Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Modelado de escenarios para electromovilidad en entornos urbanos

Autor: Juan Carlos García Brenes

Tutor: Carlos Vivas Venegas

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

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Modelado de escenarios para electromovilidad en entornos urbanos

Autor:
Juan Carlos García Brenes

Tutor:
Carlos Vivas Venegas
Profesor Titular

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

Sevilla, 2023

Trabajo Fin de Grado: Modelado de escenarios para electromovilidad en entornos urbanos

Autor: Juan Carlos García Brenes
Tutor: Carlos Vivas Venegas

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar quisiera darle las gracias a mi familia que siempre me ha estado apoyando y me han dado la oportunidad de realizar estos estudios universitarios.

También me gustaría mostrar mi agradecimiento a mi tutor, Carlos Vivas Venegas por ayudarme resolviendo o aclarando algunas dudas que me iban surgiendo durante la realización de este trabajo. Sus sugerencias y comentarios han sido importantes para la realización de este.

Para terminar quería darle las gracias a todos aquellos que, de una u otra manera, han contribuido en este trabajo y han sido parte fundamental de mi formación académica.

Juan Carlos García Brenes
Sevilla, 2023

Resumen

Los vehículos eléctricos representan cada día un mayor porcentaje del parque móvil global. Por ello, existe la necesidad de dotar de infraestructuras y procedimientos de carga adecuados a estos vehículos, lo que presenta un problema aún no resuelto que implica grandes retos desde el punto de vista de la generación eléctrica y de la distribución de estas infraestructuras.

En este proyecto se pretende utilizar herramientas de simulación apropiadas para escenarios de este tipo. Se busca adaptar el simulador de código abierto llamado Eclipse SUMO (Simulation of Urban Mobility) e integrarlo con Matlab o Python como herramienta de monitorización y control de gestión de tráfico.

La idea principal es crear un conjunto de escenarios en SUMO donde circulen continuamente un gran número de vehículos eléctricos. Estos circularán respetando los semáforos, límites de velocidad y prioridades en las intersecciones, con el fin de que la simulación sea lo más realista posible. En este escenario de simulación también se incluirán numerosas estaciones de carga donde los vehículos se pararán a cargar su batería. Dicha simulación se controlará a través de un código por medio de Matlab, empleando las funciones de la librería *traci4matlab* que nos permite la comunicación entre Matlab y el simulador SUMO. El control consiste en monitorizar el estado de la batería con el objeto de tomar las acciones apropiadas para garantizar un estado mínimo de carga en los vehículos en circulación. En este control también se incluirán diferentes procedimientos en el caso de que los vehículos no se paren de forma correcta en la estación con el fin de evitar atascos.

El objetivo de estas simulaciones es obtener estadísticas de uso de la red de carga de modo que permitan desarrollar procedimientos para la toma de decisiones en relación a la mejor ubicación geográfica y capacidad de las mismas.

Abstract

The escalation in the global vehicle fleet's adoption of electric vehicles has prompted a pressing need to establish comprehensive charging infrastructure and protocols for these vehicles. This issue presents an unresolved challenge, fraught with significant considerations concerning electricity generation and infrastructure distribution.

The objective of this project is to leverage simulation tools tailored to such scenarios. Specifically, the aim is to adapt the open-source simulator known as Eclipse SUMO (Simulation of Urban Mobility) and seamlessly integrate it with either Matlab or Python as a traffic management, monitoring, and control system.

The central concept revolves around crafting a diverse array of scenarios within SUMO, featuring a continuous flow of electric vehicles. These vehicles will adhere to traffic regulations, including traffic lights, speed limits, and intersection priorities, to emulate a realistic simulation environment. Additionally, these scenarios will incorporate an array of charging stations where vehicles will pause to recharge their batteries.

The execution of this simulation will be governed by Matlab code, employing the capabilities of the 'traci4matlab'library to facilitate communication between Matlab and the SUMO simulator. The control mechanism will primarily involve monitoring the battery state to implement appropriate measures, ensuring a minimum state of charge for vehicles in circulation. This control framework will also encompass various procedures to address situations in which vehicles fail to stop adequately at the charging stations, thus mitigating the risk of traffic congestion.

The primary aim of these simulations is to acquire statistical insights into the utilization of the charging infrastructure. This data serves as a foundation for crafting informed decision-making protocols concerning the optimal geographical placement and capacity allocation of the charging networks.

Índice Abreviado

Índice

1 Motivación del proyecto

Debido al aumento del precio del carburante de estos últimos años y a las cada vez más frecuentes restricciones de acceso a los vehículos de combustión interna a ciertas zonas de circulación en las ciudades, los usuarios se han planteado cada vez más la idea de comprar un vehículo eléctrico o de cero emisiones.

A este aumento desde el punto de vista económico se le une que la Comisión Europea quiere conseguir que el 90 % de los vehículos que circulen por carretera en el año 2050 sean de cero emisiones con el objetivo de reducir drásticamente la contaminación derivada de los vehículos de combustión interna, para conseguir esto se espera que todos los automóviles nuevos que se destinen a la venta en Europa en 2035 sean vehículos eléctricos, híbridos o enchufables. [12]

Para ayudar e incentivar la compra de vehículos eléctricos se están llevando a cabo una serie de iniciativas por parte de nuestro país. Estas consisten en destinar dinero para la reestructuración de la industria del automóvil. Dentro de estas medidas se encuentra el programa de ayudas directas a las comunidades autónomas llamado Plan Moves III [15], que sirve para incentivar la compra de vehículos eléctricos y la instalación de infraestructuras necesarias para su recarga.

Recientemente se aprobó en España el Real decreto 266/2021, aprobado en Abril de 2021 [19] que pretende que en 2023 se hayan instalado 100.000 puntos de recarga de uso público con el fin de dinamizar la movilidad sostenible. Además en este decreto también se contempla la creación de un mapa oficial en el que se puedan encontrar todos los puntos de recarga existentes.

Para incentivar la compra de estos vehículos no contaminantes es esencial la mejora de las infraestructuras y estudiar cuidadosa y responsablemente la distribución de estas en las ciudades, ya que uno de los principales problemas de estos vehículos junto a su poca autonomía, es que la relación entre el tiempo de carga y la distancia que puede recorrer con una sola carga es bastante superior a la que se daría para los vehículos de combustión interna. A continuación, se muestran una tabla comparativa 1.3 donde se puede ver la diferencia entre un vehículo de combustión y uno eléctrico. Los tiempos de carga y distancias recorrida suele variar por ello se ha tomado tiempos y distancias medias.

Para rellenar la tabla comparativa se van a tomar las características de un modelo de vehículo eléctrico puro como es el Volvo EX30 [2]. En la tabla 1.1, se muestran los datos que se van a usar para obtener el tiempo de carga y la distancia media.

Tabla 1.1 Características de un vehículo eléctrico puro.

Vehículo Volvo EX30 eléctrico puro	
Capacidad batería	51.0 kWh
Consumo energía eléctrica (combinado)	16.7 kWh/100 km
Consumo energía eléctrica (ciudad)	12.1 kWh/100 km
Estación carga CC	175 kW
Estación carga CA	11 kW

Una vez conocido las especificaciones del vehículo eléctrico se va a calcular el tiempo de carga y la distancia media para el caso más y menos desfavorable.

- **Cálculo caso más desfavorable**: para este cálculo se va a tomar la estación de carga más lenta de las dos que se han considerado y el mayor consumo de energía eléctrica que es el combinado. Para realizar este cálculo la eficiencia de la estación de carga se va a tomar de un 95 %.

$$t_{carga} = \frac{CapBateria}{P_{estacion} * Eficiencia} = \frac{51000}{11000 * 0.95} = 4.88 \quad \text{h} = 17570 \quad \text{segundos} \tag{1.1}$$

$$dist_{media} = \frac{CapBateria}{Consumo_{energia}(combinado)} * 100 = \frac{51000}{16700} * 100 = 305 \quad \text{Km} \tag{1.2}$$

- **Cálculo caso menos desfavorable**: para este cálculo se va a tomar la estación de carga más rápida de las dos que se han considerado y el menor consumo de energía eléctrica que es el que se da por ciudad. Para realizar este cálculo la eficiencia de la estación de carga se va a tomar de un 95 %.

$$t_{carga} = \frac{CapBateria}{P_{estacion} * Eficiencia} = \frac{51000}{175000 * 0.95} = 0.307 \quad \text{h} = 1104 \quad \text{segundos} \tag{1.3}$$

$$dist_{media} = \frac{CapBateria}{Consumo_{energia}(ciudad)} * 100 = \frac{51000}{12100} * 100 = 422 \quad \text{Km} \tag{1.4}$$

Al igual que se ha hecho para obtener unos valores reales del vehículo eléctrico, también se va a tomar las especificaciones reales de un vehículo de combustión interna, que es el Volvo XC60 diésel [4], a continuación se muestra una tabla con las especificaciones necesarias para realizar los cálculos.

Tabla 1.2 Características de un vehículo de combustión interna.

Vehículo Volvo XC60 diésel	
Capacidad depósito de combustible	71 l
Consumo combustible	6.1 l/100 km

Una vez presentadas las especificaciones de un vehículo de combustión interna, ya se pueden realizar los cálculos a comparar.

$$dist_{media} = \frac{CapDep}{Consumo_{combustible}} * 100 = \frac{71}{6.1} * 100 = 1164 \quad \text{Km} \tag{1.5}$$

El tiempo repostaje de un vehículo de combustión interna se va a tomar entorno a 5 minutos.

Tabla 1.3 Comparativa entre un vehículo eléctrico y uno de combustión.

Vehículo		Tiempo repostaje o carga (s)	Distancia media (km)	Relación (s/km)
Eléctrico puro	-Más desfavorable	17570	305	57.6
	-Menos desfavorable	1104	422	2.62
Combustión		300	1164	0.26

Sería necesario disponer a lo largo de todo el núcleo urbano y de las carreteras de puntos de recarga bien distribuidos sin que supongan un grave problema para la correcta circulación de los vehículos. Por ello será muy importante el estudio del comportamiento de los vehículos eléctricos en el tráfico urbano, ya que en estos núcleos van a existir un gran flujo de vehículos y grandes zonas con estaciones de carga, lo que podría provocar grandes problemas de movilidad en las ciudades.[12]

1.1 Objetivos del proyecto

El propósito de este proyecto es analizar mediante simulación el impacto de la infraestructura de carga en el el flujo de tráfico de un área geográfica determinada. Para ello se emplearán herramientas de micro simulación de tráfico de código abierto, como SUMO [18] , que se describirá más adelante, con el fin de ver el comportamiento de los vehículos y controlarlos en función de sus niveles de batería.

Para conseguir esto, primero se va a estudiar la herramienta SUMO y sus prestaciones. Este estudio sobre SUMO se realiza para ver el realismo que se puede conseguir con los escenarios de SUMO y con el modelo del vehículo eléctrico.

Este trabajo se centra en la simulación de vehículos eléctricos por lo que se va a estudiar como se pueden incluir estos en los diferentes escenarios y como se van a incluir las diferentes infraestructuras necesarias como son las estaciones de carga. Además, de incluir las estaciones de cargas se van a incluir en la configuración del escenario todo lo necesario para que la simulación sea lo más realista posible, incluyendo la interacción con otros vehículos, semáforos, límites de velocidad y otros elementos del tráfico real.

Tras estudiar todo lo anterior se va a realizar un control de la simulación por medio de MATLAB, que será nuestra herramienta de monitorización para controlar el tráfico del escenario. Para adaptar el simulador SUMO a MATLAB se va a utilizar la toolbox *traci4matlab*. *Traci4matlab* carece de una documentación amplia para entender su funcionamiento, pero la propia web de MATLAB que nos permite descargar esta toolbox [5] contiene un apartado de funciones donde se puede ver el código de cada función y un pequeño resumen de su funcionamiento. Si este resumen no es suficiente para entender el funcionamiento de la función que se va a utilizar, se puede buscar la función equivalente en *traci* y ver la documentación que nos aporta la web de SUMO [13] que suele ser más amplia y de gran ayuda. *Traci4matlab* permitió dar los primeros pasos de integración de SUMO con un entorno de programación versátil (MATLAB). No obstante, debido a varios errores no documentados en el código de *traci4matlab*, se decidió pasar a usar *traci*, con Python, que está

bastante mejor mantenida y documentada.

El control que se va a realizar con Python sobre la simulación se basa en monitorizar en todo instante la batería de los vehículos del escenario y las estaciones de carga. Tras realizar un control efectivo de la red de vehículos, se van a realizar diferentes configuraciones de la simulación para obtener un campo de resultados amplio para poder analizar.

1.2 Fases del proyecto

Para realizar el proyecto, se comenzó con la búsqueda de varios simuladores de tráfico con el fin de elegir el más adecuado, tras esta búsqueda se eligió el simulador SUMO ya que dispone de una amplia documentación en su web para entender su funcionamiento y con él se puede simular vehículos eléctricos y sus estaciones de carga. Se puede ver el desarrollo de esta fase en el apartado 2.3.1.

En la segunda fase se instaló SUMO y las librerías necesarias y se añadió a MATLAB la toolbox *traci4matlab* que se utilizó para comunicar MATLAB con SUMO. Se podrá ver en el apartado 3.2.

La tercera y cuarta fase consistió en familiarizarse con el entorno de SUMO y en ver la estructura de los diferentes ficheros que se utilizan en la simulación. En concreto en la tercera fase se centró en la obtención de escenarios de simulación conocidos para utilizarlos en la simulación. Esta fase se puede ver en el apartado 3.3.2. Mientras que en la cuarta fase, se estudió cómo se pueden añadir las propiedades de los vehículos eléctricos y las estaciones de carga a la simulación, hecho que se puede seguir en el apartado 3.3.3.

En la quinta fase se realizan las primeras pruebas de simulación con MATLAB y se comienzan a probar las diferentes funciones que se utilizarán para las simulaciones finales. En esta fase surgieron varios problemas ya que existen algunos errores en la toolbox descargada. El apartado 4.1.5 refleja los estudios realizados en esta fase.

En la sexta fase debido a los errores no documentados del software de *traci4matlab* empleado en la fase anterior, se decidió migrar a la versión python, mejor mantenida y documentada, decisión que se mostró acertada con posterioridad. Fase que se refleja en el apartado 4.1.6.

En la séptima fase se realizó la simulación final con Python y se explicó el escenario y en que consistía dicha simulación. Esta fase se representa en el apartado 4.2.

En la octava y última fase que se asemeja al apartado 5 se ejecutaron varias veces la simulación final y se obtuvieron y analizaron los resultados.

1.3 Material utilizado

En el desarrollo de este Trabajo Fin de Grado se va a utilizar las siguientes herramientas o software:

- Simulador Eclipse SUMO (1.14.1). *Enlace instalación*
- Herramienta Netedit (1.14.1) proporcionada por SUMO.
- MATLAB (2021b). *Enlace instalación*
- Herramienta OSM Web Wizard proporcionada por SUMO.
- Herramienta Netconvert proporcionada por SUMO.

- Toolbox *traci4matlab*. *Enlace instalación*
- PYTHON (3.10). *Enlace instalación*
- Spyder (anaconda3). *Enlace instalación*
- Módulo *Traci* proporcionada al instalar SUMO.

2 Estado del Arte

Actualmente, en España en 2023 hay contabilizados alrededor de 250.000 vehículos eléctricos que se encuentran en circulación, cantidad significativamente insuficiente si se quiere alcanzar el objetivo de la agenda 2030 para España que prevé 5 millones de vehículos eléctricos, aproximadamente un 16 % del parque móvil. [17]

Cabe destacar que España se encuentra en los puestos de cola en la implantación de la movilidad eléctrica debido a los altos precios de estos vehículos y a la falta de una red de carga pública operativa. Debido a todo esto el salto a los vehículos 100 % eléctricos se ve muy complicado de asumir ya que se sabe que España cuenta con uno de los parques automovilísticos más envejecidos de la Unión Europea con una media de 13,5 años por vehículo, por lo que esto sería una magnífica oportunidad para aprovechar los fondos de ayuda europeos y el proyecto PERTE [10] con el fin de incentivar la compra de vehículos eléctricos.

Llegar a la cifra de 5 millones de vehículos enchufables al cierre de 2030 se antoja un objetivo muy ambicioso ya que las infraestructuras de carga de nuestro país no están preparadas para este flujo de vehículos eléctricos. Solo se disponen de aproximadamente 15.000 puntos de carga en todo el país, cifra baja en relación al objetivo de 100.000 puntos de carga que se marcó el Gobierno para 2023. Además cabe destacar que la distribución de estos puntos debería mejorar ya que existen grandes diferencias entre las diferentes comunidades autónomas del país.

La distribución de estos puntos de carga es muy importante debido a que uno de los principales problemas de los vehículos eléctricos es la autonomía de las baterías que suele estar en torno a 200 y 350 kilómetros, condicionando la duración de los trayectos y por eso de la necesidad de una buena distribución de puntos de carga por todo el país. Para ayudar a la ubicación de los diferentes puntos de carga el Gobierno decretó la publicación del primer mapa de recarga pública del país.[16]

2.1 Vehículos eléctricos

El vehículo eléctrico se define como un automóvil que es propulsado por uno o por varios motores eléctricos, estos pueden alimentarse de una fuente externa que les suministre energía eléctrica o pueden ser autónomos al tener instaladas baterías, paneles solares o generadores eléctricos que tranforman un combustible en electricidad.

A partir de la década de los 90, los vehículos eléctricos han tenido un rápido desarrollo en la industria del automóvil con el principal objetivo de impulsar su utilización para poder disminuir los

problemas relacionados con las emisiones de gases y la dependencia de los combustibles fósiles.

Además de los beneficios que se han nombrado anteriormente, usar vehículos eléctricos tienen otras ventajas para el usuario, ya que estos vehículos requieren de menos mantenimientos que los de combustión interna y son menos ruidosos lo que ayuda a reducir considerablemente la contaminación acústica que sufren las ciudades debido al tránsito de vehículos.

A pesar de todas estas ventajas la movilidad eléctrica se enfrenta a grandes retos como la inversión inicial que requiere la adquisición de un vehículo eléctrico, la menor autonomía que presentan en relación a los vehículos convencionales y al tiempo requerido para recargar las baterías, a todo esto se le une que en la actualidad existe una desconocimiento por parte del consumidor sobre las ganancias que se lograrían con el uso de un vehículo eléctrico, lo que conlleva que los consumidores se lo piensen a la hora de cambiar el coche convencional por el eléctrico.

A la hora de hablar de vehículos eléctricos se pueden distinguir en la industria automovilística varios tipos de coches electrificados: [1]

- **Vehículo eléctrico híbrido**

  Estos utilizan la combinación de dos fuentes de energía para mover el vehículo, ya que se combina el motor de combustión interna con el motor eléctrico. Con esta tecnología se consigue reducir el consumo de combustible en un 40 % gracias a la utilización de una batería y del motor eléctrico que se utilizan con el fin de mejorar el rendimiento del combustible en relación a un vehículo convencional.

  Existen dos tipos de vehículos híbridos:

  – **Convencional (HEV):** cuyo motor principal es de combustión y se ayuda por medio de un motor eléctrico cuya batería se auto-recarga mediante un sistema de frenado regenerativo que convierte la energía cinética en energía eléctrica.

  – **Enchufables (PHEV):** el motor eléctrico es mayor y tiene capacidad para mover por sí solo el vehículo durante una determinada distancia. La batería de este no se auto-recarga por lo que necesitará de una recarga externa.

