gazebo actue en cada una de nuestras articulaciones(joints) se tiene que agregar un bloque llamado <transmission> para cada una de las articulaciones, figura 18. Este bloque describe la relación que hay entre un actuador y una articulación. A parte de esto se tiene que añadir un plugin llamado gazebo_ros_control que lee en todas las etiquetas de <transmission>.

También es necesario un archivo de configuración de extensión "*.yaml*", que es donde se guardan los parámetros que se necesitarán para el controlador y que se cargarán y leerán en el servidor de Parámetros: nombre de las articulaciones del robot a controlar y los valores de los coeficientes de los PIDs, en este caso del Renault Twizy.

El archivo "*.yaml*", será lanzado a través del archivo roslaunch con extensión "*.launch*". En este archivo roslaunch se inician los controladores "*ros_control*", además aquí se carga el modelo 3D del vehículo desde un archivo "URDF", y los nodos necesarios para mandar instrucciones desde un código Python.

```
281  <!-- transmission -->
282
283  <transmission name="tran1">
284    <type>transmission_interface/SimpleTransmission</type>
285    <joint name="left_rear_wheel_joint">
286      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
287    </joint>
288    <actuator name="motor1">
289      <mechanicalReduction>1</mechanicalReduction>
290    </actuator>
291  </transmission>
292
293  <transmission name="tran2">
294    <type>transmission_interface/SimpleTransmission</type>
295    <joint name="right_rear_wheel_joint">
296      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
297    </joint>
298    <actuator name="motor2">
299      <mechanicalReduction>1</mechanicalReduction>
300    </actuator>
301  </transmission>
302
303  <transmission name="right_steering_trans">
304    <type>transmission_interface/SimpleTransmission</type>
305    <joint name="right_steering_joint">
306      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
307    </joint>
308    <actuator name="right_steering_act">
309      <mechanicalReduction>1</mechanicalReduction>
310    </actuator>
311  </transmission>
312
313  <transmission name="left_steering_trans">
314    <type>transmission_interface/SimpleTransmission</type>
315    <joint name="left_steering_joint">
316      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
317    </joint>
318    <actuator name="left_steering_act">
319      <mechanicalReduction>1</mechanicalReduction>
320    </actuator>
321  </transmission>
322
323  <gazebo>
D:\kevin\Ubuntu\CWS\my_robot\urdf\wally.urdf*  215:50
```

Figura 18 – Bloque transmisión en URDF

5.2. Control longitudinal

El control longitudinal que se desarrollará en este trabajo es el de *Cruise and headway control*. Las directrices para este control son las de una conducción en carretera.

El control que se empleará será el de un espacio de estados realimentado. Solo se tendrá como información la distancia "R" entre ambos coches que pueden venir de un sensor de rango montado en el coche y la velocidad del vehículo que se controla v_c, que puede ser obtenida desde la IMU. No se obtendrá otra información, ni de otros coches ni de la infraestructura.

Como se sabe, el movimiento longitudinal de un vehículo viene dado por la ecuación (5.1)

$$m\frac{du}{dt} = F_x - m.g.sin\theta - f.m.g.cos\theta - 0.5.\rho.A_f.C_d.(u_0 + u_w)^2 \tag{5.1}$$

La ecuación (5.1) es no linear en la velocidad, $u(t)$, pero por otra parte es un Sistema dinámico simple, solo se tiene una variable de estado. Sin embargo existen dos dificultaddes, una de ellas es que plantea incertidumbre debido al cambio de peso del vehículo y perturbaciones externas debido a la pendiente de la carretera. Sin embargo, las simulaciones se llevarían a cabo con una masa constante y con pendiente nula, además un buen algoritmo de control tendría que trabajar bien bajo esas condiciones. Así que podemos linealizar la ecuación (5.1) sobre una condición nominal de operación, por ejemplo, en un equilibrio donde $du/dt = 0$ la ecuacion (5.1) puede ser resulta par:

$$F_{x0} = m.g.sin\theta_0 + f.m.g.cos\theta_0 + 0.5.\rho.A_f.C_d.(u_0 + u_w)^2 \tag{5.2}$$

Para los siguientes valores numéricos:

$$g = 9.81m/s,\ u_0 = 20m/s,\ \theta = 0, m = 1000kg,$$
$$\rho = 1.202kg/m3\,, A = 1m2, C_d = 0.5, f = 0.015, u_w = 2m/s;$$

Se obtiene $F_{x0} = 292.6\,N$. Linearizando la ecuación (5.1) sobre el estado de operación especificado usando una expansion de la serie de Taylor se obtiene la ecuación linearizada:

$$\tau.\dot{u}' + u' = K(F' + d) \tag{5.3}$$

Donde las variables incrementales o perturbadas se definen como:

$$u = u_0 + u';\ F_{x0} = F_{x0} + F';\ \ \theta = \theta_0 + \theta';\ \ d = mg(f\,sin\theta_0 - cos\theta_0)\theta';$$

$$\tau_c = (m/(\rho.C_d.A(u_0 + u_w)));\ K_c = (1/(\rho.C_d.A(u_0 + u_w))): \tag{5.4}$$

Y de esta manera se obtienen la constante de tiempo y la ganancia.

El controlador de estado realimentado quedaría de la siguiente manera.

$$\dot{x}_1 = -x_2 + v_l$$
$$\dot{x}_2 = -(1/\tau_c).x_2 + (K_c/\tau_c).(u + w_c)$$
$$\dot{x}_3 = (R - r) = (x_2 - r) \tag{5.5}$$
$$\dot{x}_4 = x_3$$
$$u = -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4$$

Este controlador realimenta los valores medidos $x_1 = R$, $x_2 = V_c$, x_3, la integral de error, $e = d - r$; y x_4 la doble integral del error. Las ganancias del controlador se determinarán por el metodo de pole-placement. Los polos de bucle cerrado se situarán en $s_{1,2} = -\zeta.w_n \pm w_n\sqrt{1-\zeta^2}j$ y $s_{3,4} = -\alpha.\zeta.w_n$. Los valores que se usarán son: $\zeta = 0.9,\ w_n = 0.4$, y $\alpha = 3$ (Ulsoy, A., Peng, H., & Çakmakci, M[8]).

En forma de diagrama quedaría de la siguiente manera:

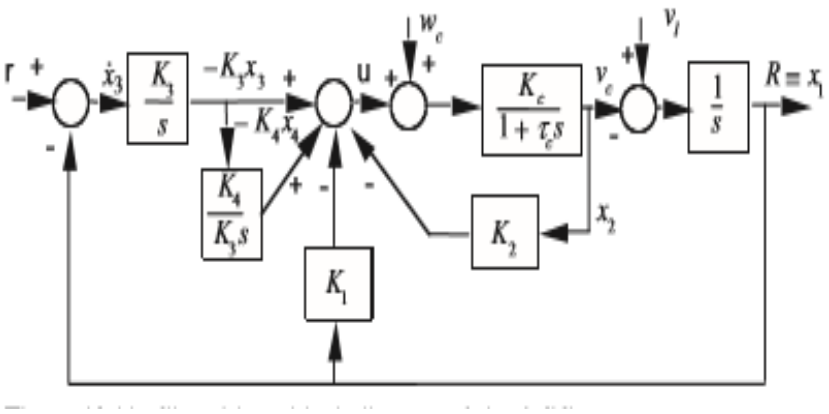

Figura 19 – Diagrama de bloques en bucle cerrado

5.3. Control de dirección

En esta parte se realizará un control de dirección que será testeado mediante el seguimiento de una ruta como se aprecia en la figura 20.

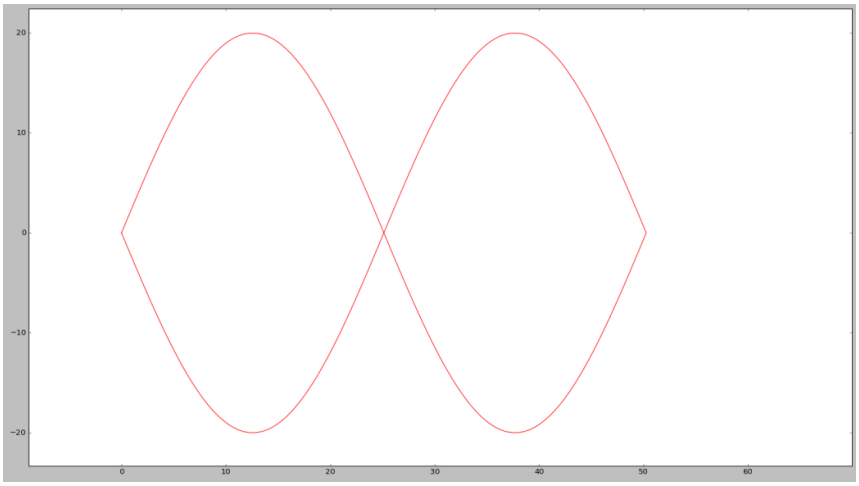

Figura 20 – Ruta de prueba del control de dirección

Para el diseño del control de dirección se usará el método pure-pursuit. Este metodo se basa en la geometría para calcular la curvatura que tendra que hacer el vehículo para llegar a un punto objetivo de la trayectoria. Este objetivo es un punto de la trayectoria que se encuentra a una *lookahead distance* de la posición actual de nuestro vehículo. Se genera un arco entre nuestra posición y la del objetivo, y la longitud de la cuerda de ese arco es la *lookahead distance*.

Imaginemos la Figura 21, donde se tiene al vehículo con sus respectivas coordenadas. El método consiste en calcular la curvatura de un arco circular que conecta la ubicación del eje trasero con el punto objetivo en la trayectoria del vehículo. El angulo de giro de las ruedas se puede determinar usando solamente la ubicación del punto objetivo y el ángulo comprendido entre el vector de dirección del vehículo y el vector de *lookahead*.