  Su contribución a la sostenibilidad, al ahorro de consumo, su autonomía y el precio elevado de su competidor directo el vehículo eléctrico puro hacen de estos vehículos una buena alternativa para el consumidor [6].

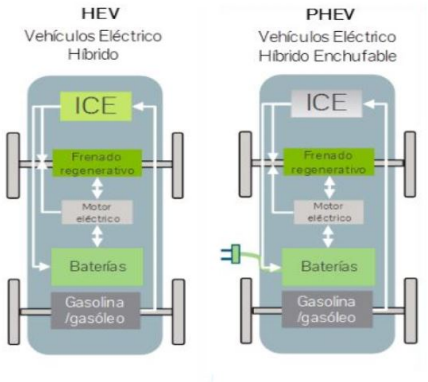

Figura 2.1 Esquemas de los dos tipos HEV y PHEV [22].

- **Coche eléctrico de pila de combustible (FCEV)**

  Se trata de un tipo de vehículo eléctrico que no usa una batería recargable por medio de la red eléctrica para almacenar la energía necesaria para alimentar el motor, si no que usa una celda de combustible que produce energía a medida que el vehículo la necesita, esta se crea a partir del hidrógeno y del oxígeno que recibe mediante un proceso electroquímico.

  El hidrógeno utilizado se almacena en una serie de tanques y se canalizan hasta la celda de combustible donde se mezcla con el oxígeno del aire produciendo electricidad, al realizar este proceso se obtiene agua que se expulsa por el tubo de escape en estado gaseoso.

  Estos vehículos también se pueden considerar híbridos, ya que se recupera la energía de frenado y se almacena en una pequeña batería con el fin de ser utilizada para reducir la demanda máxima de la celda de combustible [16][9].

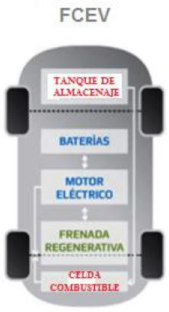

Figura 2.2 Esquema del vehículo eléctrico de pilsa de combustible (FCEV) [16].

- **Vehículo eléctrico puro (EV) o vehículo eléctrico de batería (BEV)**

  Son vehículos propulsados totalmente por un motor eléctrico alimentado por baterías que se recargan gracias a la red eléctrica. El sistema básico de un vehículo eléctrico puro está compuesto por un motor eléctrico, una batería y por un controlador que se encarga de regular la energía que se transmite al motor eléctrico proveniente de la batería.

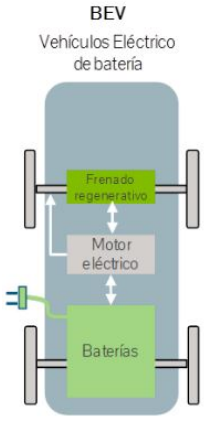

Figura 2.3 Esquema del vehículo eléctrico de batería (BEV) [22].

El mayor problema de estos vehículos se debe a su autonomía ya que está limitada por la capacidad y el peso de las baterías.

Debido a este problema es esencial la distribución correcta y abundante de puntos de carga por todo el país, para que la autonomía de estos vehículos deje de ser un problema [16].

El problema anterior provoca que los vehículos de combustión interna, híbridos y de celda de combustible, sean las fuentes de energía más utilizadas, debido a la comodidad para el usurario, disponibilidad y autonomía.

Hoy en día las baterías que componen los vehículos eléctricos basadas en la tecnología de iones de litio, no pueden competir con la densidad de energía del combustible fósil ni con las infraestructuras de carga, sin embargo a pesar de las dificultades que encuentra el vehículo eléctrico para abrirse paso en el mercado, se prevé que sea unos de los candidatos más fuerte de la industria automovilística y en los sectores de investigación.

2.2 Estaciones de cargas

El tiempo de recarga de una batería depende de la potencia eléctrica que el punto de carga es capaz de suministrar y del nivel de carga de la batería. Actualmente los vehículos eléctricos tienen baterías muy pesadas y costosas, además estas necesitan de largos periodos de carga ya que se intenta mantener la vida útil y la eficiencia. La idea de realizar largos periodos de carga cuando sea posible es para evitar la degradación de la batería del vehículo, ya que al realizar cargas rápidas se favorece la degradación de esta. Uno de los principales objetivos del campo de investigación de los vehículos eléctricos consiste en lograr reducir los tiempos de carga de los vehículos para que la autonomía de estos no sea un problema grave.

El sistema de carga que más desarrollo e implantación tiene en la actualidad se trata del sistema alámbrico, este requiere de una conexión eléctrica de suministro para cargar su batería mientras el vehículo se encuentra inmóvil. Los vehículos eléctricos actuales dependen de una única carga alámbrica previa a su uso, lo que provoca una serie de limitaciones en su velocidad de carga y su recorrido máximo. Una posible solución sería la de diseñar supercargadores y que estén distribuidos por todo el país con el fin de que sean accesibles a todos los usuarios. La instalación de supercargadores demandan mucha electricidad de la red y requieren reforzar la red eléctrica si se implantan de forma masiva. [23]

2.2.1 Tipos y modos de recarga

A la hora de hablar de las estaciones de carga hay que saber, que se le llama tipo de recarga a la forma de transmitir la energía y que el modo de recarga es el tipo de adaptación de red que se necesita.

En la actualidad se pueden diferenciar en función de la potencia suministrada tres tipos de recarga:

- **Baja potencia (<3,7 kW):** esta recarga se destina a viviendas y edificios que realicen la recarga en horas nocturnas.
- **Potencia normal (>3,7 kW y <22 kW):** es la que más se utiliza como recarga de apoyo o grupal. Se encuentran en lugares de acceso público como centros comerciales, hoteles.
- **Alta potencia (>22 kW):** está pensada para ser utilizada en casos de emergencia para obtener una recarga ultrarápida.

Sabiendo la potencia que suministra cada tipo de recarga y la capacidad de la batería del vehículo eléctrico, se puede calcular la duración estimada de la recarga. El la siguiente tabla se muestra el modo de carga que se puede utilizar en cada tipo de recarga y la duración de la recarga para un

vehículo eléctrico de 20 kWh de capacidad en función del tipo de recarga que se utilice.

Tabla 2.1 Características de los tipos de cargas que hay en la actualidad.

Tipo de carga	Potencia	Duración	Modo
Baja potencia (<3,7 kW)	2,3 kW (10 A 230 V)/3,7 kW (16 A 230 V)	9 h/ 5,5 h	1,2,3
Potencia Normal (>3,7 kW <22 kW)	11 kW (10 A 400 V)/22 kW (32 A 400 V)	2 h/ 1 h	3
Alta potencia (> 22 kW)	43,6 kW (63 A 400 V)	25 minutos	3,4

Según la UNE-EN 611851-1 existen 4 modos de carga alámbrica normalizados, con diferentes características y funciones de recarga que se definen en función de la capacidad de la estación de carga. Cada tipo de recarga e infraestructura puede ser compatible con varios modos de recarga, como se observa en la tabla representada anteriormente. El modo de carga que se puede utilizar se determinará en función del vehículo eléctrico y de los elementos de conexión disponibles. Los modos de carga se definen de la siguiente forma:

- **Modo 1:** La recarga se realiza usando una base de toma de corriente de uso general y no existe comunicación entre la estación y el VE.
- **Modo 2:** La recarga se realiza usando una base de toma de corriente de uso general con protecciones y control incluido en la caja de control del cable de recarga, tiene poca comunicación entre la estación y el VE.
- **Modo 3:** Se realiza por medio de una toma de corriente diseñada para recargar vehículos eléctricos. En este modo se usa siempre una estación de recarga donde se encuentras las funciones de protección y de control. Con este modo existe una alta comunicación entre la estación y el VE.
- **Modo 4:** Estación de recarga para uso exclusivo del vehículo, con un suministro con corriente continua de elevada potencia. Este modo de recarga usa siempre una estación de recarga y tiene un alto nivel de comunicación entre la estación y el VE.

Los VE suelen ser compatibles con un modo en concreto, por ello si un VE es compatible con el modo 3 se recomienda por cuestiones de seguridad utilizar estaciones de recarga que sean del modo 3 y que este modo sea la prioridad frente a los modos 1 y 2 [20].

2.2.2 Estaciones de carga en España

Al igual que existen redes de gasolineras fiables y de garantías para los vehículos de combustión, la movilidad eléctrica exige una red de electrolineras. En relación a la red de electrolineras, España debe pisar el acelerador en su implantación ya que su nivel de infraestructuras para los coches eléctricos se encuentra entre una de las peores de la Unión Europea, por lo que el gobierno estaría obligado a mejorar las infraestructuras y en poner a disposición del público general un registro oficial de los puntos de cargas disponibles en el país. Desde este año se dispone del primer mapa nacional donde se muestran todos los puntos de carga disponibles en el territorio español.

En la siguiente imagen se puede ver la distribución de los diferentes puntos de carga alrededor de la extensión de el territorio español. En esa imagen se pueden distinguir diferentes puntos que indican dependiendo del color a que estación de carga se corresponde, siendo de color rojo las estaciones de carga de mayor potencia y de color verde las estaciones de carga de menor potencia.

Figura 2.4 Puntos de carga para coches eléctricos repartidos por el territorio nacional [3].

Según los estudios y datos de Anfac se estima que a día de hoy hay 13.411 puntos de carga de acceso público en todo el territorio español, cifra que se aleja mucho de los 100.000 puntos de carga que quiere el Gobierno para finales del año 2023. Al bajo ritmo de instalación de puntos de carga se le une que la mayoría de los puntos de cargas que ya se han instalado se corresponden con puntos de carga de niveles de potencia menores de 22 kW lo que se asocia con la carga lenta. Solo el 12 % de los puntos de carga del país se corresponden a puntos de carga de potencias superiores a 22 kW.

Con esto se puede saber que de la estimación inicial de 13.411 de puntos de carga, 11.805 son estaciones de carga de menos de 22 kW que requieren tiempos de 3 horas o más para cargar un vehículo eléctrico por completo, mientras que solo existen 92 puntos de carga ultra rápida con potencia de 250 kW o más que permiten cargar un vehículo por completo en tan solo 10 o 15 minutos. En el estudio realizado por Anfac también se puede ver la distribución de estaciones de carga entre las diferentes comunidades autónomas del país.

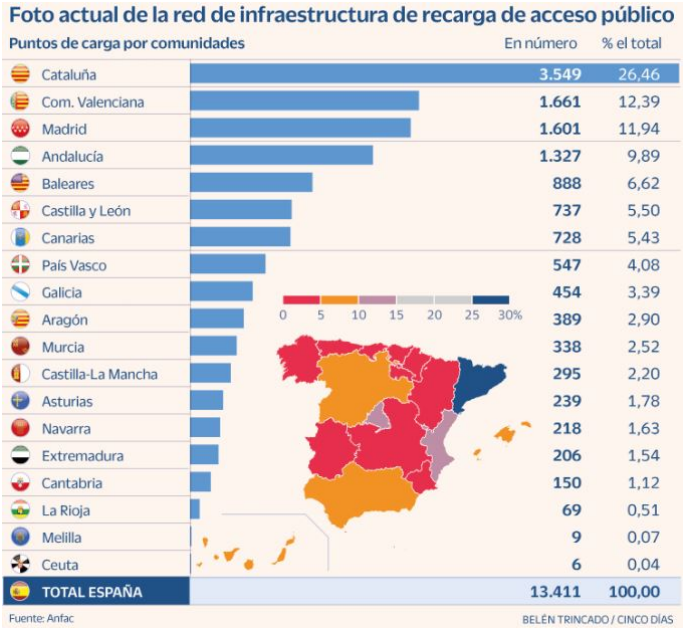

Figura 2.5 Distribución entre las diferentes comunidades de los puntos de carga [14].

Estos datos aportados no son del todo reales ya que no existe un conteo oficial del número de

puntos de carga que se han instalado en el país, por ello si se busca en otras fuentes se pueden ver datos distintos en relación al número de puntos de carga que hay. Si se consulta en la web Electromaps donde los propios usuarios van actualizando los lugares donde se encuentran las estaciones de carga, se pueden encontrar 25.497 puntos de carga de los cuales el 95 % son de uso público mientras que el 5 % restante son de uso privado.

De esta cifra aportada por esta web se indica que la mayoría de los puntos de carga (59 %) que hay en España son de Type 2 (carga normal) cuya potencia oscila entre (3,7 y 43 kW), mientras que los de mayor potencia suponen tan solo el 8,2 % del total [14].

2.3 Herramientas de simulación

Se pueden distinguir 3 tipos de simuladores: simulador de red, de movilidad y simulador Ad Hoc.

- Los **simuladores de red** se usan para probar el rendimiento de los protocolos para la movilidad de los nodos, ya que la topología de red cambia continuamente por lo que los nodos están la mayor parte del tiempo en movimiento.
- Los **simuladores de movilidad o de tránsito** son programas informáticos que permiten el modelado de distintos escenarios de tránsito vehicular. Las simulaciones permiten determinar el movimiento de los vehículos y su comportamiento sobre un mapa o escenario.
- Los **simuladores de red vehicular Ad Hoc** también llamados simuladores VANET permiten una combinación de los dos tipos de simuladores anteriores.

Para realizar este trabajo de fin de grado se van a utilizar simuladores de movilidad, aunque se va a hablar sobre algunos simuladores que trabajan con los dos tipos (movilidad y red) como es el simulador Veins. Todos los simuladores que se van a presentar van a ser de código abierto debido a que este trabajo se basa en ellos.

Al trabajar con simuladores, la introducción de datos se puede realizar mediante consola o por medio de una interfaz gráfica(GUI), normalmente se utilizan ficheros XML para la introducción de estos datos. Existen simuladores que aportan fichero XML como datos de salida y a su vez una representación gráfica en tiempo real donde se pueden ver los resultados de modo más claro y fácil. Para asegurar el funcionamiento del simulador y las operaciones que han sido introducidas por el usuario existe un controlador que actualiza la información de los vehículos que se muestran en el mapa dando como resultado el movimiento de los vehículos, este controlador se va a encargar de mostrar los resultados y de actualizar la representación tanto a nivel macroscópico como a nivel microscópico.

Los simuladores a nivel microscópicos se basan en el comportamiento de cada vehículo o conductor de forma individual. Los factores en los que se fijan estos simuladores son:

- **Comportamientos de los cruces:** Todos los vehículos no se comportan igual cuando se encuentran en un cruce, ya que se habrán definido prioridades de paso entre los diferentes vehículos. Se podría dar una situación en la que en un cruce un conductor deje pasar a otro porque el vehículo que tiene prioridad de paso está muy cerca del cruce, en esta misma situación el conductor que debería dejar paso no se esperaría a que el vehículo de mayor prioridad pasase ya que se encuentra a una distancia suficiente para pasar el cruce antes de que llegue.

- **Patrones de comportamiento del conductor:** Se puede variar la forma de conducir de cada conductor dependiendo de varios factores en relación al comportamiento del conductor. Por ejemplo, se podría configurar un conductor como agresivo haciendo que este conduzca con cambios más bruscos de velocidad.
- **Cambio de carril:** Los vehículos pueden adelantar a otros cambiándose de carril lo que puede provocar cambios en la forma de moverse los vehículos.

Los simuladores a nivel macroscópico se centran en la contrucción del escenario de simulación y de definir las rutas a seguir por los vehículos.

- **Mapas:** La topología de las vías de un escenario pueden afectar considerablemente a la simulación ya que la longitud, el ancho y el número de intersecciones de una calle pueden afectar a la velocidad de los vehículos lo que suele llevar a variaciones en los resultados de la simulación. Se pueden distinguir dos tipos de mapas o escenarios:
  - **Reales:** Estos mapas se pueden obtener de bases de datos como OSM Web Wizard o OpenStreetMap. En estás aplicaciones se pueden encontrar escenarios reales de todo el mundo.
  - **Personalizados:** El usuario se encarga de crear estos tipos de mapas con la topología que se desee, se suelen usar para recrear o probar situación muy concretas. Existen aplicaciones que permiten crear escenarios como NetEdit.
- **Posición inicial y final:** Normalmente estas posiciones se suelen definir en el código de entrada de los simuladores, con ellas se indican el punto de partida del vehículo y el punto de destino de éste. A la hora de definir estas posiciones para un vehículo es necesario indicar el tiempo en el que se inicia el trayecto del vehículo desde la posición inicial.
- **Generación de la ruta:** Una vez que se han definido las posiciones de inicio y de fin se asigna una ruta a seguir al vehículo, por medio de esta ruta este puede llegar de la posición inicial a la final. Normalmente esta ruta se genera de forma aleatoria.
- **Velocidad y aceleración:** La velocidad y aceleración de los vehículos se suele acomodar automáticamente de manera gradual a la situación del tráfico. Normalmente se define una velocidad máxima dependiendo del tipo de vehículo y dependiendo de la vía.

Una vez explicado que es un simulador y algunas de sus características, se van a presentar los diferentes simuladores que se han encontrado [7].

2.3.1 Diferentes simuladores

Existe una gran variedad de herramientas de simulación de movilidad.

Antes de elegir un simulador para realizar este trabajo de fin de grado se ha realizado un búsqueda de los diferentes simuladores y se ha creado una tabla para compararlos:

Tabla 2.2 Comparación de diferentes simuladores.

Nombre	Tipo	Estructura y programación	Depende	Software
SUMO	Movilidad	XML,Python,Matlab(toolbox)	—	Abierto
Straw	Movilidad	Java	GUI	Abierto
VanetMobiSim	Movilidad	Java	—	Abierto
Veins	Red y movilidad	C++,Java,Python,Matlab	OMNet++ y SUMO	Abierto

STRAW

Es una herramienta de generación de modelos de movilidad que está programa en Java y se basa en SWANS que es un simulador de redes gratuito. Este simulador se centra en ambos niveles (macroscópico y microscópico), ya que extrae los mapas de la base de datos TIGER (STRAW) y es capaz de simular semáforos y señales de tráfico y de gestionar los cruces de forma bastante realista. Este simulador no permite visualizar las simulaciones con claridad ya que no tiene incorporado una GUI.

A Pesar del gran realismo que nos aporta STRAW en la gestión de cruces no es una herramienta muy útil para la simulación de grandes escenarios ya que presenta grandes problemas a la hora de simular vías de varios carriles ya que los vehículos de este simulador no son capaces de cambiar de carril [7].

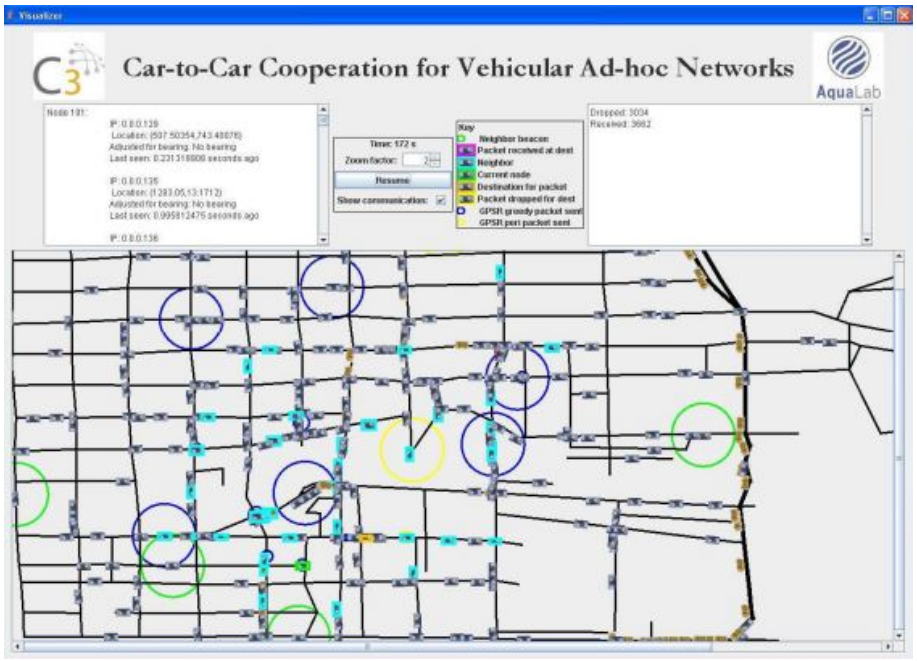

Figura 2.6 Interfaz STRAW [7].

VanetMobiSim

VanetMobiSim es una extensión para el entorno de simulación de movilidad CanuMobiSim. La extensión se centra en la movilidad vehicular en representar nuevos modelos de movimiento más ralistas tanto a nivel macroscópico como microscópico. VanetMobiSim permite importar mapas de base de datos como TIGER, generarlos aleatoriamente y también mapas definidos por el usuario. Además permite utilizar vías de varios carriles con flujo de vehículos en ambas direcciones, restricciones de velocidad en función de la vía y señales de tráfico en cruces e intersecciones. Esto hace que en la simulación se tenga en cuenta la topología, estructura y características de las vías.

Desde el punto de vista microscópico permite que los vehículos realicen adelantamientos y regulen su velocidad y aceleración en función de los vehículos que se acerquen [7].

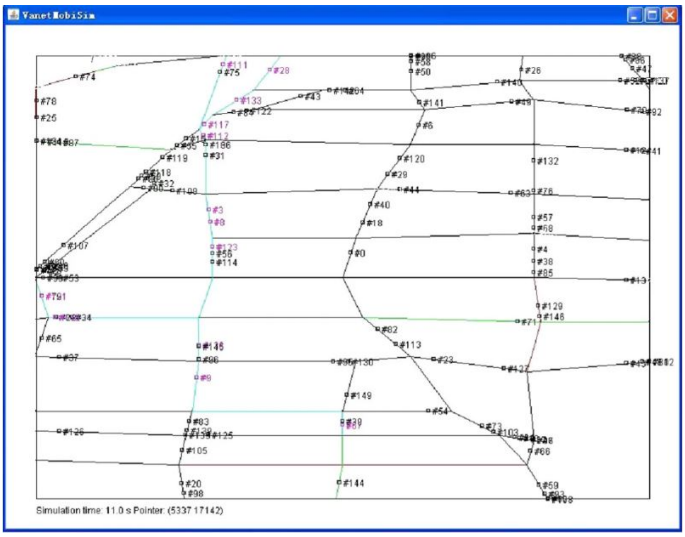

Figura 2.7 Interfaz de VanetMobSim [8].

Veins

Figura 2.8 Interfaz de Veins [21].

Veins es un software de código abierto que permite realizar simulaciones de redes vehiculares. Se basa en dos simuladores OMNeT++ y en SUMO. Las simulaciones de tráfico por carretera las realiza SUMO, mientras que las simulaciones de red la realiza OMNeT++. Veins es muy flexible, esta flexibilidad se debe a que Veins se apoya en el simulador SUMO que es el que permite que se pueda trabajar en otros entornos como Python y Matlab.

Este simulador permite a nivel macroscópico importar escenarios completos de cualquier parte del mundo con OpenStreetMap, incluido todos los edificios, señales de tráfico de las vías, restricciones de velocidad y giro dependiendo de la vía y sus características. También trabaja a nivel microscópico, ya que permite adelantamientos entre los diferentes vehículo y todo los que se permite en el simulador SUMO ya que se basa en él, este se explicará a continuación.

Veins es una buena opción como simulador ya que tiene grandes posibilidades de simulación y

el comportamiento de los vehículos simulados es muy realista, sin embargo Veins se descarta ya que se basa en dos simuladores OMNet++ y en SUMO por lo que aprender a usar este simulador es mucho más complejo que aprender a usar directamente SUMO que es otra de las opciones de simulación que se plantea [21].

SUMO

El simulador SUMO (Simulation of Urban Mobility) es un software libre y desarrollado en C++ en 2001 por el Instituto de investigación del transporte con el objetivo de dar apoyo a la comunidad de investigadores en el área de tráfico. Para hacer de SUMO una aplicación útil se ha puesto mucho cuidado en que sea lo más rápida y portable posible.

SUMO es un simulador microscópico que permite usar de forma rápida una GUI (interfaz gráfica) o bien la línea de comando. Además permite modelar escenarios con muchos detalles como vías con varios carriles, intersecciones con semáforos y vías con límites de velocidad dependiendo de sus características, también permite generar rutas de los vehículos de forma aleatoria. Permite importar y simular escenarios similares a los reales por medio de herramientas como OpenStreetMap o OSM Web Wizard, además permite que el usuario pueda crear sus propios escenarios por medio de la herramienta Netedit que está incluida en el paquete de SUMO [7].

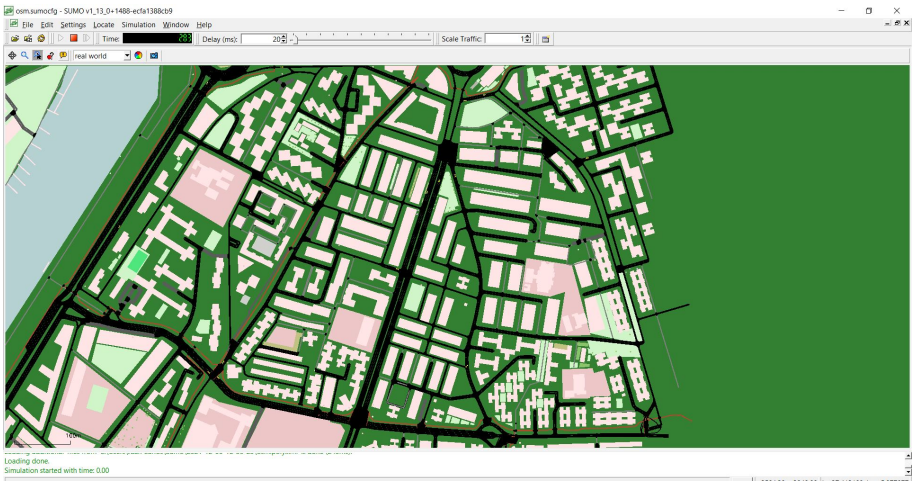

Figura 2.9 Interfaz de SUMO.

Para realizar mi trabajo de fin de grado se ha elegido utilizar el simulador de SUMO ya que permite realizar simulaciones de cualquier zona del mundo y son especialmente flexibles y precisas. Lo que hace a SUMO una buena opción es su gran velocidad de simulación y su gran interoperabilidad con otras aplicaciones en tiempo de ejecución, nos permite trabajar y controlar las simulaciones con Python gracias a la extensión de SUMO llamada *Traci*, esta extensión de Python nos permite trabajar con Matlab gracias a la extensión de Matlab llamada *traci4matlab*.

3 Simulador SUMO

3.1 Introducción

Como ya se explicó anteriormente SUMO (Simulation of Urban MObility) es un simulador de tráfico, microscópico y de código abierto diseñado para generar y simular grandes redes. Permite la simulación de cualquier tipo de vehículo, de peatones e incluye al realizar su instalación un gran conjunto de herramientas que se encargan de la creación de los escenarios de simulación. Además, SUMO es capaz de proporcionar en las simulaciones parámetros específicos de cada vehículo.

Los diferentes ficheros que se necesitan para realizar una simulación se encuentran en formato XML, estos se pueden modificar con herramientas externas a SUMO como Sublime Text. SUMO no es solo un simulador de tráfico ya que también tiene incluido un paquete de herramientas que son necesarias para realizar una simulación de tráfico microscópico [11]. A continuación, se muestran las principales aplicaciones de este paquete con una pequeña descripción de cada una de ellas:

- **Sumo:** Es el simulador que se encarga de realizar la simulación una vez definidos todos los parámetros del resto de las herramientas.
- **OSM Web Wizard:** Es una aplicación que se incluye al descargar el paquete de SUMO y que permite obtener escenarios reales de cualquier zona del mundo.
- **Netconvert:** Es una aplicación de línea de comando que es capaz de convertir el formato de escenarios generados por otras herramientas en el formato que usa SUMO para sus simulaciones. Como se van a coger los escenarios de la aplicación OSM Web Wizard cuyo formato es .osm.xml, Netconvert se va a encargar de convertir ese formato en el formato de SUMO .net.xml.
- **Netedit:** Esta aplicación se incluye dentro del paquete de SUMO y permite crear escenarios desde cero y modificar aquellos escenarios que se han importado de OSM Web Wizard.
- **Sumo-gui:** Se trata de la interfaz gráfica de SUMO que permite visualizar el escenario creado y la simulación en cada tiempo de simulación.
- **Netgenerate:** Se encargan de generar redes de carreteras de forma automática que pueden ser usadas por herramientas de SUMO. Se pueden generar 3 tipos de redes: redes con forma de tela de araña, redes con forma de rejilla y redes aleatorias.
- **Duarouter:** Es una aplicación que permite generar rutas de vehículos definiendo las características del vehículos y la posición de origen y destino.
- **Od2trips:** Esta aplicación permite importar matrices O-D para dividirla posteriormente en viajes individuales.

- **Jtrrouter:** Es otro generador de rutas de SUMO. En este caso de calculan las rutas utilizando porcentajes de giro en los cruces.
- **Dfrouter:** Permite definir y realizar cálculos de las rutas que van a seguir el conjunto de vehículos de un flujo. Esto consiste en definir un grupo de vehículos con una misma posición de origen y destino que seguirán la misma ruta de forma similar, con igual velocidad y aceleración.
- **Marouter:** Permite asignar rutas macroscópicas a partir de una red generada anteriormente y una matriz o/d de posiciones de origen y destino.
- **Polyconvert:** Permite asignar rutas. Se utiliza para importar puntos de interés y polígonos de diferentes formatos y los traduce para que se pueda visualizar en la interfaz de SUMO.
- **Activitygen:** Lee la definición de un población en una red dada y genera los posibles movimientos de esta.
- **EmissionsMap:** Esta herramienta genera matrices con los diferentes valores de emisiones dependiendo del rango de velocidad, aceleración y pendiente, teniendo en cuenta la clase de emisión del vehículo.
- **EmissionsDrivingCycle:** Esta herramienta permite calcular las emisiones para un solo vehículo dada su línea de tiempo de velocidades y aceleraciones (ciclo de conducción).

Junto a todas estas herramientas existen más herramientas adicionales que se pueden ver en la web de SUMO [18].De las herramientas descritas anteriormente se van a utilizar para este proyecto las 6 primeras herramientas que se han descrito.

En los siguientes apartados se explicará cómo se puede realizar la instalación completa de SUMO y sus paquetes y la estructura de los ficheros que necesita SUMO para realizar una simulación completa.

3.2 Instalación de SUMO

La instalación completa del paquete de SUMO se puede realizar desde la web de SUMO [18] concretamente en el apartado llamado "Instalando SUMO" de esta página. Una vez en esta página se puede ver la explicación de como instalar el paquete de SUMO para diferentes sistemas operativos, como se ha utilizado para hacer el trabajo de fin de grado el sistema operativo Window se va a explicar cómo se podría realizar la instalación completa para este sistema.

En la web existen 4 opciones diferentes de instalación dependiendo de la plataforma donde se quiera instalar (32 bits o 64 bits), también dependerá de si se quiere instalar de forma local o de forma portátil. Si se desea descargar de forma local porque se tienen los derechos de administrador de la máquina donde se ejecutará, se va a descargar el instalador preferiblemente el de 64 bits. Una vez que se descarga el instalador solo se tendrán que seguir los pasos que se indican en él.

Una vez instalado el paquete completo de SUMO se podrán encontrar dentro de la carpeta de SUMO todas las herramientas que se indicaron anteriormente y varios ejemplos de simulaciones realizadas con SUMO. Además dentro de la carpeta de SUMO se pueden encontrar todas las funciones necesarias para comunicar Matlab con SUMO, lo que hacen verdaderamente estas funciones es comunicar *traci* con Matlab ya que *traci* es la interfaz de control que proporciona sumo y que permite al usuario acceder y recuperar valores de los objetos simulados y manipular su comportamiento en línea.

Para terminar con la instalación de SUMO hay que añadir en las variables de entorno de mi equipo dentro del *patch* las dos rutas del fichero de SUMO que se muestran en la siguiente imagen.

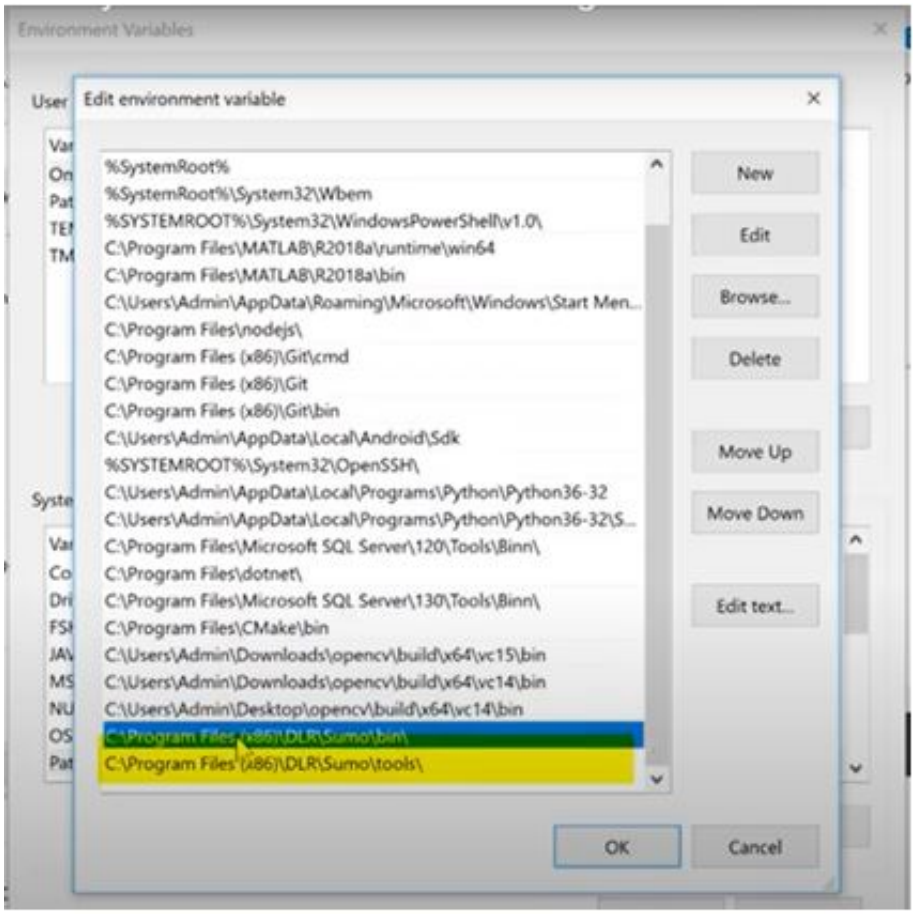

Figura 3.1 Visualización de las rutas a añadir.

Para poder trabajar correctamente con SUMO y sus herramientas se necesita tener instalado el entorno de programación de Python, este se puede descargar en cualquier web. Una vez instalado Python ya se podrá utilizar OSM Web Wizard con todas sus opciones de simulación.

3.3 Estructura de los ficheros de SUMO

Para realizar una simulación completa con SUMO se necesitan al menos 4 ficheros diferentes:

- **Fichero .net.xml:** Este se trata de un archivo de red que se encarga de definir el escenario de simulación con todas sus carreteras e intersecciones por las que circulan los vehículos.
- **Fichero .add.xml:** Este archivo se usa para definir elementos adicionales a la red como aros inductivos, semáforos y paradas de bus o de carga de vehículos eléctricos.
- **Fichero .rou.xml:** En este archivo se definen los tipos de vehículos y sus propiedades y las rutas que van a seguir los vehículos.
- **Fichero .sumocfg:** Fichero de configuración de SUMO donde se definen los ficheros de entrada y los ficheros de salida.

Dentro del fichero de red (.net.xml) se definen los nodos (intersecciones) de la red y las carreteras o calles (edge), por medio de estos se puede hacer por completo un escenario de simulación. Además en este fichero se puede definir las características de las diferentes carreteras como la longitud, el número de carriles, la prioridad de paso y las limitaciones de velocidad para las diferentes vías. Este se crea de varias formas, las cuales se explicarán más adelante.

En el fichero adicional(.add.xml) se suelen definir las diferentes zonas de parada de vehículos y permiten utilizar otras funciones de SUMO como el rerouter. Este fichero se ha usado en mis simulaciones principalmente para definir las estaciones de carga de los vehículos eléctricos y para ayudar a definir las rutas cíclicas de mis vehículos por medio de rerouter.

En el fichero de rutas (.rou.xml) se definen los diferentes vehículos y sus propiedades en función del tipo de vehículo que se quiera simular. También se definen en él las rutas que va a seguir cada vehículo y el instante en el que el vehículo va a aparecer o desaparecer del escenario de simulación. Existen muchas formas de definir las rutas de un vehículo y se explicarán más adelante.

Finalmente el fichero de configuración (.sumocfg) se utiliza para poder ejecutar correctamente la simulación, para ello se deben introducir los ficheros de entrada con su nombre completo. Estos ficheros son los que se han explicado anteriormente y cualquier fichero que sea necesario para la descripción del escenario o de los viajes de los vehículos. En este fichero también se puede indicar el tiempo de las simulación y si se quieren obtener datos durante la simulación sobre los vehículos se deben definir dentro del fichero de configuración los archivos de salida, que se encargará de mostrar datos durante la simulación.

3.3.1 Formas de generar rutas

Antes de crear las rutas que van a seguir los automoviles hay que definir los diferentes vehículos y sus propiedades por medio de los atributos *vclass* y *vtype*. Con estos se pueden definir vehículos privados, autobuses, turismos, etc. Una vez que se han definido los diferentes tipos de vehículos se puede utilizar el mismo tipo tantas veces como se quiera sin necesidad de definir otra vez por completo el vehículo y sus propiedades.

A la hora de definir las rutas a seguir por los vehículos se pueden utilizar varias opciones:

- **Manualmente:** Las rutas se definen para cada vehículo de forma manual indicando para cada uno de ellos el tiempo en el que aparece en la simulación y la ruta a seguir, que se introduce indicando por cada uno de los nodos o edge que debe pasar el vehículo.
- **Automáticamente:** Es la forma más rápida de generar las rutas de los vehículos ya que se hace de forma automática gracias al script de python *randomtrips.py*, este script permite generar rutas aleatorias y especificar varios parámetros como el instante concreto en el que generar el tráfico que se puede especificar mediante el periodo o mediante una variable aleatoria. Este programa también tiene en cuenta la longitud del carril y el número de carriles a la hora de generar el tráfico ya que se pueden controlar estos aspectos por medio de valores de probabilidad.

  El programa a la hora de generar las rutas utiliza 3 nodos diferentes que se toman de manera aleatoria, se trata del nodo de inicio el nodo de fin y el nodo intermedio por el que tiene que pasar el vehículo. La ruta se va a crear en función de estos 3 nodos y mediante un algoritmo se va a obtener la ruta más corta que une los 3 nodos.

  El script *randomtrips.py* genera un fichero del tipo *.trips* que se convierte a un fichero de ruta *.rou* por medio del comando duarouter, que se encargará de convertir los viajes aleatorios en

rutas y de descartar automáticamente aquellos viajes que no están completos porque no se haya conseguido conectar los 3 nodos generados aleatoriamente.