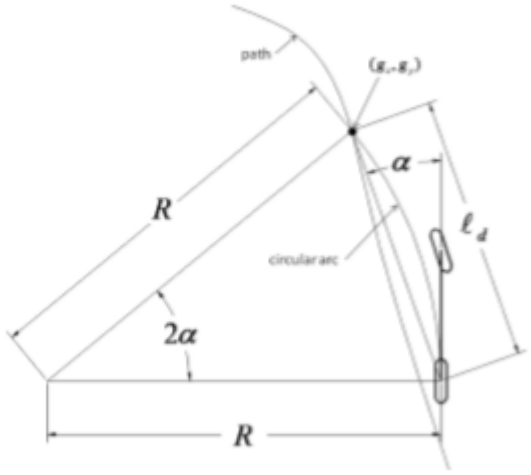

Figura 21 – Geometría Pure-Pursuit

Aplicando la ley de los senos a la Figura 21 se obtiene:

$$\frac{\ell_d}{sin(2\alpha)} = \frac{R}{\sin(\frac{pi}{2} - \alpha)}$$

$$\frac{\ell_d}{2.sin(\alpha).\cos(\alpha)} = \frac{R}{\cos(\alpha)}$$

$$\frac{\ell_d}{sin(\alpha)} = 2.R$$

$$\gamma = \frac{2.sin(\alpha)}{\ell_d} = \frac{2.x}{{l_d}^2} \tag{5.6}$$

Donde x es la coordenada x de el punto objetivo, ℓ_d es la distancia de *look ahead*, y γ es la curvatura del arco circular. Usando el modelo geométrico de la bicicleta en un coche con dirección ackerman, el ángulo de dirección será:

$$\delta = tan^{-1}(\gamma.L) \approx \gamma.L \tag{5.7}$$

La implementación de este algoritmo se daría de la siguiente manera:

- Determinar la posición actual del vehículo
- Encontrar el punto de la trayectoria más cercano al vehículo
- Encontrar el punto objetivo
- Transformar el punto objetivo a las coordenadas del vehículo
- Calcular la curvatura y el ángulo de dirección del vehículo
- Actualizar la posición del vehículo

5.4. Control de estabilidad

Sistema de control de estabilidad del vehículo previene que los coches giren y se desvíen o derrapen, tanto en sobreviraje (oversteer) como en subvirajes (understeer), figura 22.

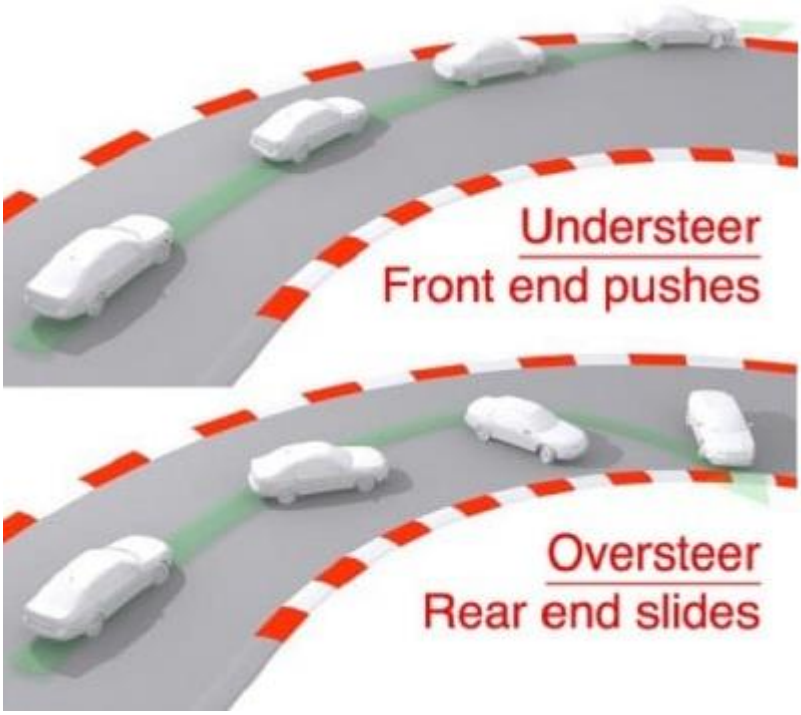

Figura 22 – subviraje y sobreviraje

El funcionamiento del control de estabilidad reside en que está constantemente comprobando las maniobras del conductor al volante y ver si estas se corresponden con el comportamiento del vehiculo. Si en algún momento detecta que la dirección es diferente a la deseada y es crítica, entonces el ordenador reacciona independientemente del conductor.

Existen varios tipos de control de estabilidad propuestos, y que han sido desarrollados por diferentes investigadores:

1. **Frenado diferencial:** Este Sistema utiliza el sisema de frenos del ABS del vehículo para aplicar un frenado diferecial entre las ruedas delanteras y traseras para el control de guiñada.
2. **Steer-by-Wire:** Este sistema modifica el ángulo de dirección del conductor de entrada y añade un ángulo de dirección de corrección a las ruedas.
3. **Distribución de par activo:** Este sistema usa diferenciales activos y tecnología de tracción total para controlar de manera independiente el par de transmisión distribuido a cada rueda y así proporcionar un control activo de la tracción y el momento de guiñada.

Por mucho el Sistema de frenado diferencial ha sido el que más atención ha recibido por la comunida y desarrolladores, y será el que se desarrollara en este trabajo.

El control de estabilidad electrónico (ESC), calcula la velocidad de guiñada y el ángulo de deslizamiento del vehículo que se desea gracias al ángulo de la rueda y la velocidad del vehículo, y la compara con la que tiene actualmente, obtenida de la medición de los sensors. Si la diferencia es muy pequeña no hace nada, pero si la diferencia supera un limite, entonces actuará aplicando freno en las ruedas para crear un par de corrección en la carroceria del vehículo.

Luego el control se caracterizará por la velocidad de guiñada deseada y el ángulo de deslizamiento deseado, ambos pueden ser obtenidos por las siguientes ecuaciones (R. Rajamani[1]).

$$\dot{\psi}_{des} = \frac{\dot{x}}{R} = \frac{v}{L + \frac{m.v^2(l_r.C_{\alpha r} - l_f.C_{\alpha f})}{2.C_{\alpha f}.C_{\alpha r}.L}}.\delta \tag{5.8}$$

$$\beta_{des} = \frac{l_r - \frac{l_f.m.v^2}{2.C_{\alpha r}.L}}{L + \frac{m.v^2(l_r.C_{\alpha r} - l_f.C_{\alpha f})}{2.C_{\alpha f}.C_{\alpha r}.L}}.\delta \tag{5.9}$$

Generalmente hablando, el diseño principal del algoritmo de control del controlador de estabilidad puede estar dividida en tres partes, una de ellas opcionales: (1) Yaw-Rate Following; (2) Slip-angle Estimation and Regulation; and (3) (opcional) rollover or wheel liftoff prevention. En este trabajo solo se trabajará con las dos primeras opciones por simplicidad.

- Yaw-Rate Following: El seguimiento de la velocidad de guiñada es muy importante a la hora de controlar la estabilidad de un vehículo, por ejemplo, cuando el conductor da un giro muy rápido para esquivar un obstáculo, entra en juego el ESC, y se aplicará un freno diferencial para lograr la velocidad de guiñada deseada por el conductor. De esta manera, si es posible calcular la velocidad de guiñada deseada y medir la actual, entonces se podría aplicar un simple algoritmo de control para disminuir la diferencia entre la velocidad de guiñada deseda y actual o medida. El control definido sería el siguiente (Ulsoy, A., Peng, H., & Çakmakci, M[8]):

$$\begin{gathered} If\ \left|\dot{\psi} - \dot{\psi}_{des}\right| \geq \dot{\psi}_{threshold}\ \ and\ \frac{\left|\dot{\psi} - \dot{\psi}_{des}\right|}{\dot{\psi}_{des}} \geq \dot{\psi}_{threshold\_perct} \\ Then\ \ M_{ESC\_yaw} = K_{yaw}\left(\dot{\psi}_{des} - \dot{\psi}\right) \\ Else\ \ \ M_{ESC\_yaw} = 0 \end{gathered} \tag{5.10}$$

- Slip-angle Estimation and Regulation: Se ha visto que el seguimiento de la velocidad de guiñada es importante para el control de estabilidad, pero también se debe tener en cuenta que el ángulo de deslizamiento no sea excesivamente grande. Luego, de la misma manera que en el seguimiento de la velocidad de guiñada, si se asume que el ángulo de deslizamiento del vehículo puede ser estimado o medido, el método de control se puede simlificar a un controlador PD (Ulsoy, A., Peng, H., & Çakmakci, M[8]):

$$\begin{gathered} If\ |\beta| \geq \beta_{threshold}\ \ and\ \ \beta.\Delta\beta > 0 \\ Then\ \ \ M_{ESC\_\beta} = K_{\beta p}.\beta + K_{\beta d}.\Delta\beta \\ Else\ \ \ M_{ESC\_\beta} = 0 \end{gathered} \tag{5.11}$$

  Donde $\beta.\Delta\beta > 0$ garantiza que la ley de control se active solo cuando la magnitud del ángulo de deslizamiento este aumentando.

Una vez obtenido los momentos correctores del control de estabilidad de cada parte, para combinarlos solo basta con sumarlos.

$$M_{vsc\_total} = M_{ESC\_yaw} + M_{ESC\_\beta} \tag{5.12}$$

Y a continuación se muestra en la tabla 2 los parametros de las ecuaciones anteriores (Ulsoy, A., Peng, H., & Çakmakci, M[8]).

Variable	Valor	Unidad
$\dot{\psi}_{threshold}$	0.5	Grados/segundo
$\dot{\psi}_{threshold\_perct}$	2	%
K_{yaw}	0.2 (o 0 apagado)	Slip por rad/s
$\beta_{threshold}$	3	Grados
$K_{\beta p}$	2.5 (o 0 apagado)	Slip por radianes
$K_{\beta d}$	0.1(o 0 apagado)	Slip por radianes

Tabla 2 – Parámetros para el controlador de estabilidad

6 SIMULACIONES

Una vez descrito el diseño de cada uno de los controladores, se dará paso a sus respectivas simulaciones para su validación.

6.1. Entorno de simulación, software usado y programación

Los modelos han sido desarrollados en el lenguaje de programación Python a traves de la interfaz Spyder.

La programación de cada modelo de control en python se ha llevado de manera independiente, pero en escencia tienen la misma estructura, unas variables con los valores del coche definidas, una serie de funciones creadas para cada una de las simulaciones que cumplen una función en específica y finalmente un bucle principal donde se realizan todas las llamadas a las funciones cuando sea necesario.