Existe otra forma alternativa de general rutas aleatorias pero sin necesidad de configurar los parámetros del srcipt *radomtrips.py*, esta forma consiste en utilizar la aplicación OSM Web Wizard y crear directamente con esta el escenario de simulación junto con las rutas, para ello solo se tendrían que ajustar unos parámetros dentro de la aplicación que indicarían los vehículos que se quieren simular y el flujo de vehículo que se quiere por paso de simulación. Esta aplicación utiliza de forma automática el script *randomtrips.py* cuando se le da a generar escenario una vez elegida la zona del mapa que se quiere simular. En el siguiente apartado donde se explican las diferentes formas que existen de generar una red se explicará cómo se puede realizar la creación de una simulación completa por medio de esta aplicación.

3.3.2 Generación de redes

Antes de comenzar a definir las rutas de los vehículos es necesario tener un escenario sobre el que definir las rutas para ello se necesita crear un fichero *.net.xml* donde se van a encontrar la definición de toda la red de carreteras y sus características. Existen varias formas de generar la red de carreteras:

- **NetEdit:** Esta es una aplicación que se incluye en el paquete instalado de SUMO, que nos permite modificar y crear redes de forma manual. NetEdit consiste en una interfaz de usuario que permite crear redes sin necesidad de utilizar código ya que se hace todo de forma gráfica esto hace que el crear redes desde cero con esta aplicación sea muy intuitivo pero a la vez muy trabajoso ya que se tienen que definir todos los parámetros de la red manualmente. Debido a esto esta aplicación es útil para modificar redes que ya han sido creadas y para crear redes muy simples. A continuación se puede ver una imagen de la interfaz de esta aplicación:

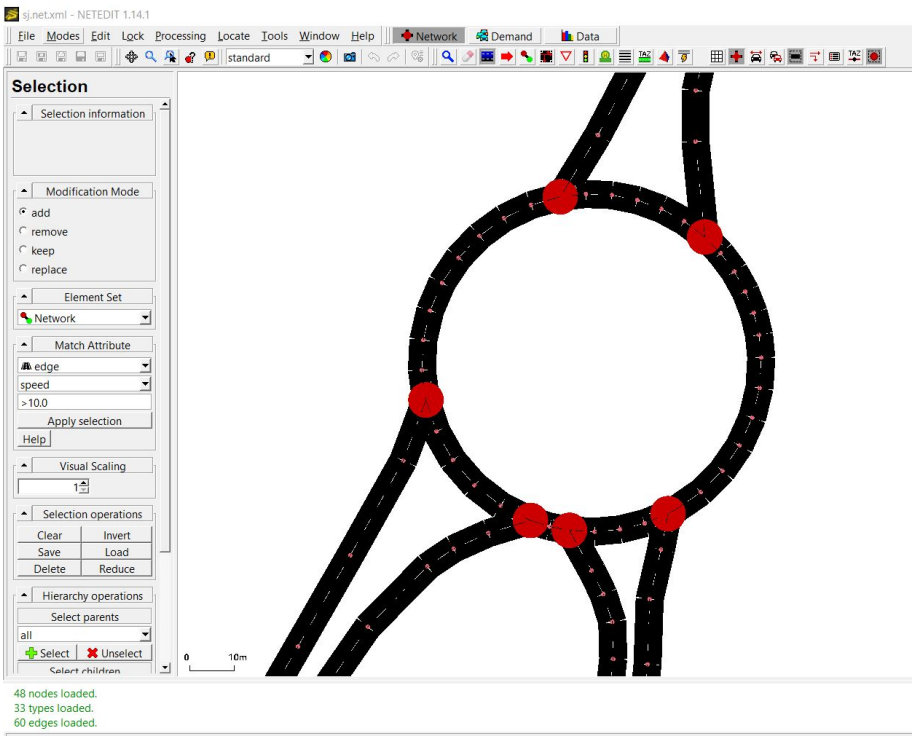

Figura 3.2 Interfaz de NetEdit.

- **Automáticamente con OSM Web Wizard:** Con esta aplicación se puede generar cualquier red de carreteras del mundo ya que SUMO gracias al script "netconvert" permite importar mapas de código abierto en formato OSM que se obtienen de OSM Web Wizard o de la aplicación similar Open Street Map. En la siguiente imagen se puede ver el mapa obtenido en formato *.net* por medio de esta aplicación.

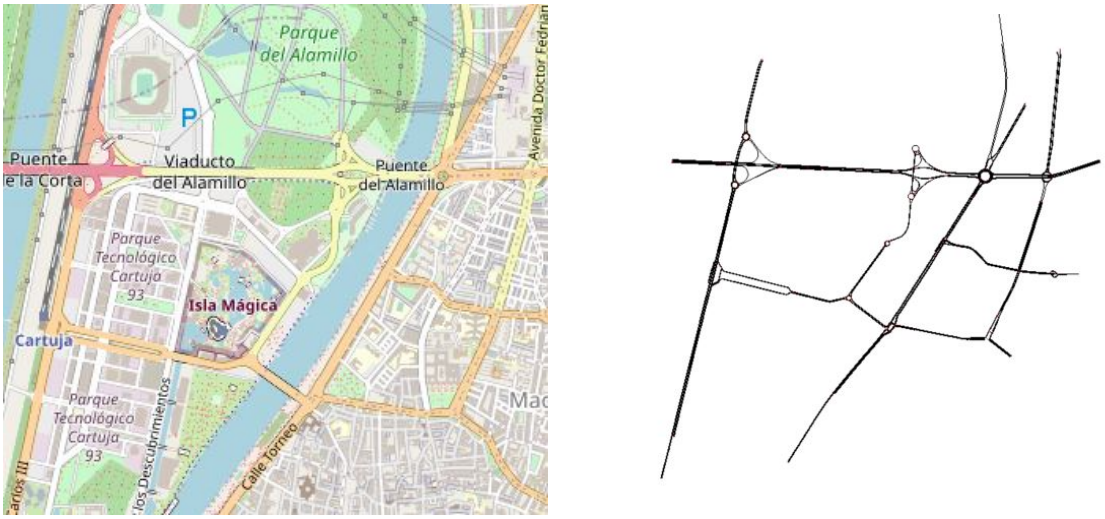

Figura 3.4 Mapa convertido de OSM Web Wizard al formato de SUMO.

Para las simulaciones se utilizará OSM Web Wizard ya que permite seleccionar cualquier zona del mundo a simular y establecer varios parámetros que se encargan de definir cómo será la visualización de la simulación, el flujo de vehículos y el tipo de vehículo que se van a incluir en el escenario.

Una vez que se ha elegido la zona a simular y se han establecido las diferentes opciones y el tiempo de simulación, se obtendrán cada uno de los ficheros necesarios para la simulación y el fichero *.sumocfg*, que si se ejecuta se podrá observar el funcionamiento de la simulación completa.

Figura 3.5 Interfaz de OSM Web Wizard.

Automáticamente con Netgenerate: Este script de SUMO permite generar redes de carreteras con formas típicas como redes en forma de malla, tela de araña o redes con forma aleatoria. Al definir el tipo de red que se quiere generar se pueden cambiar varios parámetros que depende exclusivamente de la topología de red elegida. En la siguiente imagen se pueden ver las diferentes topologías que se pueden generar:

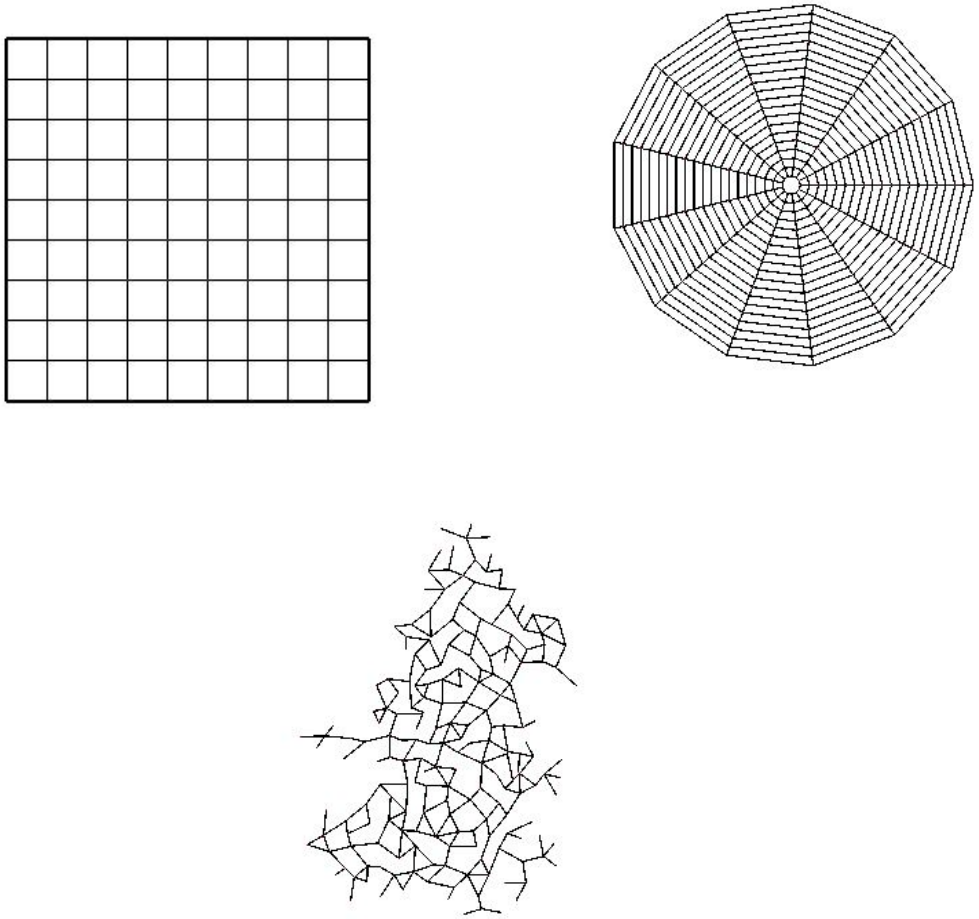

Figura 3.8 Redes generadas con "netgenerate" con forma de malla, tela de araña y aleatoria.

3.3.3 Fichero adicional (.add)

Este fichero se usa para complementar la simulación, en mi simulaciones como se indicó anteriormente se ha utilizado para definir las diferentes estaciones de carga, detectores de área y para crear "rerouter" que será esenciales para conseguir una ruta cíclica para cada vehículo.

- **Estaciones de carga:** Se trata de una serie de superficies que se definen en un carril específico del escenario donde los vehículos con baterías pueden cargarse durante el tiempo que se mantengan en la superficie de carga.

Al realizar la definición de la estación de carga hay que introducir varios parámetros de configuración como la eficiencia, la potencia, el tiempo que tarda desde que llega un vehículo y empieza a realizar la carga, etc. También hay que indicar el nombre de la estación, su ubicación es decir el carril donde se va a colocar y la posición en relación a ese carril donde comienza y termina la estación de carga.

A continuación, se puede ver un ejemplo del código que se necesita para definir una estación de carga:

```
<chargingStation chargeDelay="2" chargeInTransit="0" power
    ="200000" efficiency="0.95" endPos="25"
id="station1 " lane="2to19_0" startPos="10"/>
```

Una vez que se ha configurado la estación de carga se puede observar como en la visualización del escenario aparecerá un rectángulo en el carril especificado de color celeste lo que representará la estación de carga. Cuando sobre la estación de carga se sitúa un vehículo y este empieza a cargarse la visualización de la estación cambia de color con el objetivo de que se vea fácilmente cuando un vehículo se está cargando.

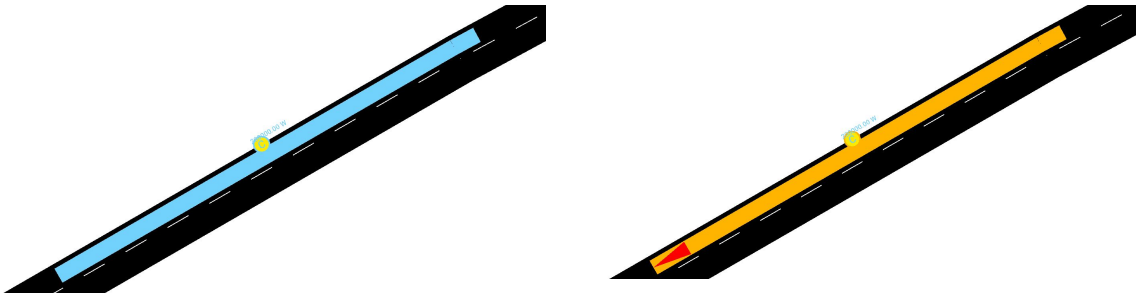

Figura 3.10 Visualización en SUMO de una estación de carga.

Si en la misma posición de la estación de carga se ha introducido un detector de área la zona de la estación de carga puede que no cambie de color cuando empiecen a cargar los diferentes vehículos. Por lo que en algunas simulaciones donde se utilicen los detectores de áreas no se va a observar en ningún momento que la estación de carga se ponga de color naranja.

- **Detectores de área:** Los detectores de área sirven para detectar los vehículos que pasan por un área concreta del escenario. En este caso se van a utilizar para detectar los vehículos que se han parado en la estación de carga y en el carril contrario a la estación. La idea principal de usar dichos detectores de áreas en la cercanía de las estaciones de carga es para detectar los vehículos que se paren de forma errónea en la estación de carga.

  Para realizar la configuración de los detectores de áreas se debe indicar el carril de la vía donde se va a situar, la posición inicial y final del detector con respecto al carril donde se ha situado, el fichero de salida donde se van a guardar los datos y la frecuencia de muestreo.

```
<laneAreaDetector id="station1" lane="22776976#0_0" pos="100"
    endPos="160" file="detectores.xml" freq="1"/>
```

- **Rerouter o redireccionador:** Es una función que nos aporta SUMO y que permite cambiar la ruta de destino del vehículo tan pronto como el vehículo se mueva hacia el borde que se

le ha especificado, esta función será muy útil para las simulaciones ya que el vehículo se quedará siempre circulando por la red y al no desaparecer se tendrá toda la información de su comportamiento durante toda la simulación.

Para realizar la configuración de un redireccionador hay que indicar el borde del escenario donde se quiere colocar y el borde al que se va a mover el vehículo cuando llegue al redireccionador, además se debe indicar el identificador del redireccionador y otros parámetros que son opcionales como la probabilidad de que se realice el redireccionamiento y el tiempo máximo que estará visible el redireccionador en el borde indicado, a continuación se puede ver un ejemplo de configuración:

```
<rerouter id="rerouter_20" edges="22776976#0">
 <interval end="1e4">
    <destProbReroute id="156217859#0"/>
 </interval>
</rerouter>
```

Si se ha configurado correctamente se puede visualizar en SUMO en el borde colocado, de la siguiente forma:

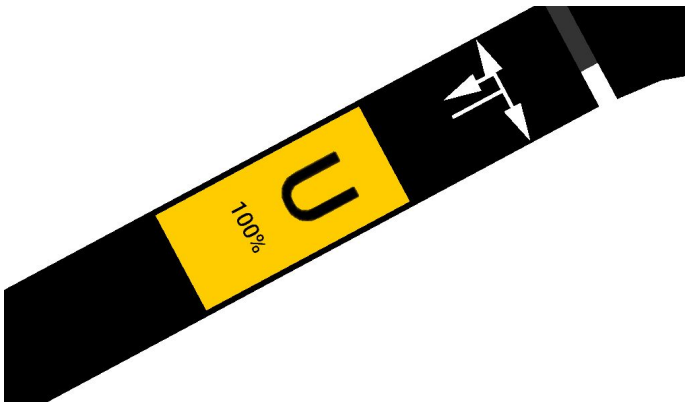

Figura 3.11 Visualización en SUMO de un redireccionador.

3.3.4 Fichero de simulación (.sumocfg):

Una vez que se han generado todos lo fichero necesarios para la simulación solo falta crear el fichero de configuración (.sumocfg), para ello se tiene que introducir como ya se indicó anteriormente todos los nombres de los ficheros necesarios para la simulación los cuales deben estar en la misma carpeta donde ha colocado el fichero de configuración. El fichero de configuración estará dividido en dos partes:

- **Entrada (input):** En esta zona se deben incluir los ficheros donde se encuentra la información para realizar la simulación (.rou, .add, .net, etc) y también se pueden incluir modificaciones acerca de la simulación final, en todas mis simulaciones se ha incluido una línea código que permite quitar el teletransporte de los vehículos cuando estos se paran, esto se explica más adelante en el apartado 4.1.4.

  A continuación, se muestra como se rellenaría el apartado de entradas del fichero de configuración.

```
<input>
   <net-file value="nombrefichero.net.xml" />
   <route-files value="nombrefichero.rou.xml" />
   <additional-files value="nombrefichero.add.xml" />
   <time-to-teleport value="-1"/>
</input>
```

- **Salida (output):** En esta parte se deben incluir los ficheros de resultados de la simulación que se encargan de guardar datos durante la simulación. En mis simulaciones se van a guardar los datos de las estaciones de carga y los datos de la batería que tienen los vehículos eléctricos, gracias a estos datos se pueden sacar conclusiones acerca del gasto de energía de cada vehículo y de la energía y el tiempo de carga que necesitan cada uno de los vehículos.

  Además, de introducir los ficheros de resultados se deben incluir varias líneas de códigos necesarias en las simulaciones, ya que estas líneas de código se necesitan para definir correctamente un vehículo como eléctrico y la precisión de los resultados.

  A continuación, se muestra el código del apartado de salida del fichero de configuración:

```
<output>
    <battery-output value="Battery.out.xml"/>
    <chargingstations-output value="chargingstations.xml"/>
    <battery-output.precision value="4"/>
    <device.battery.probability value="1"/>
    <summary-output value="summary_100.xml"/>
</output>
```

  Una vez creado por completo el fichero de configuración de SUMO existen dos formas de ejecutar la simulación por medio de la interfaz gráfica, esta permite visualizar la simulación y ver los problemas de tráfico que se pueden dar ya que en la visualización se puede ver en cada paso de simulación el comportamiento del vehículo.

  Una forma de ejecutar la simulación sería desde la línea de comando y la otra sería desde un script de un lenguaje de programación con el comando *sumo-gui -c nombrefichero.sumocfg*.

4 Resultados de simulación

En este capítulo se van a exponer las diferentes simulaciones de SUMO que se han realizado y se va a explicar con todo detalle cada uno de los ficheros que se han creado y como se han obtenido. En estas simulaciones se han utilizado script de lenguajes de programación como Matlab y Python para controlar en todo momento el tráfico de las simulaciones. Además se van a ir explicando los problemas que han ido surgiendo durante la programación de los diferentes ejemplos y cómo se han resuelto.

El objetivo de estas simulaciones se basa principalmente en realizar un estudio completo de las prestaciones de SUMO en el mundo de los vehículos eléctricos. A este objetivo se le une el de controlar el tráfico durante la simulación con programas como Matlab o Python actuando sobre los vehículos en función de la batería que disponen en cada instante.

Se buscan realizar simulaciones en la que se puedan observar un número considerable de vehículos moviéndose por una red de carreteras. Estos vehículos presentarán un seguimiento de su batería y su estado durante la simulación, gracias a este seguimiento se va a realizar un control que permita parar el vehículo cuando la batería se esté acabando en las diferentes estaciones de cargas que se han distribuido por la red de carreteras. Al mismo tiempo se va a monitorizar las estaciones de cargas teniendo en cuenta su capacidad, con el fin de que estas no se saturen. Debido a las dificultades que se han encontrado al trabajar con SUMO, el objetivo de la simulación final se ha dividido en varias simulaciones en las que se va a ir explicando los pequeños avances que se han logrado.

4.1 Simulación 1: Haciendo pruebas

En esta simulación se busca lo siguiente:

- Dotar a los vehículos de SUMO de una batería para que estos tengan propiedades de vehículos eléctricos.
- Leer en cada paso de simulación valores que nos indiquen el estado del vehículo.
- Probar las estaciones de carga que nos permite usar SUMO y como hacer que el vehículo se pare en ellas y retome la circulación cuando se ha cargado completamente.
- Poder ver gráficamente cuando un vehículo se está quedando sin batería.
- Realizar todo el control de los vehículo en tiempo real con Matlab.
- Hacer que los vehículos siempre estén en circulación para no perder la información sobre ellos y poder comparar los resultados al final de la simulación.

Para cada simulación se mostrará el código creado para cada tipo de fichero y el código que se ha utilizado para controlar el tráfico de la simulación.

4.1.1 Escenario (circles.net.xml)

Esta primera simulación sirve para probar las diferentes opciones que tiene SUMO y para aprender a utilizar las funciones de *traci4matlab*, por lo que se va a tomar un escenario sencillo. Además como se quiere ver el comportamiento de los vehículos en todo momento para poder comparar sus resultados, se busca que los vehículos tengan rutas cíclicas y siempre se encuentren en la visualización, por ello se va a utilizar una red octogonal que permite crear de forma más sencilla las rutas repetitivas. Esta red se va a crear por medio de NetEdit.

La red creada en NetEdit es la siguiente:

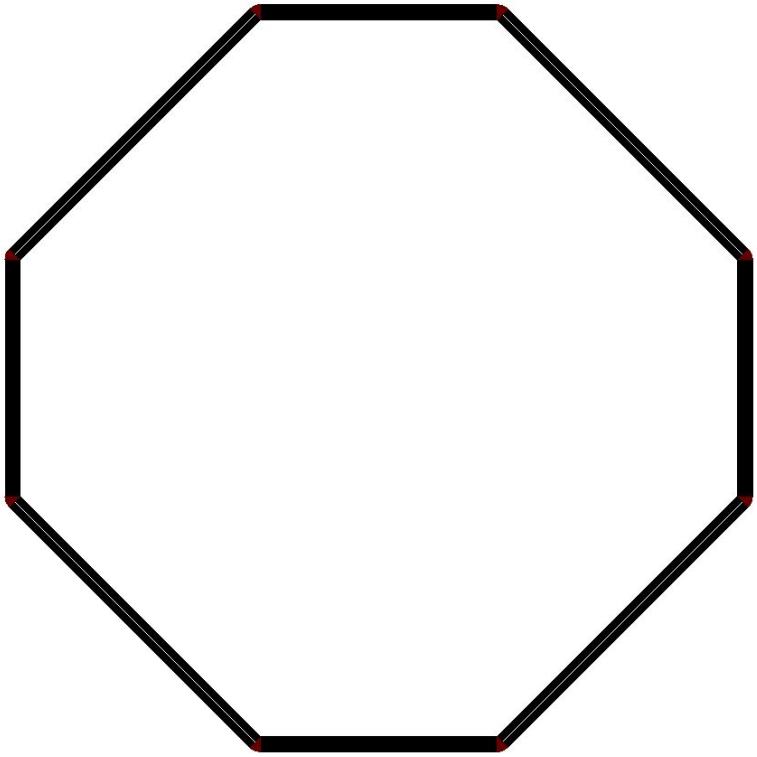

Figura 4.1 Visualización del escenario en NetEdit.

Esta red como se puede ver está formada por 8 tramos de carreteras de 2 carriles de un único sentido. La idea de que tuviera 2 carriles se debe a que al introducir la estación de carga esta ocupará un carril, por lo que si se parase un vehículo para cargarse, el vehículo que se encontrase detrás podría adelantar automáticamente y evitar la colisión. Por el contrario si solo hubiese un carril, al pararse un vehículo en la estación el siguiente también se pararía debido a una colisión.

Al crear un escenario con NetEdit se genera automáticamente un fichero .net.xml donde se encuentra todo el código que se necesita para generar dicha red.

4.1.2 Añadir estación de carga y rerouter (circles2.add.xml)

Como se explicó en el apartado 3.3.3, los ficheros adicionales se han utilizado con el fin de añadir y configurar la estación de carga en la red y para crear los diferentes redireccionamientos que se van a utilizar para crear las rutas repetitivas.

Se va a explicar por encima las dos partes del código de este fichero:

```
<chargingStation id="station1" lane="E85_0" startPos="20" endPos
   ="80" chargeDelay="2" chargeInTransit="0" power="200000"
               efficiency="0.95"/>
```

Con este código se añade al carril de la derecha (0) de la carretera (edge:"E85") una estación de carga con una potencia de 200.000 W y una eficiencia de carga de 0,95. En la siguiente tabla se explican los parámetros que se han utilizado para definir la estación de carga.

Tabla 4.1 Parámetros para la definición de una estación de carga.

Nombre	Descripción	Tipo	Valor por defecto
id	Identificador de la estación de carga	sting	-
lane	Carril donde ucbicar la estación	string	Id carril valido
startPos	Posición inicial de la estación en el carril especificado	float	0
endPos	Posición final de la estación en el carril especificado	float	-
chargeDelay	Retraso de tiempo antes de que empiece la transferencia de energía	float	0
chargeInTransit	Habilitar o desabilitar la carga en transito	bool	0
power	Potencia de carga de la estación $P_{chrg}(W)$	float	0
efficiency	Eficiencia de carga η_{chrg}	float	0.65

```
<rerouter id="rerouter_0" edges="edge1">
 <interval end="1e4">
    <destProbReroute id="edge2"/>
 </interval>
</rerouter>
 <rerouter id="rerouter_1" edges="edge2">
 <interval end="1e4">
    <destProbReroute id="edge1"/>
 </interval>
</rerouter>
```

Con estas líneas de código se consigue que al llegar el vehículo a una de las carreteras indicadas el vehículo pase a dirigirse a la otra carretera que se indica, de esta forma el vehículo circulará de la carretera llamada edge1 a la edge2 y de la edge2 a la edge1 y así sucesivamente creando una ruta repetitiva.

En las siguiente imágenes se indican donde se ha colocado la estación de carga y el redireccionamiento:

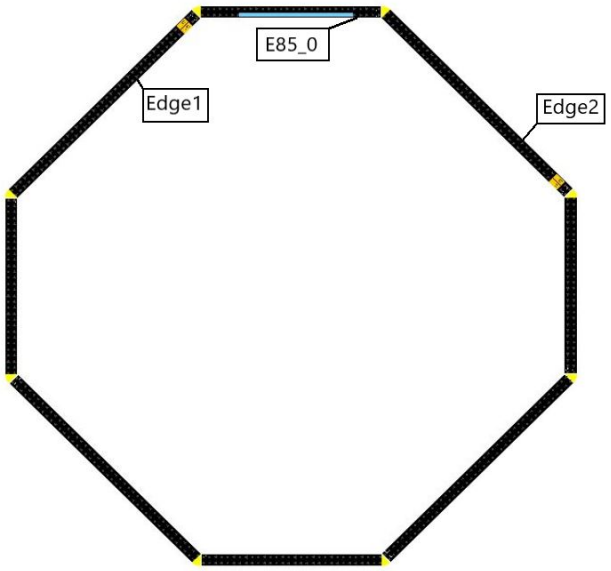

Figura 4.2 Visualización del escenario en NetEdit.

4.1.3 Propiedades de los vehículos y rutas (circles.rou.xml)

En este fichero se deben definir las propiedades del vehículo que se quiere simular, en este caso será un vehículo eléctrico por lo que se debe incluir en los vehículos la batería y todas las propiedades necesarias. La definición del vehículo se debe realizar entre las siguientes sentencias <vType>-</vType>.

En las siguientes líneas de código se puede ver como se puede definir un vehículo eléctrico por completo:

```
<vType id="car" vClass="passenger" length="5" accel="3.5" decel="2.2"
    sigma="1.0" color="0,0,1">
     <param key="has.battery.device" value="true"/>
    <param key="maximumBatteryCapacity" value="60000"/>
    <param key="maximumPower" value="272000"/>
    <param key="vehicleMass" value="1500"/>
    <param key="frontSurfaceArea" value="2.5"/>
    <param key="airDragCoefficient" value="0.6"/>
    <param key="internalMomentOfInertia" value="0.01"/>
    <param key="radialDragCoefficient" value="0.5"/>
    <param key="rollDragCoefficient" value="0.01"/>
    <param key="constantPowerIntake" value="100"/>
    <param key="propulsionEfficiency" value="0.9"/>
    <param key="recuperationEfficiency" value="0.9"/>
    <param key="stoppingTreshold" value="0.1"/>
</vType>
```

Se van a explicar el significado de cada uno de los parámetros que se han incluido en la definición del vehículo.

Tabla 4.2 Parámetros de la definición de un vehículo eléctrico.

Key	Descripción	Tipo	Valor por defecto
has.battery.device	Añadirle batería al vehículo	string	false
maximumBatteryCapacity	Capacidad máxima de la batería (Wh)	float	35000
maximumPower	Potencia máxima del vehículo (W)	float	150000
vehicleMass	Masa del vehículo (kg)	float	1830
frontSurfaceArea	Área frontal del vehículo $A_{veh}(m^2)$	float	2.6
airDragCoefficient	Coeficiente radial de arrastre C_{rad}	float	0.1
rollDragCoefficient	Coeficiente de resistencia a la rodadura C_{roll}	float	0.01
constantPowerIntake	Promedio de potencia consumida $P_{const}(W)$	float	100
propulsionEfficiency	Eficiencia de propulsión η_{prop}	float	0.98
recuperationEfficiency	Eficiencia de recuperación η_{recu}	float	0.96
stoppingTreshold	Velocidad mínima para empezar a cargar (km/h)	float	0.1

Los parámetros que se han descrito en la tabla anterior son los necesarios para configurar el vehículo como uno real y que por medio de un modelo y esas constantes se puedan obtener los gastos del vehículo. El primer parámetro de la tabla es el encargado de definir al vehículo como eléctrico ya que al añadir el dispositivo batería al vehículo este adquiere las propiedades de uno eléctrico.

Cualquiera de estos parámetros se pueden definir cuando se realice la definición del viaje que va a seguir el vehículo, normalmente el parámetro que se encarga de indicar la batería actual del vehículo se suele definir fuera de la definición del tipo de vehículo, como se puede ver en la siguiente página.

Tabla 4.3 Parámetros de la definición de el tipo de vehículo.

Nombre	Descripción	Tipo	Valor por defecto
id	Identificador del tipo de vehículo creado	id (string)	-
vClass	Clase de vehículo	class (enum)	"passenger"
length	Longitud del vehículo	float	5.0
accel	Aceleración máxima (m/s^2)	float	2.6
decel	Desaceleración máxima (m/s^2)	float	4.5
sigma	Forma de conducción (0, conducción perfecta)	float	0.5
color	Color del vehículo en RGB	"float,float,float"	"1,1,0" (amarillo)

En esta tabla se muestran los parámetros que se encargan de definir el tipo de vehículo y sus características principales, existen más parámetros para definir el tipo de vehículo, se pueden ver en la web de SUMO.

En este fichero además de definir el vehículo y sus propiedades, se debe definir las rutas que deben seguir cada vehículo. Existen varias formas de definir rutas, en este caso utilizamos para definir rutas repetitivas el siguiente código:

```
<route id="routecar1" edges="edge1 E85 edge2 E79 E80 E81 E82 E83"
   color="yellow" repeat="10" cycleTime="140">

</route>
```

En la siguiente tabla se describe el significado de los parámetros definidos en el anterior código:

Tabla 4.4 Parámetros utilizados para la definición de una ruta.

Nombre	Descripción	Tipo
id	Identificador de la ruta	id (string)
edges	Bordes por el que circulará el vehículo	id list
color	Color de la ruta	float or string
accel	Aceleración máxima (m/s^2)	float
repeat	Número de veces que se repetirá esta ruta (m/s^2)	int
cycleTime	Tiempo de duración del ciclo (s)	int (tiempo)

Esta forma de definir rutas repetitivas de vehículos no es muy útil, ya que para definir completamente la ruta se necesita indicar cada una de las vías por donde tiene que pasar el vehículo, lo que hace muy costoso el definir una ruta larga de esta forma. Se utiliza esta definición de rutas como ejemplo pero para crear rutas cíclicas se utilizará la función rerouter de SUMO.

A la hora de asignarle a un vehículo su ruta correspondiente se puede hacer de dos formas. Se puede asignar la ruta a un solo vehículo, entre las siguientes sentencias vehicle <vehicle->-</vehicle>:

```
<vehicle id="car1v" type="car1" depart="0.00" route="routecar1">
  <param key="actualBatteryCapacity" value="3250"/>
</vehicle>
```

Este código se utiliza para asignarle a un solo vehículo una ruta definida anteriormente, con este código en concreto se define un vehículo llamado "car1v" del tipo creado anteriormente "car1" y que comienza en el tiempo 0 con la ruta "routercar1". Con estas líneas de código también se indica la batería actual que debe tener el vehículo al comenzar su viaje.

Tabla 4.5 Parámetros utilizados para la asignación de una ruta (vehicle).

Nombre	Descripción	Tipo
id	Identificador del vehículo	id (string)
type	Identificado del tipo de vehículo que se va a usar	id
depart	Paso de tiempo en el que el vehículo entra en la red	float
route	Identificado de la ruta por la que circulará el vehículo	id

Estos parámetros son específicos de la asignación de una ruta a un vehículo. Mientras que el parámetro que se presenta en la siguiente tabla se trata de un parámetro propio de los vehículos eléctricos y que sirve para terminar con la definición de los vehículos eléctricos, indicando la energía actual de la batería cuando comienza a realizar su ruta asignada.

Tabla 4.6 Parámetro para definir la batería inicial del vehículo.

Key	Descripción	Tipo	Valor por defecto
actualBatteryCapacity	Batería actual del vehículo (Wh)	float	maximumBatteryCapacity/2

Tambien se pueden utilizar las sentencias <flow->-</flow> a la hora de asignar rutas a un vehículo, esta forma de asignación de rutas se suele usar cuando se quiere que varios vehículos sigan la misma ruta. Las sentencias "flow" describen flujos de vehículos.

```
<flow id="carflow" type="car" begin="0" end="0" number="2" from="edge
    1" to="edge2" >
      <param key="actualBatteryCapacity" value="2500"/>
</flow>
```

Con estas líneas de código se define un flujo de 2 vehículos llamado "carflow" del tipo "car" que inicia su viaje en el instante 0 desde el borde de carretera edge1 hasta edge2.

Tabla 4.7 Parámetros utilizados para la asignación de una ruta (flow).

Nombre	Descripción	Tipo
id	Identificador del flujo de vehículos, cada vehículo se genera con este nombre	id (string)
type	Identificado del tipo de vehículo que se va a usar	id
begin	Tiempo de inicio del flujo descrito	int
end	El tiempo de finalización del flujo descrito	int
number	Número de vehículos que se insertarán en este flujo	int
from	Nombre del borde donde comienza la ruta	edge id
to	Nombre del borde donde termina la ruta	edge id

Cada uno de los vehículos de este flujo tendrán una batería de 2500 Wh al comenzar su viaje.

Tabla 4.8 Parámetro para definir la batería inicial del vehículo.

Key	Descripción	Tipo	Valor por defecto
actualBatteryCapacity	Batería actual del vehículo (Wh)	float	maximumBatteryCapacity/2

4.1.4 Configuración de SUMO (circles.sumcfg)

Como ya se explicó al presentar la estructura de una simulación en SUMO en el fichero de configuración se deben incluir los ficheros que se han creado para esta simulación. En este caso los ficheros que se van a introducir como entradas entre las sentencias <input>-</input> son los ficheros "circle.net.xml", "circle.rou.xml" y "circle2.add.xml". Además de estos ficheros se va a incluir una línea de código que permite eliminar el teletransporte de los vehículos que se produce cuando el vehículo se queda parado en medio de la carretera.

- **Teletransporte de los vehículo**

  Este teletransporte se produce cuando el vehículo se para, obstaculizando el paso del resto de vehículos. En esta simulación el vehículo se va a parar cuando este se quede sin batería o cuando esté parado en la estación de carga. Cuando este se para, el programa lo detecta y automáticamente hace que el vehículo se teletransporte a otra posición ya que ha detectado que el vehículo se encontraba parado en una intersección.

  Cuando esto se produce la propia aplicación de SUMO te advierte de que se ha producido este teletransporte, ya que te muestra lo siguiente en la interfaz de SUMO:

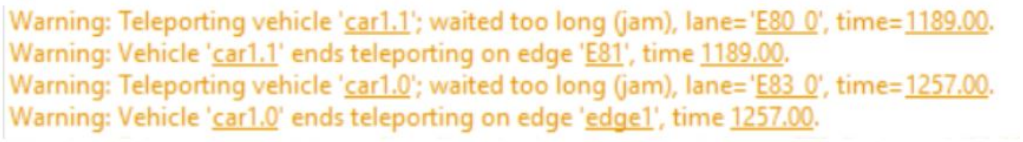

Figura 4.3 Advertencia de teletransporte en la interfaz de SUMO .

El teletransporte se efectúa cuando un contador llega al máximo impuesto por defecto en este caso será 300 segundos, el contador cuenta cada vez que el vehículo esté parado consecutivamente con una velocidad menor de 0,1 m/s. Este teletransporte lo realiza el programa ya que intenta quitar el vehículo de la ruta derivando a este a cualquier salida libre, si no existe una salida libre este seguirá en bucle teletransportandose cada 300 s (valor máximo del contador por defecto) o hasta que se acabe la simulación donde se produce la desaparición de todos los vehículos.

El valor del umbral del contador se puede modificar con el comando "–time-to-teleport <INT>" , que establece el tiempo máximo en segundos del umbral, si ponemos un valor negativo se deshabilita el teletransporte debido al bloqueo de cuadrícula.

Como se quiere evitar el teletransporte se introduce entre las sentencias <input>-</input> del fichero de configuración la siguiente línea de código:

```
<time-to-teleport value="-1"/>
```

De esta forma ya se habrá arreglado el problema del teletransporte, por lo que los vehículos se quedarán permanentemente parados en la posición donde se quedaron sin batería inicialmente o en la posición de carga.

Además de incluir los ficheros para realizar la simulación se deben incluir ficheros de resultados que se van a generar una vez que haya terminado la simulación. Estos ficheros se suelen incluir entre las sentencias <output>-</output>. Con el siguiente código se obtiene información sobre la energía que hay en cada momento en cada una de las baterías de los coches.

```
<battery-output value="Battery.out.xml"/>
```

Para obtener información sobre los vehículos que se han parado en las diferentes estaciones de carga y ver la batería que se ha transferido al vehículo en cada momento se introduce el siguiente código:

```
<chargingstations-output value="chargingstations.xml"/>
```

Finalmente el último fichero que se obtiene se trata de un resumen sobre la simulación que carece de interés en el contexto actual.

```
<summary-output value="summary_100.xml"/>
```

Entre las sentencias <output>-</output> se han incluido dos líneas de código para terminar con la definición de los vehículos eléctricos. La que se presenta a continuación sirve para equipar a todos los vehículos con batería para que adquieran las propiedades de vehículo eléctrico.

```
<device.battery.probability value="1"/>
```

Y la siguiente y última línea de código del fichero de configuración se utiliza para definir la precisión con la que se escribe los valores de la batería dentro del fichero de seguimiento de la energía de consumo de la batería.

```
<device.battery.probability value="1"/>
```

Al completar el fichero de configuración la simulación ya estará en condiciones de ser ejecutada y lista para ser controlada por medio de programas externos como Python o Matlab.