6.1.1 Control longitudinal

Para el caso del control longitudinal, se parte de las ecuaciones de estado de nuestro Sistema:

$$\begin{aligned} \dot{x} &= A.x + B.u \\ y &= C.x \end{aligned} \tag{5.13}$$

Y la realimentación se expresa de la siguiente manera:

$$u = -K.x \tag{5.14}$$

Luego se calcula la ganancia "K" de la realimentación, para ello se usará el método de "pole placement", y dentro de este se usará el método de Ackermann. Una vez calculada la ganancia, solo nos quedaría por calcular los valores iniciales que tendremos en nuestra simulación.

Valores iniciales de las variables $K_c \; y \; \tau_c$, con ellas los valores de nuestras matrices A, B y C. También calcular la integral del error x_3, la doble integral del error x_4,la velocidad controlada $x_2 = v_c$ y la distancia entre vehículos $x_2 = R$. Con estos valores iniciales calcular también el valor inicial de la entrada a nuestro modelo dinámico del vehículo $u = -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4$.

Una vez definidas y calculadas todas estas variables se realiza un bucle "for" de 120 segundos, tiempo suficiente para nuestra simulación. Aquí se realiza una llamada a la función "odeint" que resuelve el modelo dinámico longitudinal no lineal del vehículo y devuelve el valor de la velocidad controlada $x_2 = v_c$ y el valor de la distancia entre los dos vehículos, Además en este bucle "for" se lleva a cabo las actualizaciones de las variables y se define la actuación del vehículo para el próximo estado.

Por último, se encuentra la función que es llamada por odeint. A esta función se le pasan los valores de la velocidad de coche controlado, la distancia que hay entre ambos coches, el tiempo de simulación y la variable de entrada al modelo dinámico del vehículo u.

En esta función se actualizan los valores de $K_c \; y \; \tau_c$ y se calcula los valores de $\dot{x}_2$ y $\dot{x}_3$ para que luego el odeint devuelva $x_2 \; y \; x_3$. Además, se calcula la velocidad del coche líder en función del tiempo, figura 23.

6.1.2 Control de dirección

Para el caso del control de dirección es algo más simple que el longitudinal, pero algo más largo. Como se ha visto en el apartado 5.4 el cálculo para obtener el ángulo de dirección de nuestro vehículo es un cálculo meramente geométrico donde se tiene como incognita la coordenada x del siguiente punto objetivo respecto a la parte trasera del vehículo. Para lograr esto se tiene que seguir varios pasos.

Pero antes de describir las funciones hay que decir que la trayectoria preestablecida está en dos vectores que almacenan las coordeanas "x" y las coordeandas "y" cada una.

Se tiene una función que calcula el índice del punto objetivo dentro de los vectores que contienen los puntos de la trayectoria. Primero se calcula el índice del punto más cercano hacienda un barrido por los vectores de la trayectoria. Una vez obtenido el índice mas cercano, se calcula la distancia desde la parte trasera del vehículo hasta el punto más cercano obtenido previamente y el siguiente mediante un bucle "while", si la distancia del más cercano es menor que la del siguiente, entonces se sale del bucle y se toma ese indice como referencia para empezar a buscar el punto objetivo look ahead.

Luego se realiza otro bucle while desde el índice cogido hasta la longitud del vector de las coordenadas de la trayectoria, calculando la distancia de cada uno de esos puntos hacia la parte trasera del vehículo. Si esa distancia supera el valor del look ahead establecido "ℓ_d", figura 21, se sale del bucle y se toma ese índice como el índice del punto objetivo, valor que devuelve como función.

Como se ha nombrado en la función anterior, Tambien se tiene una ecuación para calcular la distancia entre un punto de la trayectoria y la parte trasera del vehículo. Esta función simplemente toma los valores "x" e "y" de ambos puntos los resta y aplicando el teorema de Pitágoras se obtiene la distancia entre ambos puntos, devolviendo dicho valor como función.

También se cuenta con una función que simula un controlador proporcional para controlar la velocidad, se le pasa como variables la velocidad deseada y la actual y devuelve la aceleración necesaria para obtener la velocidad deseada, su ganancia proporcional es de: $K_p = 1$.

Otra función es la que calcula el ángulo de dirección del coche δ. En esta función obviamente se calcula el ángulo de dirección del coche con el punto objetivo que indica el índice obtenido en las funciones anteriores, obviamente llamando a la función que calcula el índice del punto objetivo. Primero se calculo en ángulo α, para posteriormente calcular el ángulo de dirección, valor el cuál devuelve como función después de su llamada.

Otra de las funciones usadas es una función que actualiza los valores de las variables del vehículo, como las coordenadas "x" e "y" del vehículo que se usa para representar el vehículo en el plot, su ángulo de orientación "yaw", su velocidad y sus coordenadas "x" e "y" de la parte trasera del vehículo.

Por último, se crea una función para pintar la evolución del vehículo, en forma de flecha.

Ahora se explicará la parte principal del código. Como en el control longitudinal en esta parte principal, que es la funcion *main* del código, se hacen las llamadas a las funciones cuando sea pertinente. Se comienza guardando las coordenadas "x" e "y" de la trayectoria planificada, en 2 vectores separados. También se define los valores iniciales de las variables como la velocidad, la velocidad objetivo y las coordenadas iniciales del vehículo, y se llama a la función que calcula el índice del punto objetivo. A continuación, empieza el bucle *while* que concluye cuando el índice indica las últimas coordenadas de la trayectoria. En este bucle se llama a la función del control proporcional que devuelve la aceleración del vehículo, también se llama a la función que calcula el ángulo de dirección que devuelve dicho ángulo y el índice del actual punto objetivo, luego se actualizan los valores de las variables llamando a la función que actualiza las variables, para finalizar con la llamada a la función que pinta la gráfica animada en cada iteración.

6.2. Control de crucero y seguimiento

Para el testeo del controlador longitudinal se hará uso de un simplepero eficaz ejercicio. Se tendrá dos vehículos, uno líder y otro controlado.

Para empezar, vamos a estipular que la distancia de seguridad entre los dos coches es como mínimo de 100 metros y que la velocidad máxima de nuestro vehículo controlado es de 40 m/s. La simulación durará 120 segundos, ambos vehículos partirán con una velocidad similar de 30 m/s, y distanciados 80 metros entre si, luego la velocidad de nuestro coche controlado tiene que variar para que la distancia entre ambos vehículos tenga el valor de la distancia de seguridad, a continuación, se subirá la velocidad del coche líder gradualmente hasta los 45 m/s, por lo que el coche controlado empezará a subir su velocidad hasta los 40 m/s manteniendo siempre la distancia de seguridad de 100 metros, y una vez alcanzada la velocidad máxima del vehículo controlado la distancia entre los vehículos empezará a crecer.

En la figura 23 se muestra la evolución que tendría la velocidad del coche líder en función del tiempo.

Figura 23 – Evolución de la velocidad del vehículo líder

En la figura 24 se representa la distancia entre vehículos en función del tiempo y además representa la velocidad del coche controlado en función del tiempo, es decir la simulación del control de crucero y seguimiento.

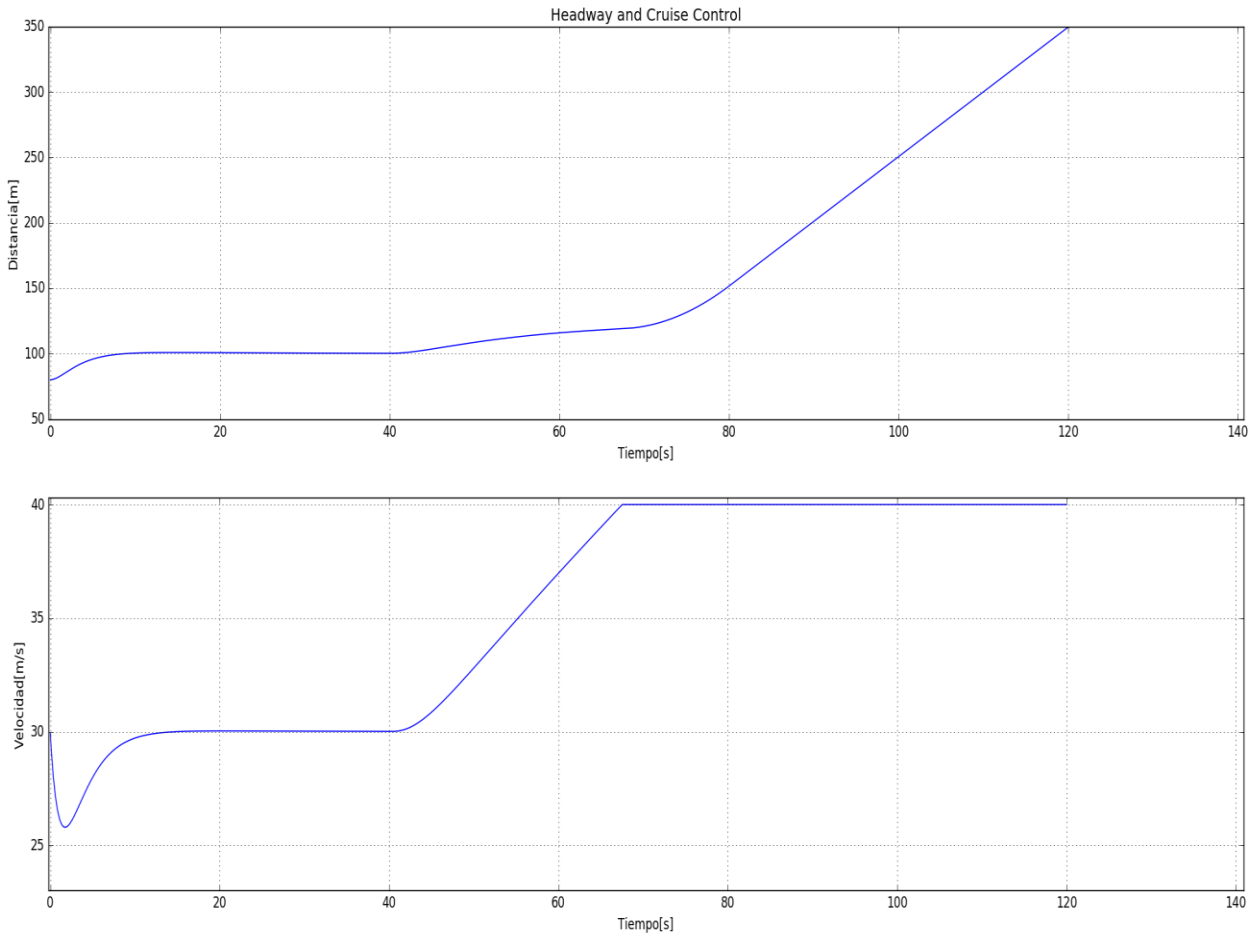

Figura 24 – Headway and Cruise Control

Como se puede observar en las figuras anteriores, el control longitudinal cumple su función correctamente. En los primeros 40 segundos el vehículo controlado primero se ciñe a mantener la distancia de seguridad estipulada, una vez alcanzada dicha distancia sube su velocidad para mantenerla.