4.1.5 Control con Matlab

Una vez creada la simulación se va a realizar el control por medio de Matlab utilizando las funciones de *traci4matlab* que permite conectar Matlab con las funciones de *traci* que permiten el control de la simulación de SUMO.

Antes de comenzar a explicar el programa completo de control que se ha realizado, se va a explicar los principales aspectos y directrices que hay que seguir para comenzar a programar el control de SUMO con Matlab.

Para comenzar a programar en Matlab y controlar una simulación es necesario importar las constantes de *traci* al principio del fichero, esto se hace con la siguiente línea de código:

```
import traci.constants
% esto se incluye siempre para poder importar las constantes de traci
```

Traci es la abreviatura "Traffic Control Interface" (interfaz de control de tráfico), que permite recuperar valores de objetos simulados y manipular su comportamiento en línea.

Una vez que se han importado las constantes de *traci* se utiliza la función "traci.start" para iniciar sumo-gui con el archivo de configuración (.sumocfg).

```
%se utiliza para poner en marcha sumo por medio del fichero
%sumocfg que he generado previamente
traci.start('sumo-gui -c ./nombredelfichero.sumocfg --start');
```

Esta llamada de inicio se usa para conectar nuestro script con sumo-gui ejecutándose como servidor.

Cuando se haya importado las constantes de *traci* y añadido el nombre del fichero de configuración el script estará listo para programarse, para programar hay que tener en cuenta que para ejecutar la simulación de SUMO hay que hacerlo por pasos de tiempo gracias a la función "traci.simulationStep". La simulación en Matlab se realiza desde un ciclo principal que se puede realizar utilizando “for” ya que se simulará por medio de un bucle hasta un tiempo determinado. Otra forma sería simular hasta que todos los vehículos hayan llegado a su destino para la cual se utilizaría la función "traci.simulation.getMinExpectedNumber". En todas mis simulaciones se utilizará un bucle for hasta un tiempo determinado que será el fin de la simulación.

```
for i = 1: duration-1
       traci.simulation.step();%se realiza un paso de simulación
       .
       .
       .

end
```

Además de utilizar este ciclo principal que se encarga de los pasos de simulación, se va a utilizar otro ciclo anidado al principal, que se comportará como ciclo secundario y que se encargará de recorrer en cada paso de simulación todo el vector donde se encuentran los identificadores de cada vehículo. Con este nuevo bucle "for" se consigue que en cada paso de simulación del primer bucle "for" se puedan controlar los diferentes vehículos. El ciclo secundario terminará cuando se hayan recorrido todos los vehículos que se encuentran en simulación.

```
for i = 1: duration-1
       traci.simulation.step();%se realiza un paso de simulación
       n=length(ID);

           %bucle para trabajar con los diferentes vehículos
         for j=1:n

          vehID=ID{j};
           .
           .
           .
         end

  end
```

Si se hacen estos dos ciclos puede que el programa no funcione correctamente, porque en el primer paso de simulación no se encuentren los diferentes identificadores de todos los vehículos. Para solucionar esto se recomienda hacer un primer bucle al comienzo de la simulación que permita simular los primeros pasos de simulación y guardar todos los identificadores de los vehículos, de esta forma cuando se cree el ciclo secundario ya se tendrá el vector completo de los identificadores de cada vehículo y el tamaño de este vector que será el que marcará el final del ciclo secundario.

A continuación, se muestra el pequeño código del bucle que se ha utilizado para solucionar el error.

```
%hago esto para darle valor a la variable ID y poder saber el
%número de coches que se van a simular
for i=1:10
traci.simulation.step();
ID=traci.vehicle.getIDList();
end
```

Como se puede observar en el código anterior se utiliza la llamada "traci.vehicle.getIDList" para obtener los identificadores de cada vehículo, los identificadores se guardan en un vector de celdas en el que cada coponente es una cadena de carácteres que se correponde con el nombre que se le ha dado a cada vehículo que se encuentra en ruta.

Al final del código cuando haya acabado el bucle principal se cierra la conexión con el servidor de SUMO y se detiene por completo la simulación con el siguiente código de *traci*.

```
traci.close()
```

A continuación se van a explicar los diferentes script de control que se han realizado con Matlab.

Cambio de color en función del nivel de batería

Se va a realizar un script de Matlab con el fin de probar cómo se leen ciertos parámetros desde Matlab y como se pueden cambiar las diferentes propiedades de un vehículo. Para esta simulación se van a utilizar los siguientes ficheros:

- **Escenario (cricles.net.xml):** Este fichero define el escenario de la simulación, en este caso será un escenario circular para que sea sencilla la simulación y la visualización de ella.
- **Fichero adicional (cricles.add.xml):** En este fichero se definen los rerouter que se van a utilizar en esta simulación.
- **Vehículos y rutas (circles.rou.xml):** Se definen para esta simulación dos vehículos con idénticas propiedades y 4 rutas. Por lo que en esta simulación al existir 4 rutas habrá 4 vehículos en movimiento.
  Los vehículos de la simulación:

Tabla 4.9 Vehículos definidos en esta simulación .

IDveh	Tipo	Ruta	Batería inicial (W)
car1v	carv	Circular alrededor de todo el escenario	3250
car1v2	carv	Circular alrededor de todo el escenario	3250
carflow.1	car	Circular alrededor de todo el escenario	2500
carflow.2	car	Circular alrededor de todo el escenario	2500

  El tipo de vehículo hace referencia a las propiedades que se le han dado al vehículo, estas se pueden observar en el código que se adjunta en el anexo.
- **Configuración (circles.sumocfg):** En este fichero de configuración hay que introducir los nombres de los ficheros creados y de los que se quieren generar.

Ya que tenemos preparado el entorno de simulación se va a explicar el programa que se ha realizado en Matlab. En este script se busca leer la capacidad de la batería de cada vehículo en cada instante de tiempo y en función de los valores de esta se va a cambiar el color del vehículo. De este modo el vehículo pasará por 3 fases distintas que se verán reflejadas en la simulación por el color rojo, naranja y azul, cada color representa un nivel distinto de batería, los niveles se dividen en 3 intervalos:

- **Nivel de batería alto:** Se representa con el color rojo, un vehículo tendrá un nivel alto de carga si el vehículo tiene una energía disponible mayor que 2/3 de batería inicial.
- **Nivel de batería medio:** Se representa con el color naranja, un vehículo tendrá un nivel medio de carga si el nivel de energía del vehículo se encuentra entre 2/3 y 1/3 de la batería inicial.
- **Nivel de batería bajo:** Se representa con el color azul, un vehículo tendrá un nivel bajo de carga si el nivel de energía del vehículo es menor que 1/3 de la batería inicial.

Si el vehículo pasa por los 3 niveles y antes de que se termine la simulación se queda sin energía este se quedará parado permanentemente. Para obtener la carga de la que dispone el vehículo en cada instante se va a utilizar la siguiente línea de código:

```
eng_act(j,i)=str2double(traci.vehicle.getParameter(vehID,'device.battery
    .actualBatteryCapacity'));
```

Con la función "traci.vehicle.getParameter" se puede obtener cualquier parámetro que se necesite de un vehículo, en este caso se utiliza la función para obtener la energía actual del vehículo. Al llamar a la función devuelve el valor de energía actual de un vehículo eléctrico en formato de cadena de caracteres por lo que para poder operar con ese dato se decide pasarlo al formato "double".

Se ha creado una matriz con tamaño j*i donde j se corresponde con el número de vehículos que hay en ruta e i con el tiempo que se ha simulado, con esta se consigue guardar cada una de las energías que se han obtenido para cada vehículo e instante.

Una vez que se sabe la energía que tiene cada vehículo en cada instante solo se tiene que ver entre que intervalos se encuentra dicha energía y en relación al intervalo en el que se encuentre se mostrará el vehículo con un color u otro. Para cambiar el color del vehículo se utiliza la función "traci.vehicle.setColor" que permite indicar al vehículo que se le quiere cambiar el color y el color que se le quiere poner, el color se introduce entre corchetes y hay que introducirlo respetando el formato RGBA.

```
traci.vehicle.setColor(vehID,[0,0,255,255]);
```

Como se comentó anteriormente cuando el vehículo se quede sin batería hay que hacer que el vehículo se quede parado permanentemente ya que al no tener batería no se podrá seguir moviendo, para ello se cambia la velocidad del vehículo a 0 con la función "traci.vehicle.setSpeed" que permite cambiar la velocidad de cualquier vehículo que se encuentre en la simulación.

Resultados:

En este apartado se van a mostrar diferentes imágenes y gráficas donde se vea claramente el funcionamiento correcto del script de matlab que se ha realizado. En este caso se van a mostrar imágenes donde se vea el cambio de color de los vehículos y gráficas donde se vea la energía que tenía el vehículo en el instante que cambia de color.

Se van a incluir varias imágenes de la interfaz de SUMO donde se observa el color del vehículo y el instante de tiempo de la simulación:

- En la primera imagen 4.4 se puede ver como al comienzo de la simulación todos lo vehículos son de color rojo porque no han gastado mucha batería.
- En la segunda imagen 4.5 hay dos vehículos de color rojo porque siguen en el nivel alto de batería y dos vehículos de color naranja porque ya han pasado al nivel medio.
- En la tercera imagen 4.6 se pueden ver como todos los vehículos son de color naranja, por lo que hay dos vehículos que en la imagen anterior eran de color rojo (nivel alto) que han pasado a ser de color naranja (nivel medio).
- En la cuarta imagen 4.7 se pueden ver dos vehículos de color azul (nivel bajo) y otros dos de color naranja (nivel medio).
- En la quinta imagen 4.8 todos los vehículos son de color azul (nivel bajo) aunque dos de ellos si se observan las gráficas 4.11 se puede ver como su energía es 0, por lo que permanecerán parados en esa misma posición al quedarse sin batería.
- En la sexta imagen 4.9 los vehículos siguen con el color azul pero esta vez todos están parados porque se puede ver como su energía es 0 al observar las gráficas 4.11.
- La última imagen 4.10 se ha añadido para comprobar que los vehículos siguen en la misma posición que en la imagen anterior ya que se han quedado sin batería. También se puede ver que el valor del tiempo de simulación es de3000 segundos por lo que la simulación ha finalizado.

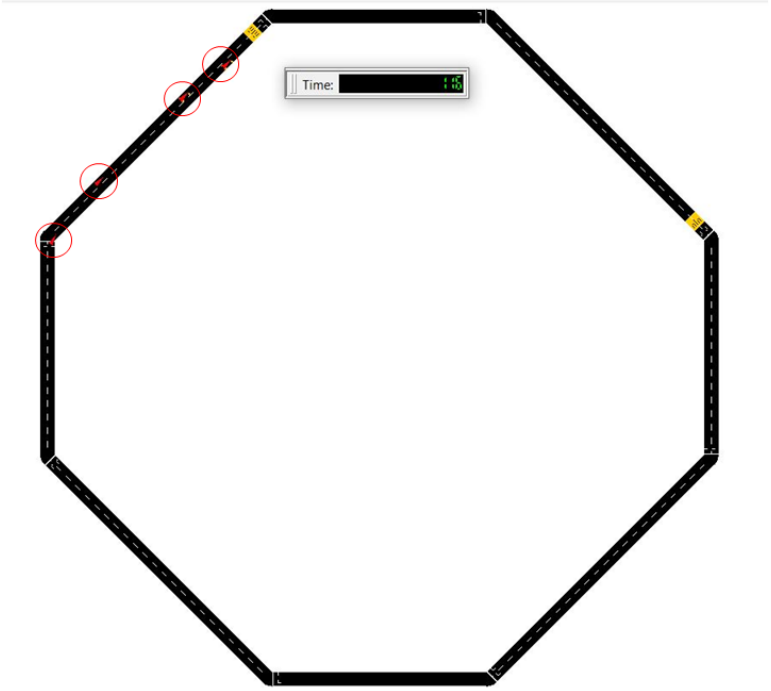

Figura 4.4 Visualización de la simulación en SUMO: 1º-Vehículos rojo.

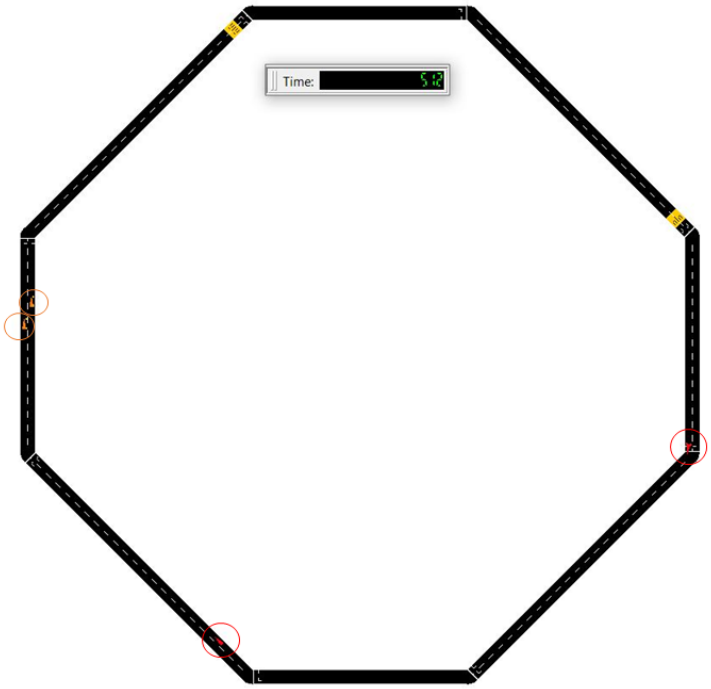

Figura 4.5 Visualización de la simulación en SUMO: 2º-Vehículos rojo y naranja.

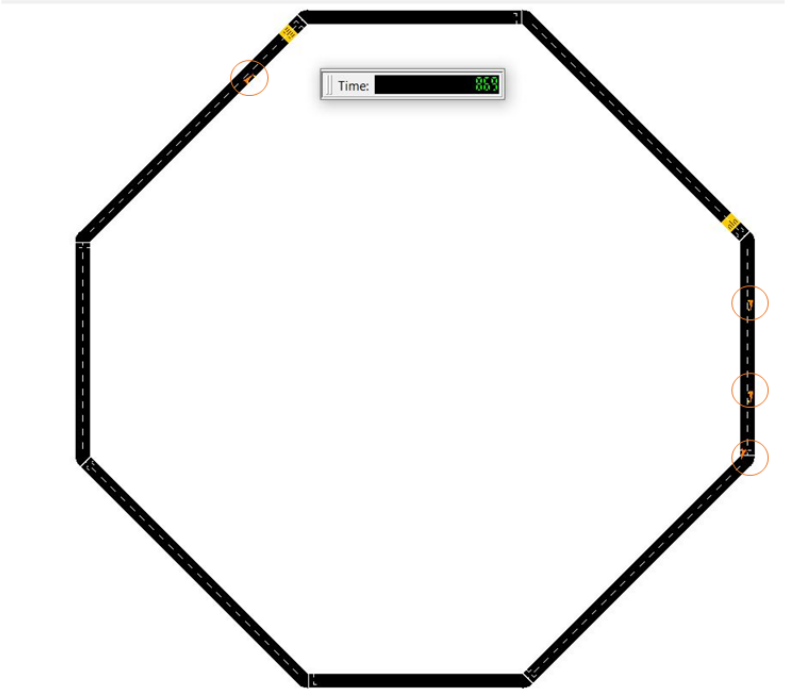

Figura 4.6 Visualización de la simulación en SUMO: 3º-Vehículos naranja.

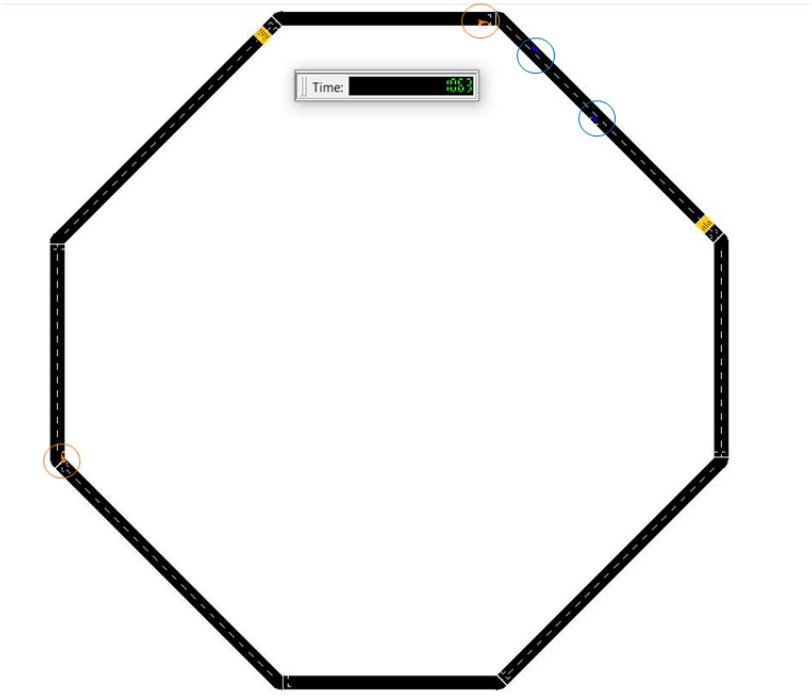

Figura 4.7 Visualización de la simulación en SUMO: 4º-Vehículos naranja y azul.

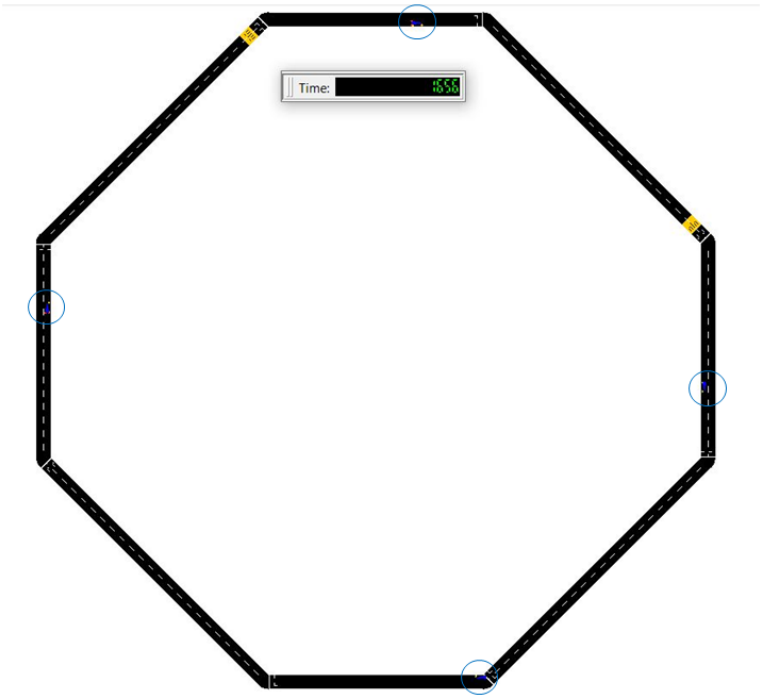

Figura 4.8 Visualización de la simulación en SUMO: 5º-Vehículos azul y 2 sin batería.

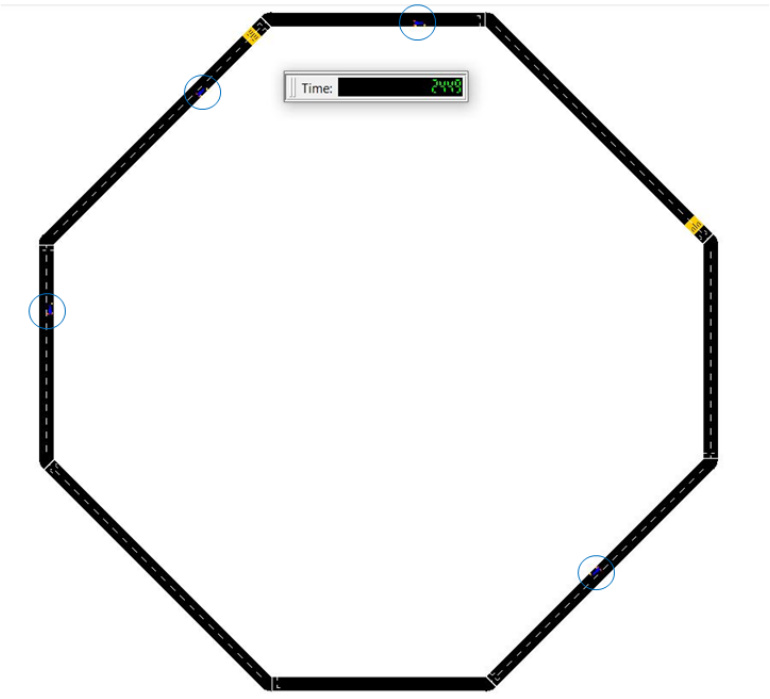

Figura 4.9 Visualización de la simulación en SUMO:6º-Vehículos azul y sin batería.

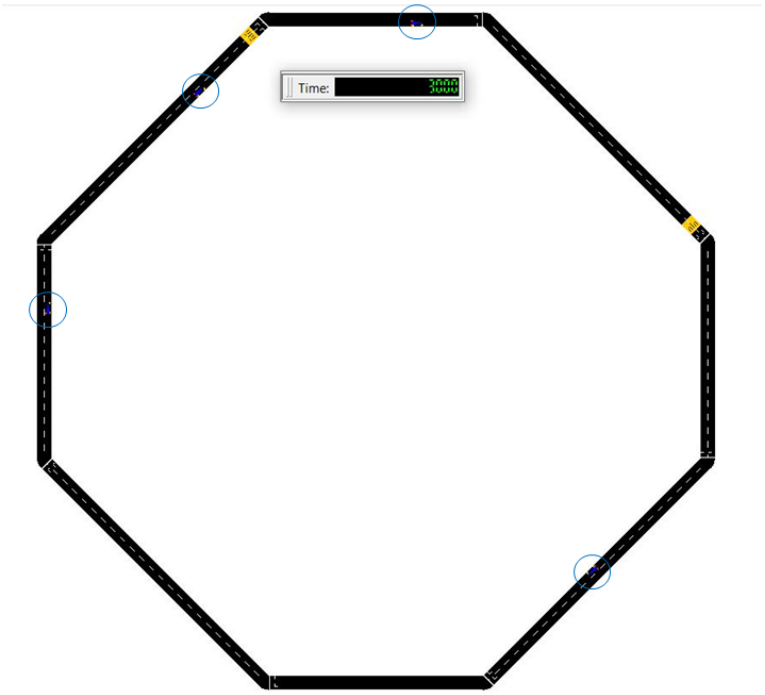

Figura 4.10 Visualización de la simulación en SUMO: 7º-Vehículos azul y fin de la simulación.

Todas las imágenes representadas anteriormente se han tomado en distintos tiempos de simulación, por lo que para ver que el color del vehículo corresponde con el nivel de batería que tiene se va a representar 4 gráficas donde se puede observar la batería de cada uno de los vehículos en cada instante de tiempo.

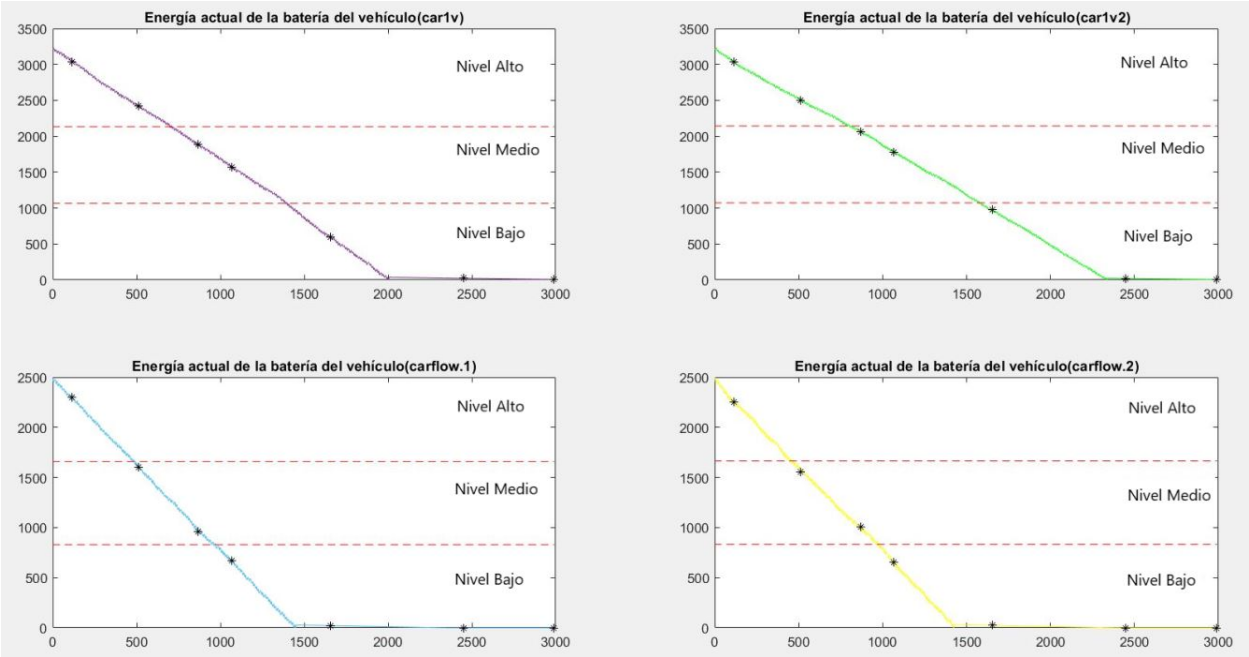

Figura 4.11 Gráficas que representan la energía de la batería de cada vehículo en cada instante de tiempo.

Las marcas realizadas con asteriscos ayudan a seguir las imágenes que se han mostrado anteriormente, ya que dicha marca muestra que batería tiene cada vehículo en los instantes de tiempo en los que se sacaron las diferentes imágenes de la simulación. Estas marcas ayudan a ver con facilidad que el color del vehículo se corresponde con el nivel de batería que dispone.

Las marcas que aparecen en las gráficas están en orden con las imágenes que se han mostrado ya que la primera imagen se corresponde con la primera marca y así sucesivamente.

Prueba de la parada de un vehículo en una estación de carga

Se va a realizar un script de Matlab con el fin de probar cómo se realizaría la parada de un vehículo en una estación de carga. Para esta simulación se van a modificar algunos ficheros antes de comenzar a realizar el script ya que se busca un escenario sencillo a la hora de probar la función de parada de los vehículos.

A continuación, se van a comentar los cambios que se han realizado en los ficheros:

- **Escenario (cricles.net.xml):** El escenario de simulación es el mismo que se ha utilizado anteriormente, por lo que seguirá siendo el escenario circular.
- **Fichero adicional (cricles2.add.xml):** Como se necesita una estación de carga y un rerouter se va a utilizar el fichero explicado en el apartado 4.1.2.
- **Vehículos y rutas (circles2.rou.xml):** Con el objetivo de facilitar el seguimiento de la simulación se ha reducido el número de rutas y vehículos, por lo que se ha modificado el fichero circle.rou.xml que se utilizó en el programa anterior. El nuevo fichero llamado circles2.rou.xml solo tendrá definido un tipo de vehículo eléctrico y una sola ruta.

```
<vType id="car1" vClass="passenger" length="5" accel="3.5" decel
   ="2.2" sigma="1.0" color="1,0,0">
      <param key="has.battery.device" value="true"/>
     <param key="maximumBatteryCapacity" value="60000"/>
     <param key="maximumPower" value="272000"/>
     <param key="vehicleMass" value="1500"/>
     <param key="frontSurfaceArea" value="2.5"/>
     <param key="airDragCoefficient" value="0.6"/>
     <param key="internalMomentOfInertia" value="0.01"/>
     <param key="radialDragCoefficient" value="0.5"/>
     <param key="rollDragCoefficient" value="0.01"/>
     <param key="constantPowerIntake" value="100"/>
     <param key="propulsionEfficiency" value="0.9"/>
     <param key="recuperationEfficiency" value="0.9"/>
     <param key="stoppingTreshold" value="0.1"/>
</vType>
```

```
<flow id="carflow" type="car1" begin="0" end="0" number="1" from
   ="edge1" to="edge2" >
     <param key="actualBatteryCapacity" value="1000"/>
</flow>
```

Al existir un solo vehículo este será con el que se va a trabajar en la simulaciones y esto permitirá depurar más fácilmente el código y ver si funciona correctamente la parada del vehículo y su posterior carga en ella.

- **Configuración (circles2.sumocfg):** Este fichero será muy parecido al circles.sumocfg ya que la mayoría de los ficheros que se han utilizado son los mismos, solo se tendrán que cambiar los nombres de los fichero que había antes por los que se han creado para esta simulación.

Como ya se ha dicho, el objetivo principal de este script es probar la función "traci.vehicle.setStop" que es la encargada de establecer la parada de un vehículo y hacer un pequeño programa en el que se controle cuando se para el vehículo en la estación de carga y durante el tiempo que se mantiene en dicha estación. Para ello hemos definido un solo vehículo con una sola ruta, al existir solo un vehículo solo se tiene que utilizar en el script un solo bucle "for" principal. La estructura general suele ser idéntica en todos los script de matlab que se han realizado para trabajar con SUMO.

En este script se indica que en un instante concreto de la simulación el vehículo debe pararse en la estación de carga justo cuando pase por ella. Para realizar la parada utilizamos la función "traci.ehicle.setStop" esta permite que el vehículo haga una parada durante un tiempo determinado y en la estación que se ha indicado, para introducir estos términos en la función hay que tener en cuenta el orden y la forma.

- **Función "traci.vehicle.setStop":**

Tabla 4.10 Parámetro de la función "traci.vehicle.setStop".

Orden	Nombre	Tipo	Descripción
1º	vehID	string	Id del vehículo que realiza la parada
2º	edgeID	string	Id de la estación de carga
3º	pos	double	Posición de parada en el carril
4º	laneIndex	integer	Carril donde se realiza la parada
5º	duration	double	Duración de la parada
6º	flags	integer	Valor entero que indica que tipo de parada se está realizando
7º	startPos	double	Posición inicial en el carril de la estación de carga
8º	until	double	Hasta que tiempo de simulación dura la parada

Pueden existir 7 tipos diferentes de paradas, estas se indican por medio de la variable que se introduce en 6º lugar en la función:

Tabla 4.11 Diferentes tipos de paradas (flags)".

Valor	Nombre	Descripción	Cambio
1	parking	La parada es de estacionamiento	–
2	triggered	–	–
4	containerTriggered	–	–
8	busStop	Para de autobus para recoger personas	edgeID a busStopID
16	containerStop	Parada para cargar	edgeID a containerStopID
32	chargingStation	Parada en una estación de carga	edgeID a chargingStationID
64	parkingArea	Parada en un área de parking	edgeID a parkingAreaID

De la tabla anterior referente a los diferentes flags solo se utilizará en los códigos el valor 32 ya que es el que hace referencia a la parada de un vehículo en una estación de carga.

A continuación, se puede observar un ejemplo de como usar la función "traci.vehicle.setStop":

```
traci.vehicle.setStop(ID{1},'station1',20,0,400,'32',60,0);
```

Con la función explicada anteriormente se hace que el vehículo se pare en la estación de carga, pero si se llama a esta función en cualquier momento de la simulación sin saber si el vehículo se encuentra cerca de la estación de carga se dará un error.

Error interfaz de matlab:
"110 196 255 Error chargingStation station1 for vehicle carflow.0 on lane E85_0 is not downstream the current route."

Error interfaz de SUMO:
"Error: Answered with error to command 0xc4: chargingStation station1 for vehicle carflow.0 on lane E85_0 is not downstream the current route."

Para solucionar este error se tiene que asegurar que el vehículo se encuentre pasando por la estación de carga en el momento que se llame a la función "traci.vehicle.setStop", para ello se lee en todos los pasos de simulación y se guarda en una matriz la posición del vehículo y por medio de un condicional se distingue si el vehículo se encuentra pasando por la estación de carga o no.

```
%matriz 2 por N nos aporta la posición x en todo isntante
posi1(:,i)=traci.vehicle.getPosition(ID{1})';

 %se asegura que le vehículo se encuentre en la posición de la estación
    de carga
if posi1(1,i)>=20 && posi1(1,i)<65 && posi1(2,i)>200 && posi1(2,i)<300
   .
   .
   .
end
```

Una vez solucionado el error hay que indicarle al vehículo cuando se tiene que parar, esto se va a realizar activando una variable llamada "parada", cuando el índice del bucle principal tenga un valor determinado ya que este índice se corresponde con el paso de simulación. En el siguiente fragmento de código se puede ver como se activa la variable parada igualándola a 1 cuando el paso de simulación es mayor que 451.

```
            if i>451
               parada=1;
            end
```

Resultados:

En este apartado se van a mostrar imágenes y gráficas donde se vea el funcionamiento de la simulación y el control de esta por medio del script que se ha realizado.

A continuación se muestran imágenes de momentos importantes de la simulación:

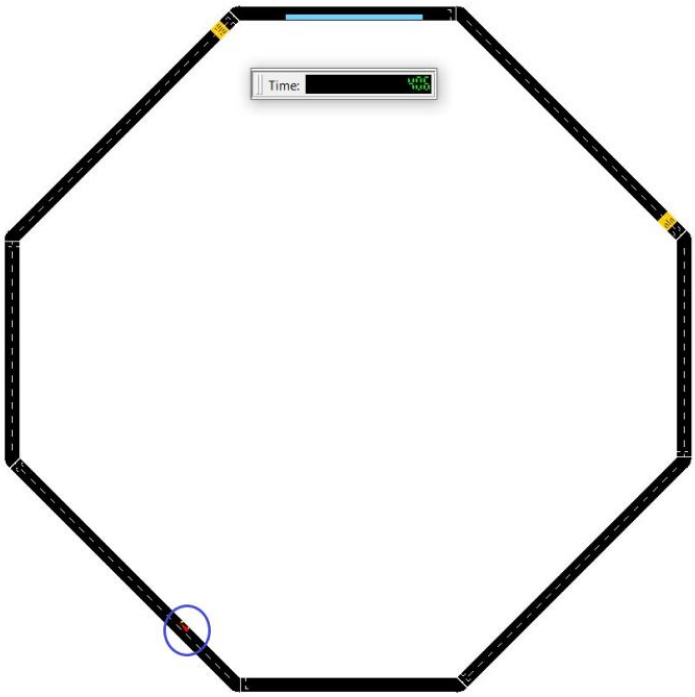

Figura 4.12 Visualización de la simulación en SUMO: Vehículo en movimiento.

En la figura 4.12 se puede ver como el vehículo sigue moviéndose por el escenario, ya que la imagen se ha hecho cuando el paso de simulación es menor que 452 por lo que todavía no se ha activado la variable de parada.

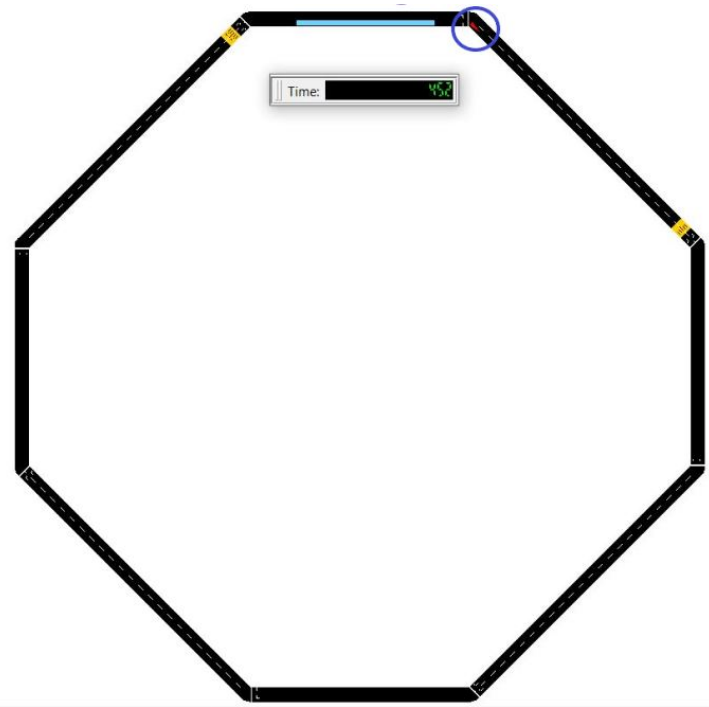

Figura 4.13 Visualización de la simulación en SUMO: Vehículo sigue moviendose.

En la imagen 4.13 ya se ha puesto la variable parada a 1 ya que el paso de simulación es 452, pero el vehículo no se ha parado porque no se encuentra en la vía donde se ha situado la estación de carga. Para que le vehículo realice la parada para cargarse debe pasar nuevamente por la vía que da acceso a la estación de carga.

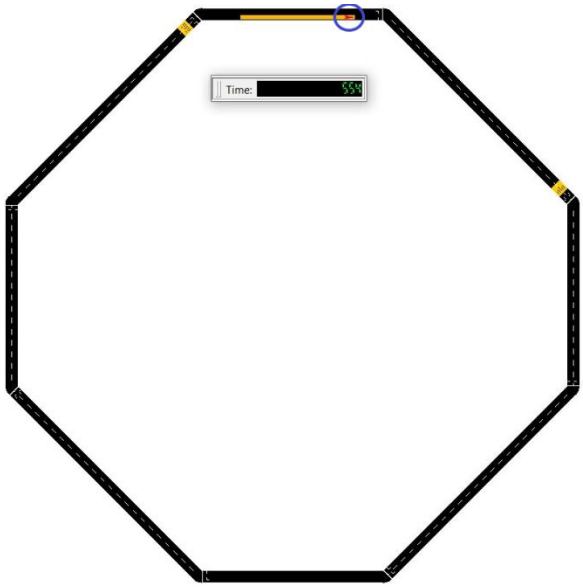

Figura 4.14 Visualización de la simulación en SUMO: Vehículo cargando en la estación de carga.

En la imagen 4.14 el vehículo se puede ver como se encuentra parado en la estación de carga, esto sucede porque el vehículo ha vuelto a pasar por la estación de carga mientras estaba activada la variable parada. Cuando se para el vehículo este debe cargarse durante 400 segundos que es el tiempo que se ha indicado en la función "traci.vehicle.setStop", como se puede ver en el código 4.1.5.