También se puede observar en la primera parte de la gráfica de la figura 24 que en los primeros instantes la distancia aumenta de manera no lineal y eso es debido al tiempo de respuesta de nuestro controlador. Algo que se podría mejorar con otro tipo de controlador, pero este sin duda cumple su cometido.

6.3. Seguimiento de trayectoria

Para el testeo del controlador de dirección, se realizará un seguimiento de una ruta preestablecida en forma de ∞ algo deformada, como se muestra en la figura 25.

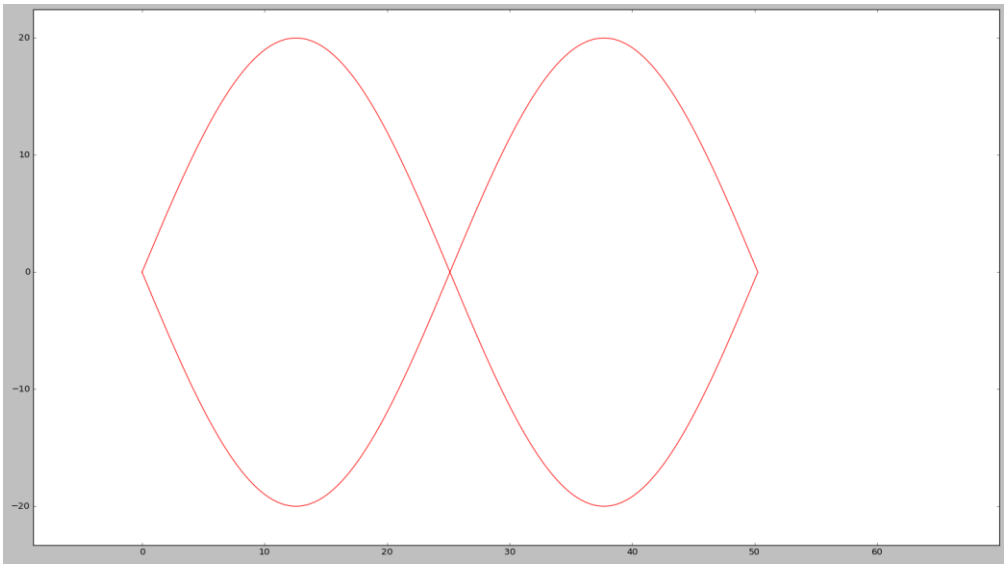

Figura 25 – Ruta preestablecida para el control de dirección

Esos giros bruscos situados al inicio y a mitad de la simulación servirán para probar como de bien se comporta nuestro controlador ante giros brucos.

En esta simulación solo existe el vehículo controlado, que seguirá la ruta trazada. El vehículo partira de una posición ajena a la trayectoria y con una orientación de cero, es decir, el ángulo de dirección de las ruedas del vehículo es cero, tal y como se muestra en la figura 26.

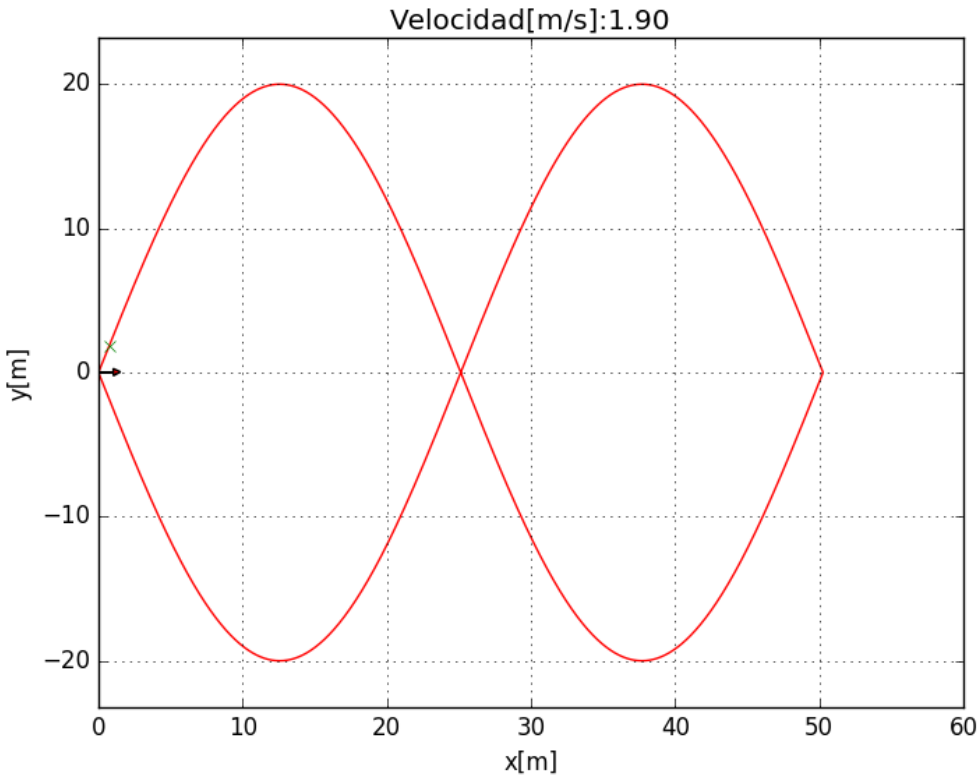

Figura 26 – Posición inicial del vehículo junto con la ruta preestablecida.

La velocidad máxima del vehículo será de 10 m/s, ya que para zonas urbanas un vehículo autónomo no tendría que tener velocidades muy altas. En las siguientes figuras, desde la 27 a la 30 se muestra la evolución de la simulación.

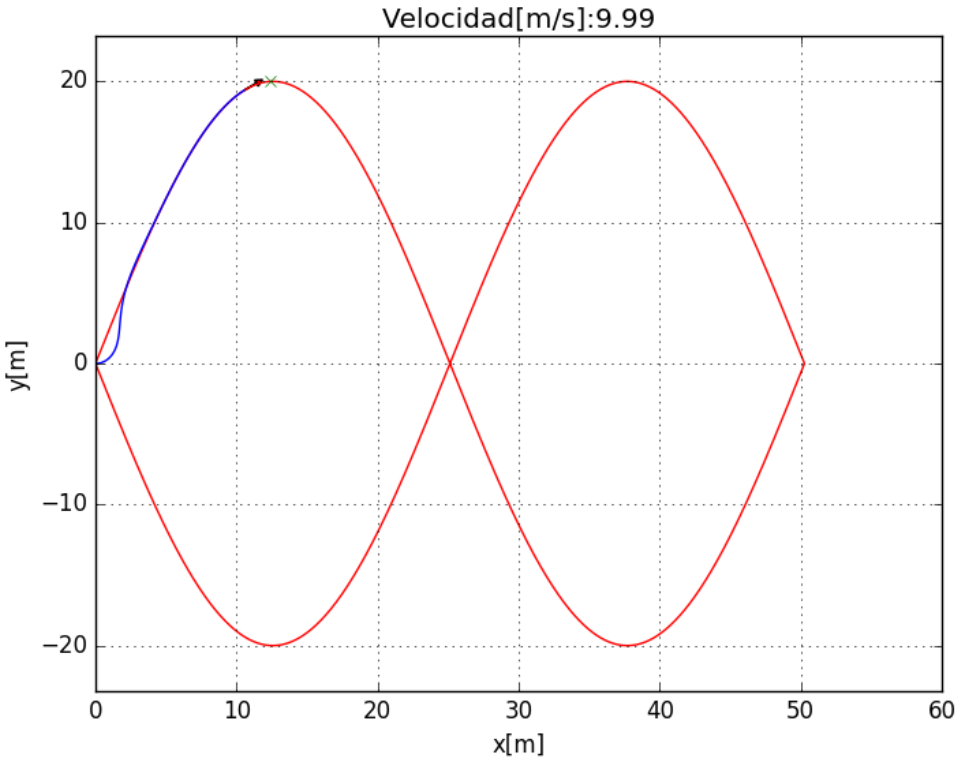

Figura 27 – Momentos iniciales de la simulación del control de dirección

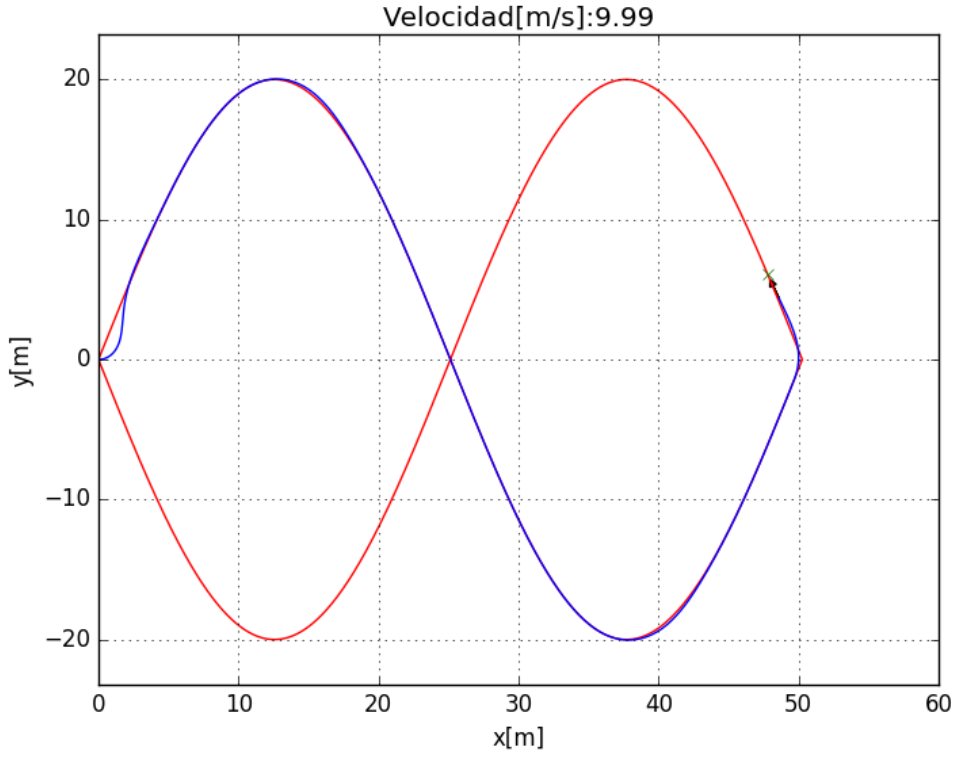

Figura 28 – Momento de un giro brusco en la simulación

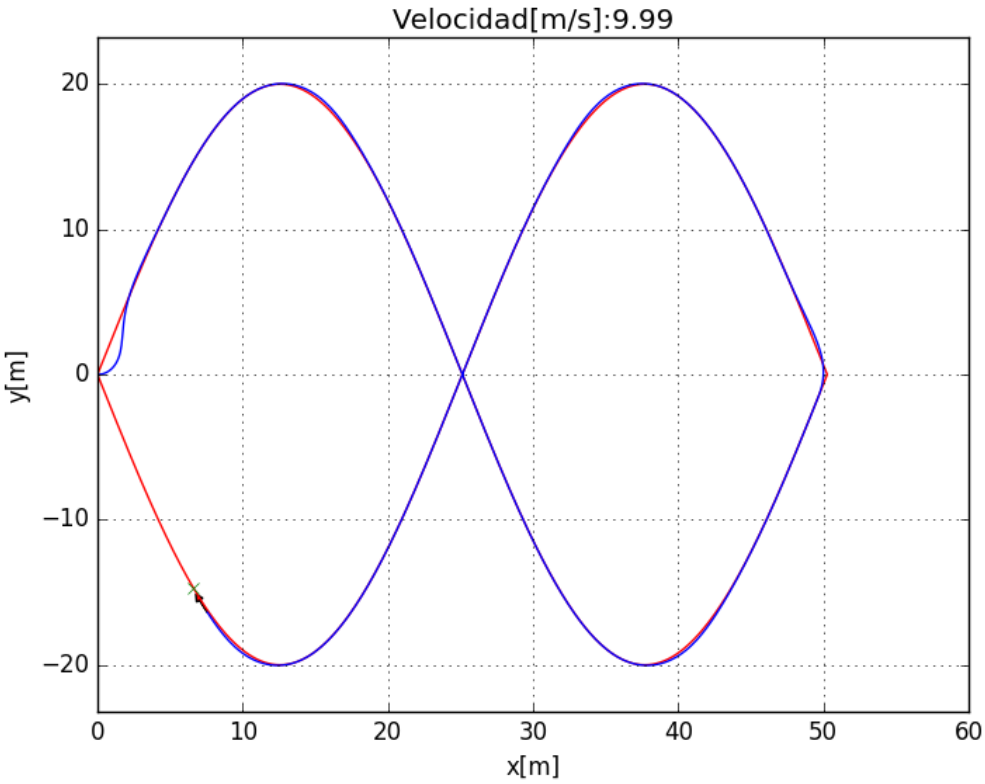

Figura 29 – Momentos finales de la simulación

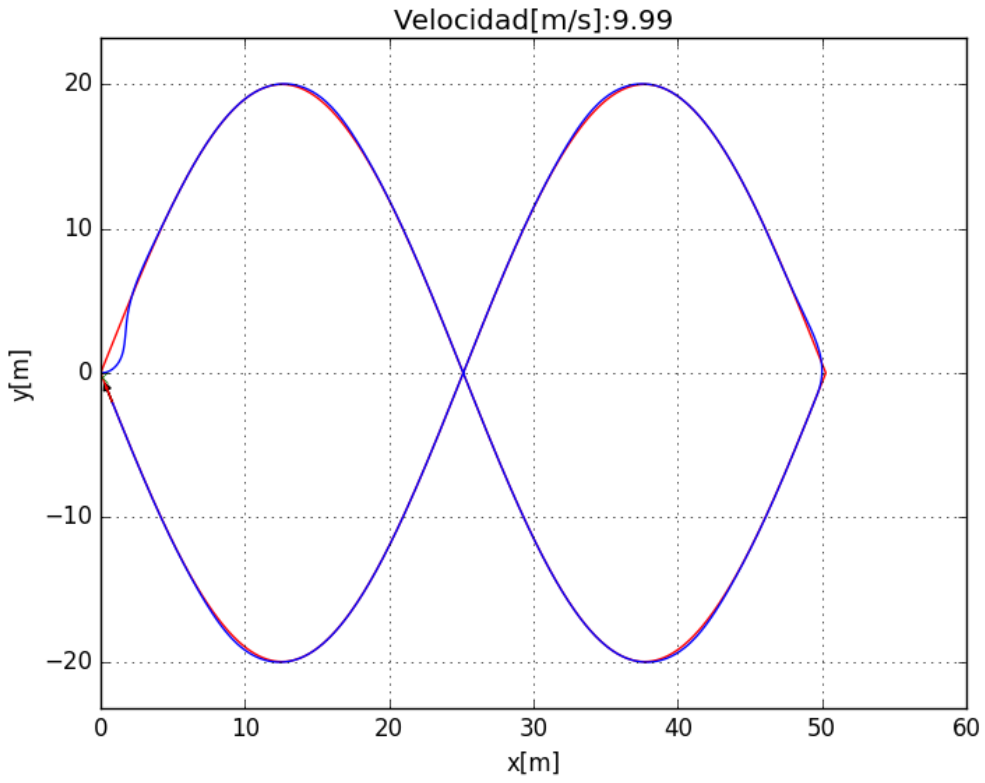

Figura 30 – Simulación del control de dirección finalizada

A partir de la simulaciones se puede decir que el control de dirección sumple con su cometido. Sin embargo, se debe decir que es mejor evitar las curvas bruscas, como las mostradas en la figura 25.

También se puede orservar que en algunas zonas el camino que toma el vehículo se desvia por my poco del camino establecido, sin embargo es algo que se podría permitir para el tipo de controlador que se esta implementando.

7 SIMULACIÓN EN GAZEBO

Una vez testeado los diferentes controladores se pasa a realizar las pruebas en 3D, en Gazebo.

Desafortunadamente como se comento en el apartado 5.1 de modelado 3D, al realizar las modificaciones de las ruedas directrices del vehículo para que puedan girar respecto a su eje "z", se obtienen errores en la simulación. Las ruedas directrices simplemente no girán y cuando lo hacen se quedan en un ángulo dado pese a estar mandadole otra información o se vuelven locas y giran respecto a su eje "z" sin control, figura 31 y 32.

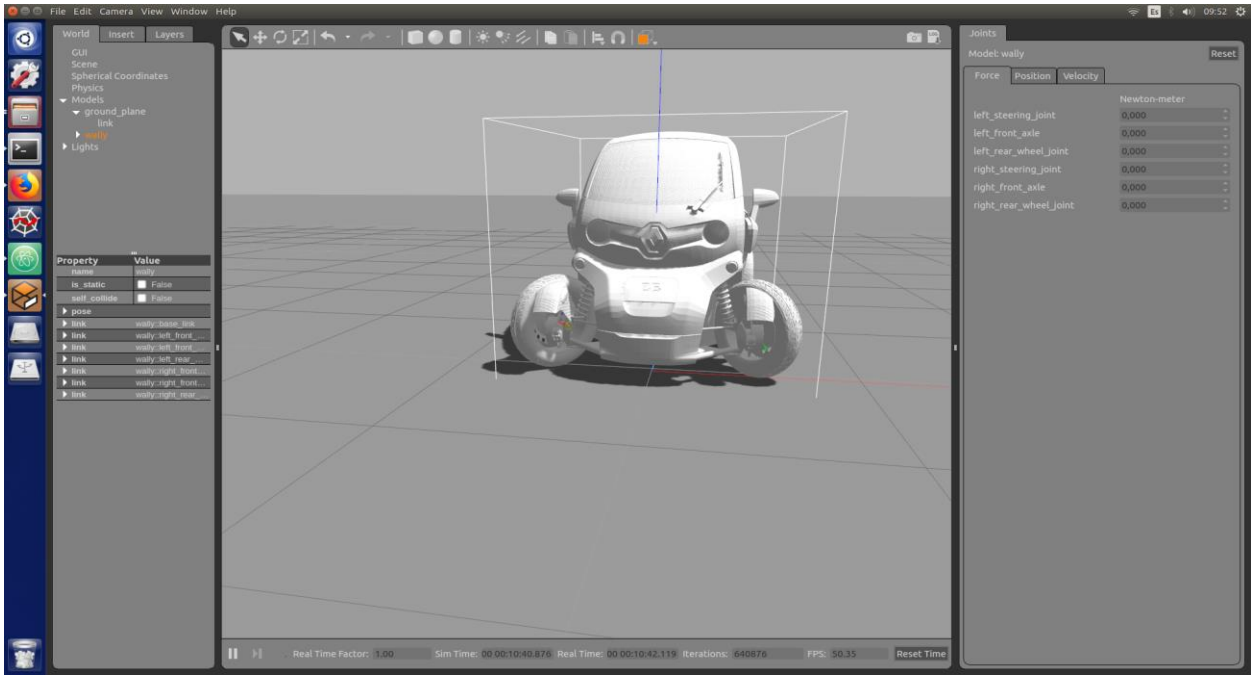

Figura 31 – Uno de los problemas mi modelo del Renault Twizy

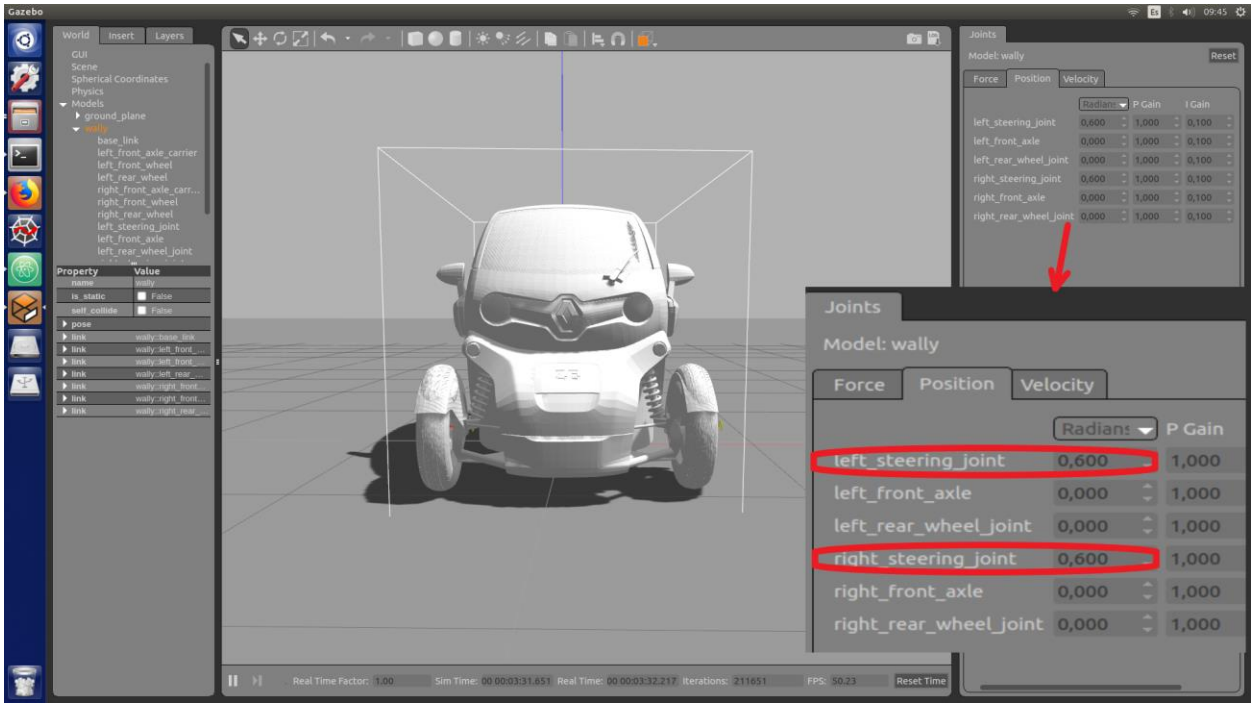

Figura 32 – Otro problema con mi modelo del Renault Twizy

Busque algún fallo en el código URDF del coche o en el archivo .yaml, pero no fui capaz de encontrar algún fallo. Después hice una búsqueda en internet de codigos URDF de coches que pudieran servirme para poder ver como hacen el diseño de las ruedas directrices y comprobarlo con mi código URDF. En escencia, como se puede observar en la figura 33, 34 y 35, sigo el mismo procedimiento que ellos al programar las ruedas directrices, realicé algunas modificaciones donde pensaba que estaba el fallo, sin embargo, el problema persistía, luego seguía sin saber donde estaba el fallo.

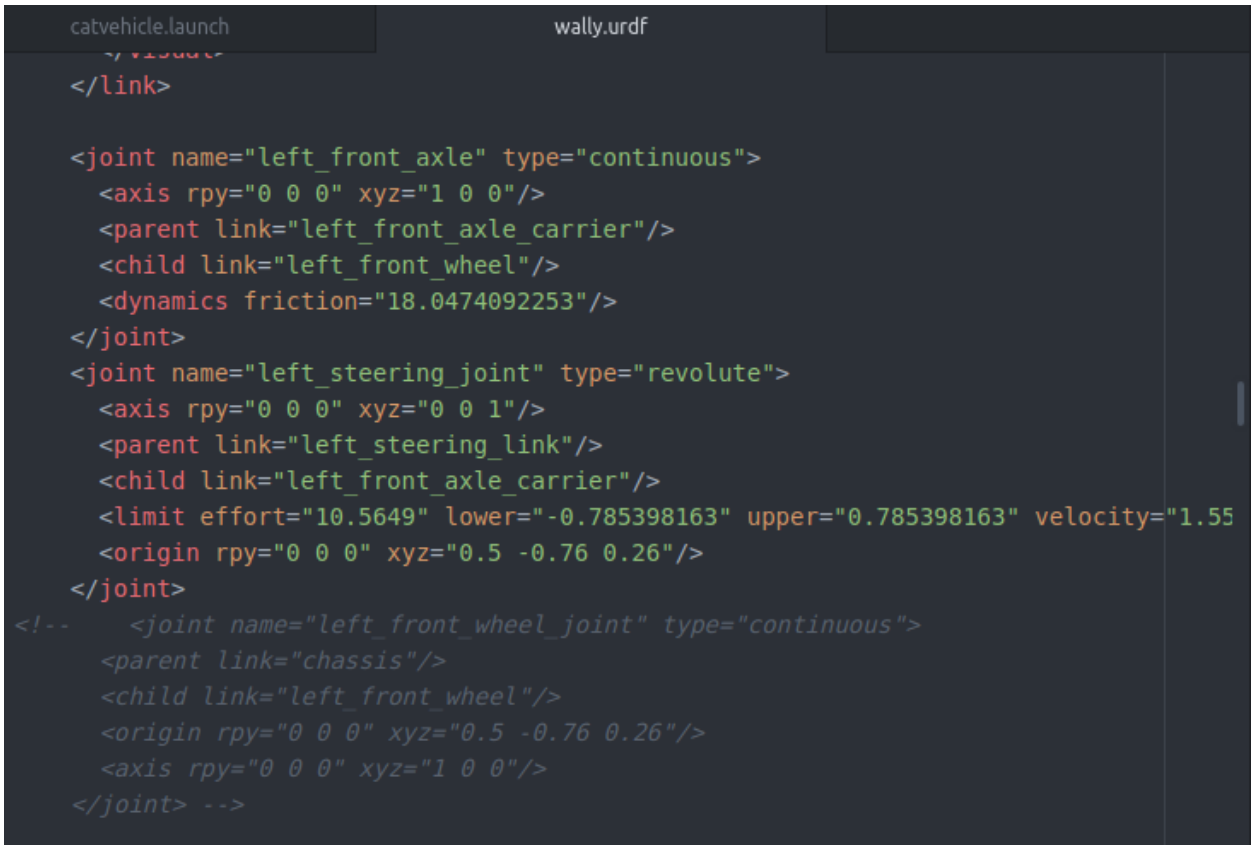

Figura 33 – Mi código XML de la dirección del Renault Twizy

catvehicle.xacro

```
243
244
245
246
247 <joint name="front_right_steering_joint" type="revolute">
248   <parent link="base_link"/>
249   <child link="front_right_steering_link"/>
250   <origin xyz="${tyre_x} ${-car_width/2 - str_length/2} ${tyre_r}" rpy="0 0 0"/>
251         <axis xyz="0 0 1"/>
252   <limit effort="1000.0" lower="${-str_angle}" upper="${str_angle}" velocity="0.5"/>
253 </joint>
254
255 <link name="front_right_steering_link">
256   <collision>
257         <xacro:insert_block name="tyre_origin"/>
258           <geometry>
259            <cylinder length="${str_length}" radius="${str_radius}"/>
260           </geometry>
261   </collision>
262
263   <visual>
264               <xacro:insert_block name="tyre_origin"/>
265                 <geometry>
```

Figura 34 - Código XML de la dirección de CatVehicle

prius.urdf

```
218   <joint name="steering_joint" type="continuous">
219     <origin xyz="0.357734 -0.627868 0.988243" rpy="-1.302101 0 0"/>
220     <parent link="chassis"/>
221     <child link="steering_wheel"/>
222     <axis xyz="0 0 1"/>
223     <limit lower="-7.85" upper="7.85" effort="10000000" velocity="1000000"/>
224   </joint>
225
226   <joint name="front_left_steer_joint" type="revolute">
227     <parent link="chassis"/>
228     <child link="fl_axle"/>
229     <origin xyz="0.767 -1.41 0.3" rpy="0 0 0"/>
230     <axis xyz="0 0 1"/>
231     <limit lower="-0.8727" upper="0.8727" effort="1000.0" velocity="0.5"/>
232   </joint>
233   <joint name="front_right_steer_joint" type="revolute">
234     <parent link="chassis"/>
235     <child link="fr_axle"/>
236     <origin xyz="-0.767 -1.41 0.3" rpy="0 0 0"/>
237     <axis xyz="0 0 1"/>
238     <limit lower="-0.8727" upper="0.8727" effort="1000.0" velocity="0.5"/>
239   </joint>
```

Figura 35 - Código XML de la dirección del Prius (proyecto Car_demo)

También realicé una comparación en Gazebo de las articulaciones de la dirección en 3D de cada uno de los modelos incluido el mio tal y como se muestra en la figura 36, 37 y 38.

El eje grande con el circulo representa el eje de giro respecto a “x”, es decir, para que la rueda gire longitudinalmente, y el eje pequeño con su circulo, con zoom, representa el eje de giro de la dirección de la rueda, es decir respecto a su eje “z”.

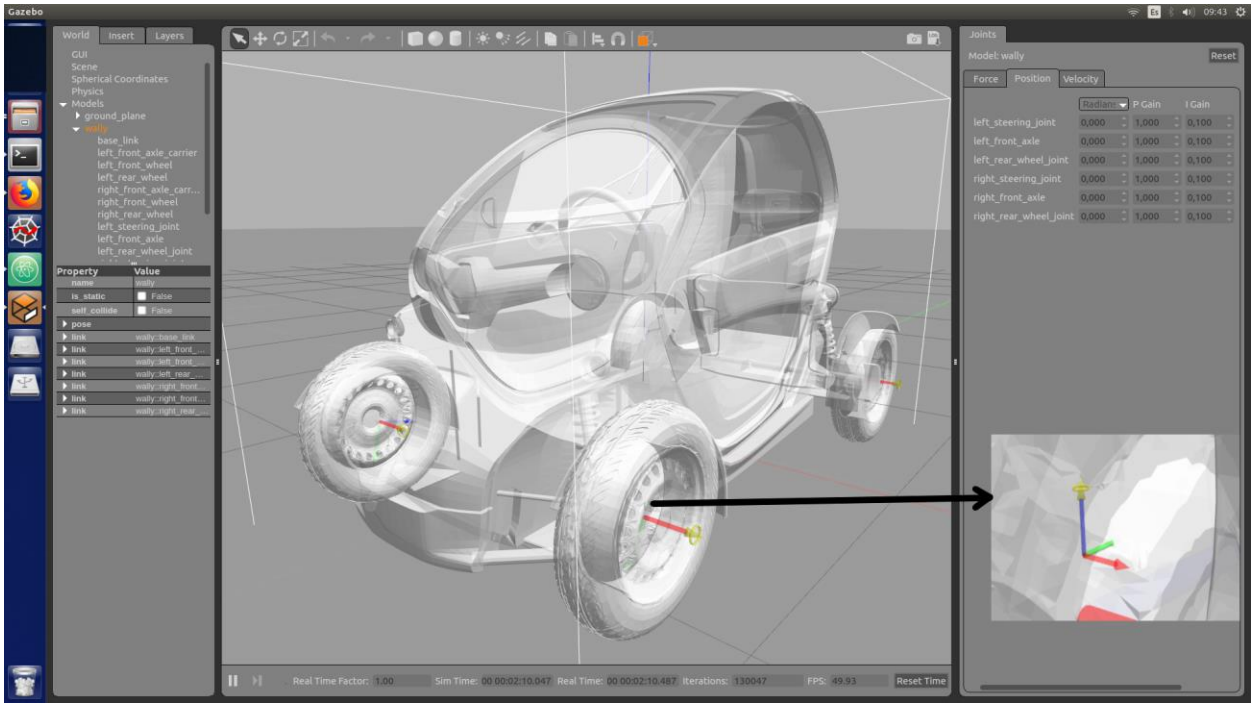

Figura 36 – Ejes de giro de la rueda directriz de mi modelo del Renault Twizy

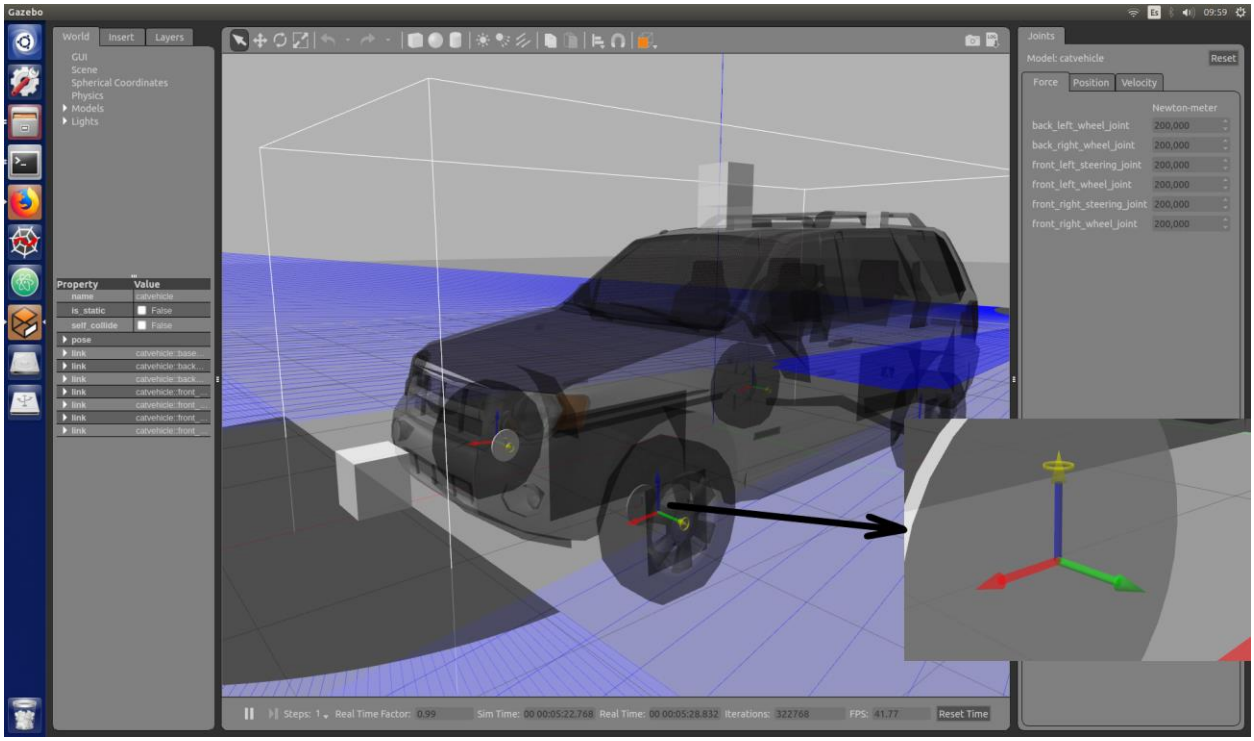

Figura 37 – Ejes de giro de la rueda directriz del modelo CatVehicle

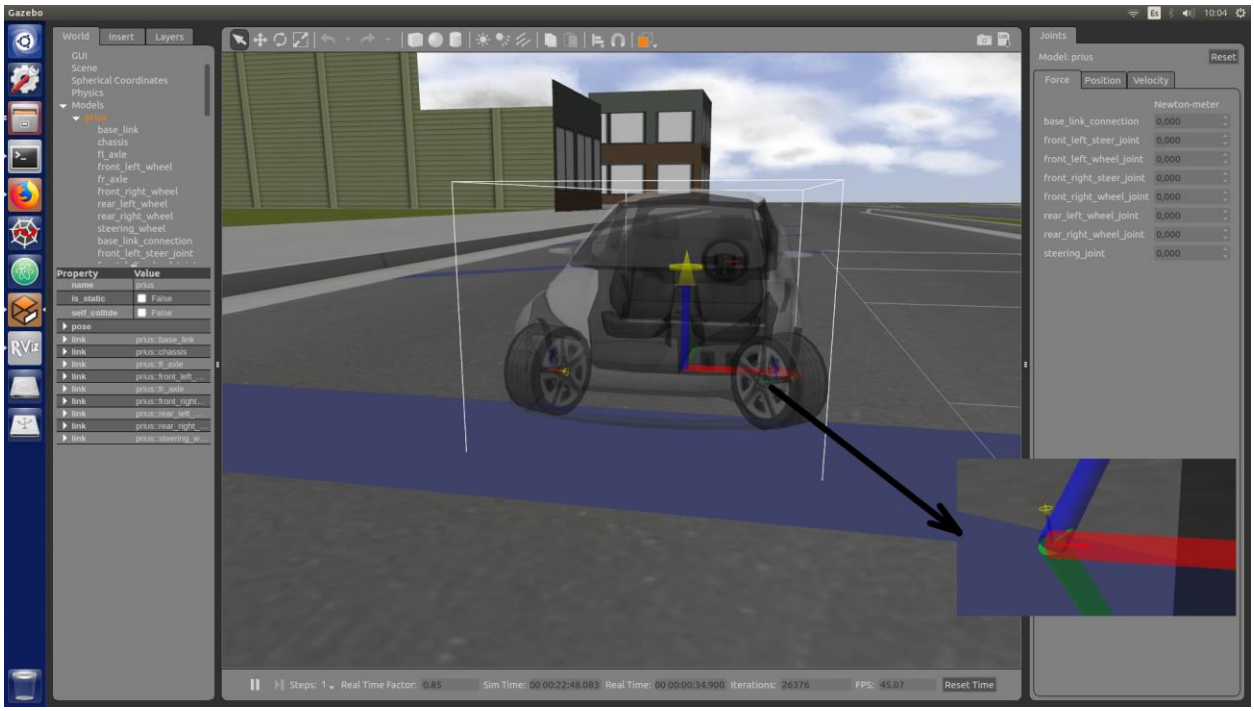

Figura 38 – Ejes de giro de la rueda directriz del modelo car_demo (Prius)

Al ver el resultado en 3D de cada uno de los modelos, incluido el mio, es notable que todos los modelos son muy parecidos, sin embargo, depues de los siguientes análisis que se explicará a continuación, puedo decir que el fallo está fuera de mi alcance de comprensión.

Ya como último recurso me propuse realizar las simulaciones con los vehículos programados por la comunidad, encontré dos vehículos, uno de un proyecto llamado Catvehicle y otro de un proyecto llamado car_demo, los anteriormente vistos.

El primero de ellos, al cargarlo en Gazebo e intentar mover el vehículo, las ruedas directrices directamente no funcionan, es decir, no avanzan longitudinalmente ni giran respecto a su eje “z” como se observa en la figura 39.

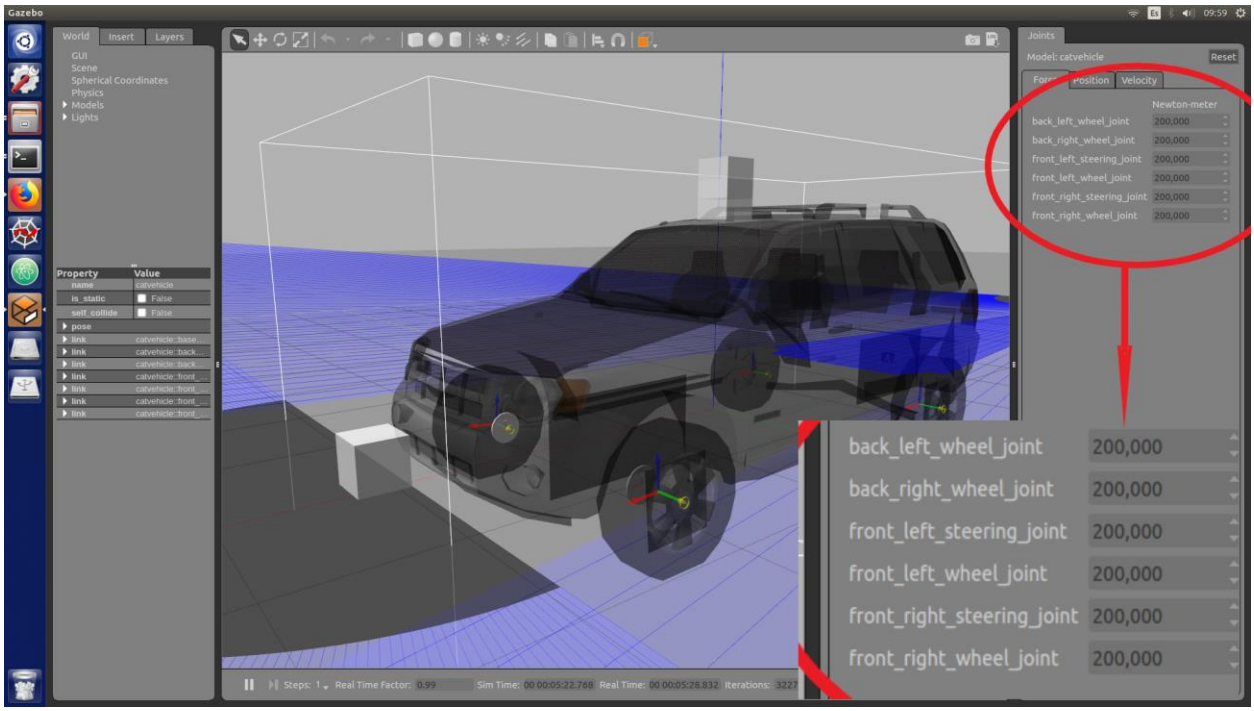

Figura 39 – Simulación de prueba en Gazeebo con el modelo de CatVehicle

Como se observa en la figura 39, se está introduciendo valores a las articulaciones del vehículo, sin embargo, el vehículo no realiza ningún movimiento ni intento de hacerlo.

El segundo de ellos, que es un Prius, al cargarlo en gazebo tras unos segundos de simulación, las ruedas directrices se vuelven completamente locas girando respecto a su eje “z”, figura 40 y 38.

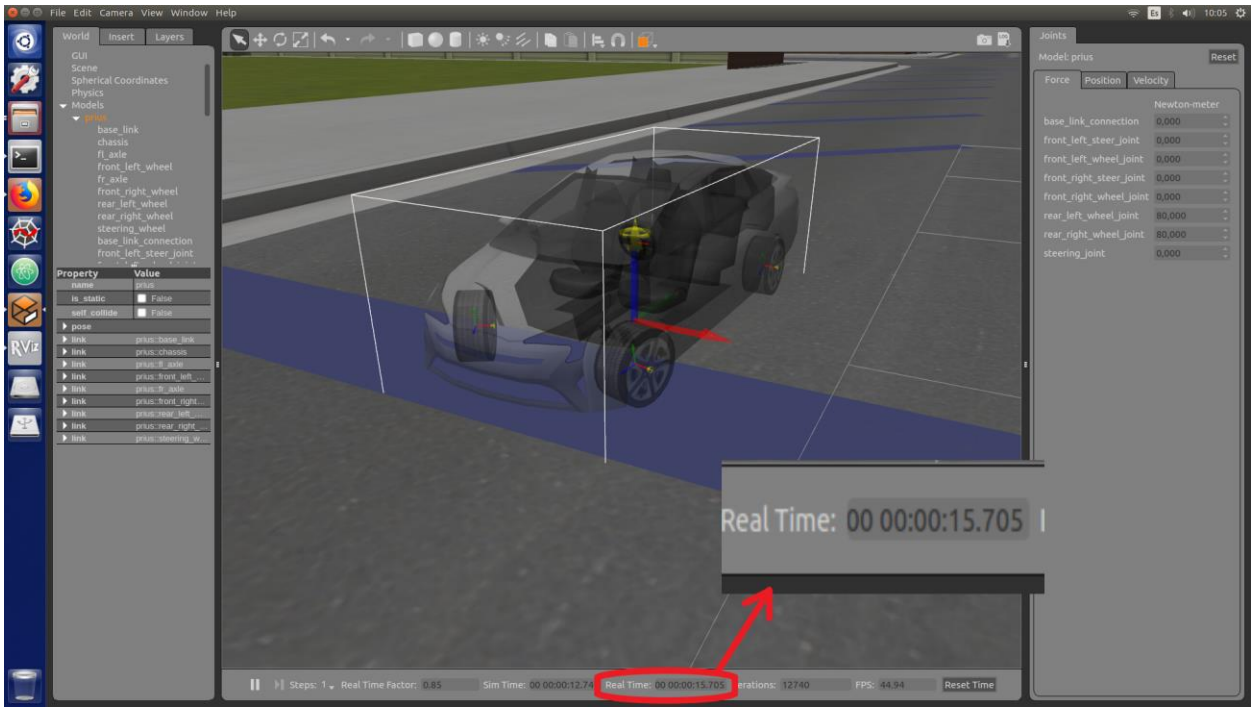

Figura 40 – Simulación de prueba en Gazebo con el modelo de car-demo (Prius)

Como se puede observar en la figura 40, con tal solo 15 segundos de simulación y sin hacer nada, las ruedas se giran solas, y no vuelven a su posición, aunque forcemos un angulo de dirección de 0º.

Tras todas estas pruebas realizadas llegué a la conclusión que el modelado en 3D del Renault Twizy está parcialmente terminado, solo fallando cuando se intenta añadir un giro respecto a su eje "z" para simular la dirección del vehículo, con esto impidiéndome realizar las simulaciones en 3D.

8 PRESUPUESTO DE LAS MODIFICACIONES

En este apartado se tratará un presupuesto realizado de las piezas mecánicas y otros componentes que serían necesarios para convertir al Renault Twizy en un coche de pruebas para conducción autónoma.

Los componentes que se tendrán en cuenta en este presupuesto son los que se han visto en el apartado de sensores y actuadores.

Componente	Nombre	Características
Laser 2D	SICK LASSER TIM551	Rango de operación: 0.05 m a 8 m, Ángulo de apertura: 270º, Frecuencia de escaneo 15 hz
Camara ToF	ToF Sentis3D-M420	Rango de operación: 7 m, FOV: 90º
Lidar 3D	Laser Bear Honeycomb	FOV vertical: 95º, FOV horizontal: 360º, Rango mínimo de cero, 2 millones de horas testeado
Camara CCD	Prosilica GT1290C	A color, 1.2 Megapixel con visión GigE y puerto Gigabit Ethernet,
Modulo RTK GNSS	REACH M+ EMILID RTK GNSS Receiver	Precisión en la posición de 7 mm
IMU	NAV440CA-200-2	Precisión en pitch y roll < 0.4º, Precisión en la posición < 0.3 m
Motor	Maxon EC60fl	Reductor GP52C, el Encoder MILE de 4096 cpt con 2 canales y un controlador electrónico EPOS4.
OVMS	OVMS	

Tabla 3 – Componentes del presupuesto

Componente	Precio por unidad	Precio total
SICK LASSER TIM551	1 713,882 €	3 427,764 €
ToF Sentis3D-M420	1 470,92 €	1 470,92 €
Laser Bear Honeycomb	10 000,00 €	10 000 €
Prosilica GT1290C	1 079,00 €	1 079,00 €
REACH M+ EMILID RTK GNSS Receiver	265 €	265 €
NAV440CA-200-2	7 541,82 €	7 541,82 €
Maxon EC60fl	-	772,64 €
OVMS	160 €	160 €
Presupuesto final		24 717,144 €

Tabla 4 – Presupuesto final con todos los componentes incluidos

Obviamente si queremos hacer un presupuesto bien hecho habría que incluir el salario de los trabajadores y del jefe en cuestión, así como tener en cuenta los años de desarrollo que tendría este hipotético proyecto, entre otras muchas cosas.

9 CONCLUSIONES Y TRABAJOS FUTUROS

8.1. Conclusiones

Después de llevar a cabo las simulaciones se puede llegar a la conclusión de que la mayoría de los objetivos principales de este proyecto se han podido realizar, como el desarrollo del controlador longitudinal y de dirección. Sin embargo, las simulaciones en un entorno virtual no se han podido llevar a cabo debido a un problema con Gazebo, pudiendo solamente realizar parcialmente su modelado en gazebo debido a una falla a la hora de programar la dirección del vehículo.

Con relación al control longitudinal del vehículo, se puede decir que el controlador cumple su función, pudiendo mejorarse el tiempo de respuesta del controlador.

Con relación al control de dirección del vehículo, también cumple su función, pero de las simulaciones se podría rescatar que se tiene que evitar giros bruscos con curvas cerradas.

También se ha propuesto una arquitectura de control con la que tendría que contar un vehículo autónomo, así como las modificaciones mecánicas y a nivel se software que se tendría que realizar para poder llevar a cabo pruebas autónomas con el Renault Twizy.

8.2. Trabajos futuros

Como trabajos futuros se propone realizar diferentes comparativas entre diferentes tipos de control, como el borroso o modelos predictivos.

También se podría trabajar con un modelo mas complejo que el de la bicicleta, usado para el control de dirección.

Resolver el problema de la dirección del modelo en 3D en Gazebo.

Realizar las simulaciones en 3D que no se han podido llevar a cabo en este proyecto, asi como la simulación del control de estabilidad.

Realizar las modificaciones mecanicas propuestas al Renault Twizy y llevar a cabo las pruebas de control en un entorno controlado real, asi como implementar los diferentes sensors propuestos en el presupuesto.

De este proyecto espero que sirva como punto de partida para otros proyectos similares, o continuarlo en un futuro trabajo fin de master.

REFERENCIAS

[1] Rajamani, R.. "Vehicle dynamics and control.", Second edition, Ney York, Springer, 2012.

[2] Snider, Jarrod M.. "Automatic Steering Methods for Autonomous Automobile Path Tracking.", 2009.

[3] Coulter, R. C.. "Implementation of the Pure Pursuit Path Tracking Algorithm.", 1992.

[4] Jafarnejad, S.; Codeca, L.; Bronzi, W.; Frank, R. y Engel, T.. "A Car Hacking Experiment: When Connectivity Meets Vulnerability.", *2015 IEEE Globecom Workshops (GC Wkshps)*, San Diego, CA, 2015, pp. 1-6.

[5] Bussemaker, K. J.. "Sensing requirements for an automated vehicle for highway and rural environments.", 2014.

[6] Borraz, R.; Navarro, P.; Fernández, C. y Alcover, P.. "Cloud Incubator Car: A Reliable Platform for Autonomous Driving.", 2018.

[7] Serban, A. C.; Poll, E. and Visser, J.. "A Standard Driven Software Architecture for Fully Autonomous Vehicles.", *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, Seattle, WA, 2018.

[8] Ulsoy, A., Peng, H., & Çakmakci, M.. "*Automotive Control Systems*.", Cambridge: Cambridge University Press. doi:10.1017/CBO9780511844577

[9] En esta página web se encontraron tutoriales de Gazebo de mucha utilidad a la hora de desarrollar este proyecto.

http://gazebosim.org/tutorials

[10] En esta página web se encontraron tutoriales de ROS de mucha utilidad a la hora de desarrollar este proyecto.

http://wiki.ros.org/ROS/Tutorials