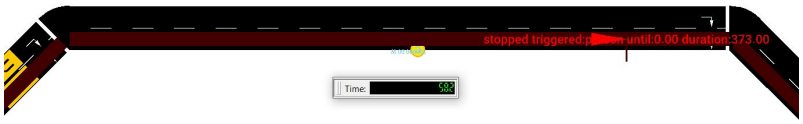

Figura 4.15 Visualización de la simulación en SUMO: Se ve la ruta del vehículo y la cuenta de la duración de la parada..

Al mirar en la interfaz de SUMO se ve como el vehículo tiene descrita una ruta, en la que se indica un valor de 373 que será el tiempo que le queda al vehículo para que salga de la estación de carga. Esta cuenta comienza a decrementar en uno por cada segundo y cuando llega a 0 el vehículo debería retomar la circulación, porque ya habrá terminado el tiempo de parada que se indicó. Pero existe un error que hace que el vehículo siga parado una vez que ha terminado la cuenta de parada. Como se puede observar en la siguiente imagen el vehículo sigue parado y la cuenta sigue decrementando su valor y toma valores negativos.

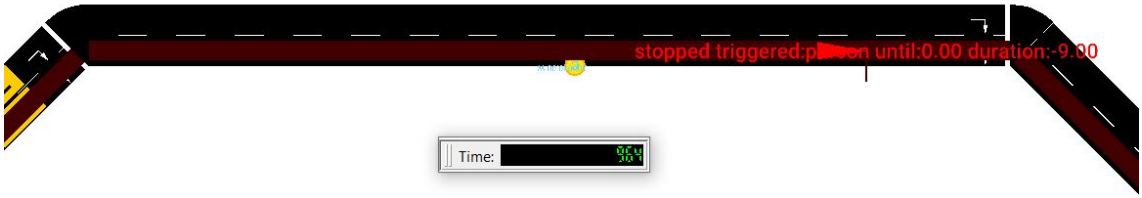

Figura 4.16 Visualización de la simulación en SUMO: Se ve la ruta del vehículo y la cuenta de la duración de la parada con valor negativo.

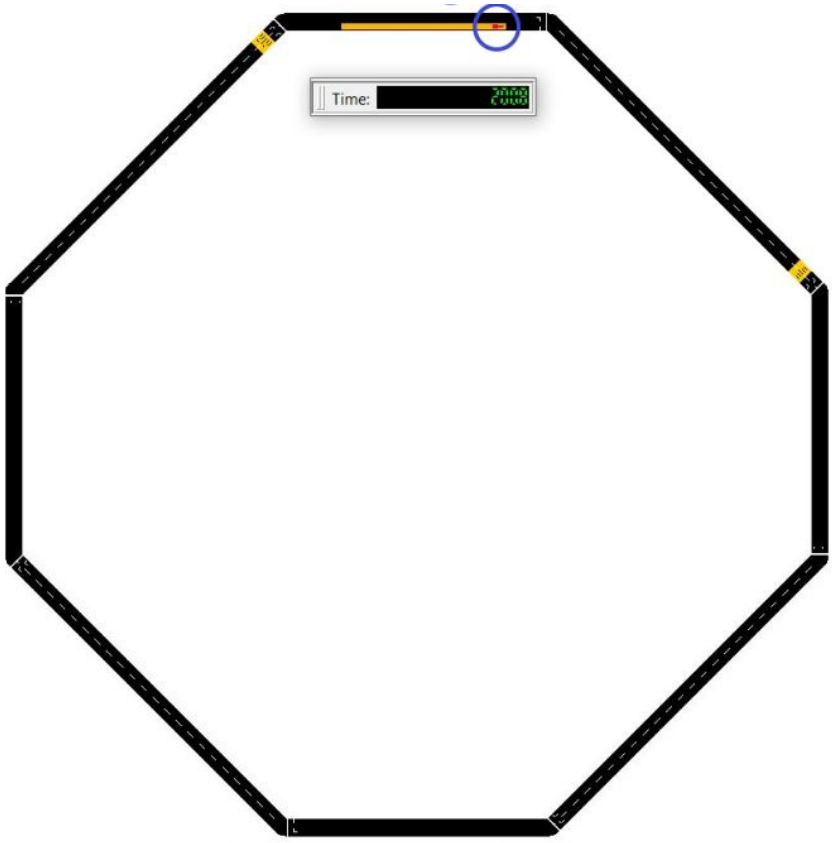

Figura 4.17 Visualización de la simulación en SUMO: 1º Vehículo en movimiento.

En esta última imagen 4.18 se puede ver como el vehículo sigue parado en la estación de carga aunque el tiempo de parada de 400 segundo haya pasado. Se ha investigado dicho error y no se ha encontrado solución ya que el error debe ser interno y propio de la comunicación entre Matlab y Python por lo que se ha decidido probar esta misma simulación en el entorno de Python que es el lenguaje que han utilizado los programadores de SUMO para desarrollar la interfaz de control de SUMO.

También se puede ver gráficamente el comportamiento del vehículo, para ello se han creado dos gráficas con Matlab en la que se pueden observar la velocidad del vehículo en todo momento y la energía que tiene el vehículo.

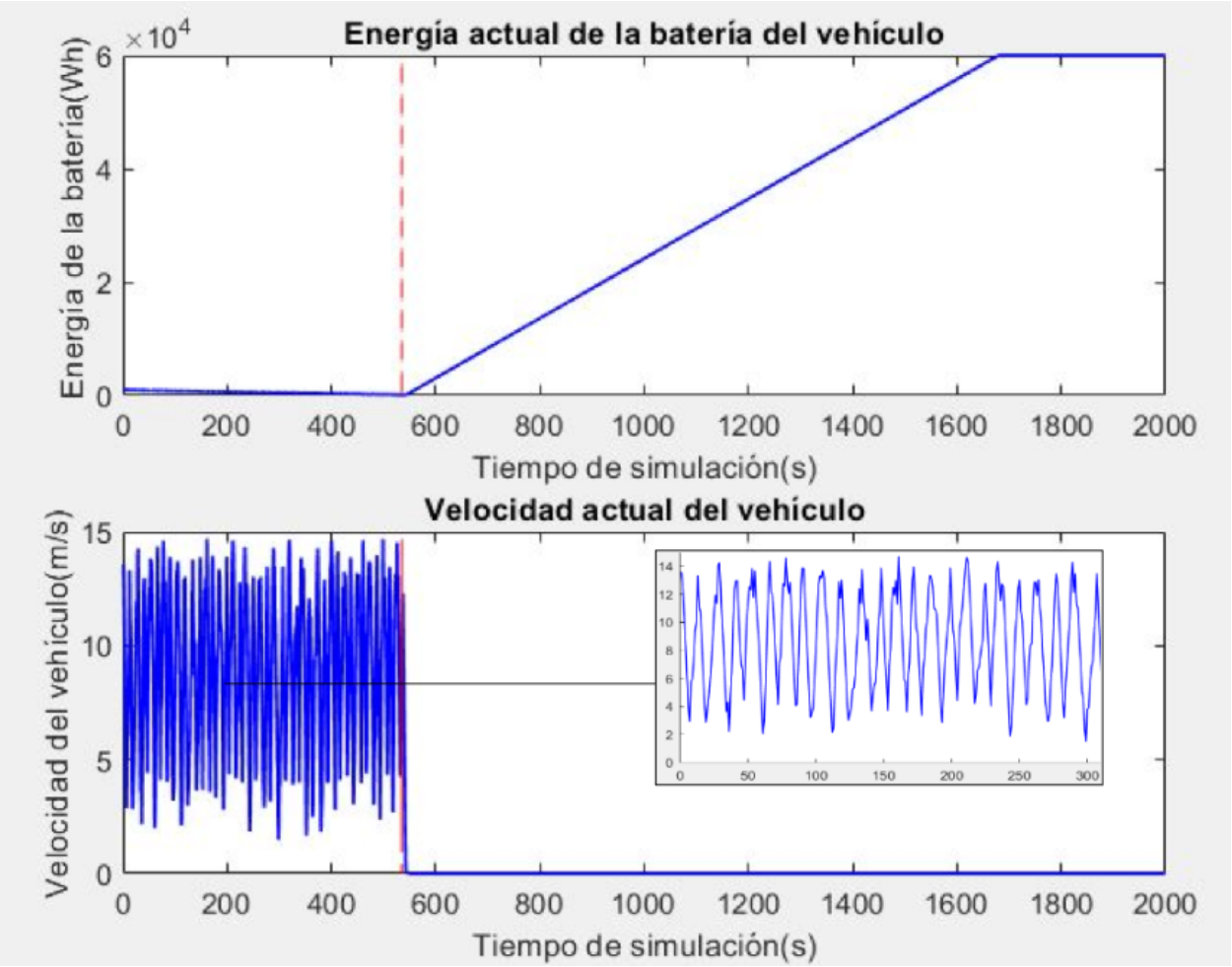

Figura 4.18 Gráfica donde se observa la velocidad y la energía del vehículo en cada instante.

En la gráfica se puede observar como la velocidad del vehículo oscila entre dos valores generándose una señal muy ruidosa, esto se debe a que los vehículos no se mueven a velocidad constante ya que su velocidad depende de los obstáculos que haya en la vía, de las intersecciones y de la propia vía. Por ello los vehículos van a circular a diferentes velocidades ya que al realizar una curva o al encontrar un obstáculo bajan considerablemente su velocidad. Si se observa el zoom que se ha realizado a la señal ruidosa de la gráfica se puede ver ese cambio de velocidad que experimenta el vehículo, la pendiente que se ve en la gráfica se asemejaría a la aceleración que sufre el vehículo para cambiar de una velocidad a otra.

En estas gráficas 4.18 se puede ver como el vehículo pasa a tener una velocidad de 0 m/s, esto se debe que le vehículo ha llegado a la estación de carga y se ha parado. Cuando este se para se puede observar un aumento de forma lineal en la gráfica donde se representa la energía actual del vehículo, esto se debe a que le vehículo a comenzado a cargarse. También se puede observar en esta misma gráfica como a partir del tiempo de simulación 1600 la gráfica se empieza a mantener constante, se debe a que la batería del vehículo se ha cargado por completo y ya no acepta más carga.

Además en estas gráficas se puede observar con más claridad el error que se indicó anteriormente, ya que el vehículo supera los 400 segundos de parada y no vuelve a moverse. Para mostrar el funcionamiento correcto de este ejemplo se ha decidido comenzar a programar este mismo ejemplo en Python.

4.1.6 Control con Python

Se va a realizar el control de la simulación anterior por medio de Python utilizando las funciones de *traci* que es una interfaz de control de tráfico creada por SUMO, el módulo *traci* se obtiene automáticamente al instalar SUMO. Para programar con Python se ha utilizado el entorno de programación Spyder 1.3 junto con la versión 3.9 de Python. Antes de comenzar a explicar el programa completo de control que se ha realizado, se van a explicar al igual que se hizo con Matlab las directrices que hay que seguir para empezar a programar el control de SUMO con Python.

Antes de comenzar a controlar una simulación de SUMO con Pyhton, se deben importar una serie de librerías:

```
import os
import sys
import traci
import sumolib
```

Estas librerías son necesarias para realizar la conexión con el servidor de SUMO y poder lanzar las simulaciones con la interfaz de SUMO.

Para poder iniciar la simulación con el fichero de configuración (.sumocfg) deseado, se utiliza al igual que en Matlab la función traci.start.

```
traci.start([sumolib.checkBinary('sumo-gui'), "-c", "nombredelfichero.
    sumocfg"] + sys.argv[1:])
```

Una vez ejecutada esta sentencia ya se puede comenzar a realizar el control de la simulación, al haber realizado la conexión del script de python con sumo-gui (interfaz de SUMO).

A la hora de controlar la simulación como se hizo con Matlab, hay que tener en cuenta que la ejecución de SUMO se hace por medio de pasos de tiempo que se realizan gracias a la sentencia " traci.simulationStep()". Por ello la simulación se va a iniciar con un bucle principal donde se ejecuta en cada ciclo del bucle la sentencia de paso de simulación. A este bucle se le anida un bucle secundario, encargado de recorrer el vector formado por identificadores de los diferentes vehículos que hay en simulación.

```
for i in range(N-1): #N:duración de la simulación
    traci.simulationStep() #se realiza un paso de simulación
    for j in range(n): #n:número de vehículos en la simulación
      vehID = vehicleIDs[j]
    .
    .
    .
```

A estos dos bucles hay que añadirle un bucle inicial encargado de inicializar todas la variables, ya que para ejecutar el bucle secundario es necesario conocer todos los vehículos que hay en la simulación. Por ello se ejecuta un bucle hasta un tiempo de simulación suficiente en el que ya se pueden conocer todos los vehículos que se van a introducir en la simulación, esto es necesario porque hasta que el vehículo no entre en circulación no se puede conocer su identificador.

```
for i in range(x): #se ejecutan x pasos de simulación
    traci.simulationStep()
```

```
#tras ejecutar esos pasos de simulación ya se conocen el conjuto de vehí
    culos totales
#que habrá en circulación dentro de la simulación
vehicleIDs = traci.vehicle.getIDList() #se leen los vehículos
```

Al final del código para poder cerrar la conexión con el servidor de SUMO se debe introducir la siguiente sentencia:

```
traci.close()
```

Prueba de la parada de un vehículo en una estación de carga

Se va a realizar un script de Python para probar que se puede realizar la parada de un vehículo dentro de una estación de carga. Esta simulación se va a realizar como consecuencia del error que se daba al realizar una para de un vehículo en una estación por medio de Matlab. Por lo que todos los fichero que se van a usar para realizar la simulación son idénticos a los descritos al realizar la simulación con Matlab.

A continuación, se van a mostrar los ficheros que se han utilizado en la simulación:

- **Escenario (cricles.net.xml):** El escenario de simulación se corresponde con el escenario octogonal utilizado anteriormente.
- **Fichero adicional (cricles2.add.xml):** Este fichero permite añadir una estación de carga y crear los rerouter como se hizo para controlar la parada de un vehículo con Matlab.
- **Vehículos y rutas (circles2.rou.xml):** Se utiliza un fichero con una sola ruta de vehículo como se hizo anteriormente, esto se hace para facilitar el control y probar si funciona correctamente la parada del vehículo en la estación de carga.
- **Configuración (circles2.sumocfg):** El fichero de configuración también será idéntico al utilizado en el sigiente apartado 4.1.5.

Como ya se ha explicado anteriormente el objetivo principal de esta simulación es solucionar el error al usar la función de parada de vehículos con Matlab. Por ello se va a crear un código muy similar al creado en el apartado 4.1.5, en el que se va a utilizar la función “traci.vehicle.setStop” encargada de realizar la parada del vehículo dentro de la estación de carga.

El script de python tiene la estructura descrita en la sección 4.1.6 pero con una diferencia, al no utilizar el bucle secundario ya que solo se va a trabajar con un vehículo.

La idea principal del código es la misma que se explicó en el control con Matlab, por lo que se va hacer un pequeño resumen del funcionamiento. En un instante de simulación determinado por la variable “i” se activa “charge” que se encarga de indicar que el vehículo debe pararse en la estación.

```
if 451<i<500:
  charge=True
```

Por lo que una vez activa la variable cuando el vehículo vuelva a pasar por la estación se parará. Para realizar la parada se llama a la función “traci.vehicle.setStop” indicando el tiempo que va a durar la parada.

```
if 20 >= vehPosLead[0] < 65 and 200 < vehPosLead[1] < 300:
  if charge:
    traci.vehicle.setStop(vehicleIDs[0],'station1',20,0,400,32,60,0)
    charge = False
```

Tras llamar a la función parada se desactiva “charge” para evitar que el vehículo se vuelva a parar, con esto se consigue ver si la parada se realiza correctamente respetando el tiempo de parada que se ha asignado.

Resultados:

Se van a mostrar una serie de imágenes y gráficas donde se vea que el funcionamiento de la simulación es correcto.

A continuación se muestran imágenes de momentos importantes de la simulación:

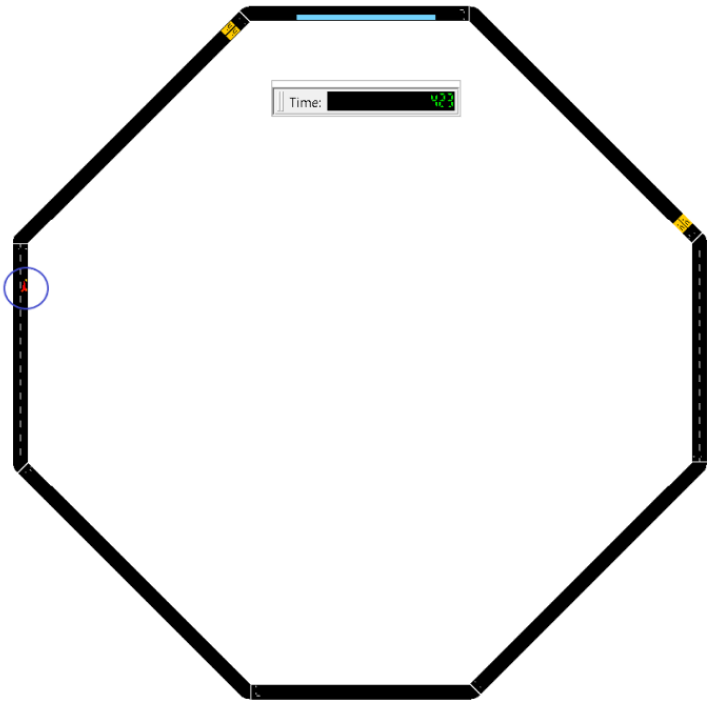

Figura 4.19 Visualización de la simulación en SUMO: Vehículo en movimiento.

En la imagen 4.19 se puede ver como el vehículo ha pasado por la estación y no se ha parado, esto se debe a que todavía no se ha llegado al tiempo de simulación 451 donde se activa la variable “charge”.

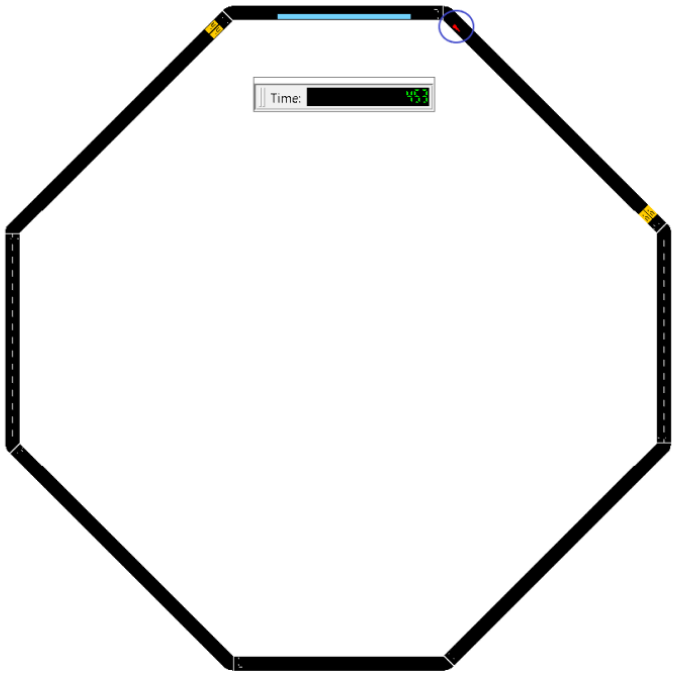

Figura 4.20 Visualización de la simulación en SUMO: Vehículo sigue en movimiento.

En la figura 4.20 se muestra como el tiempo de simulación es mayor que 451, por lo que ya se puede asegurar que el vehículo se va parar en la estación cuando vuelva a pasar por ella.

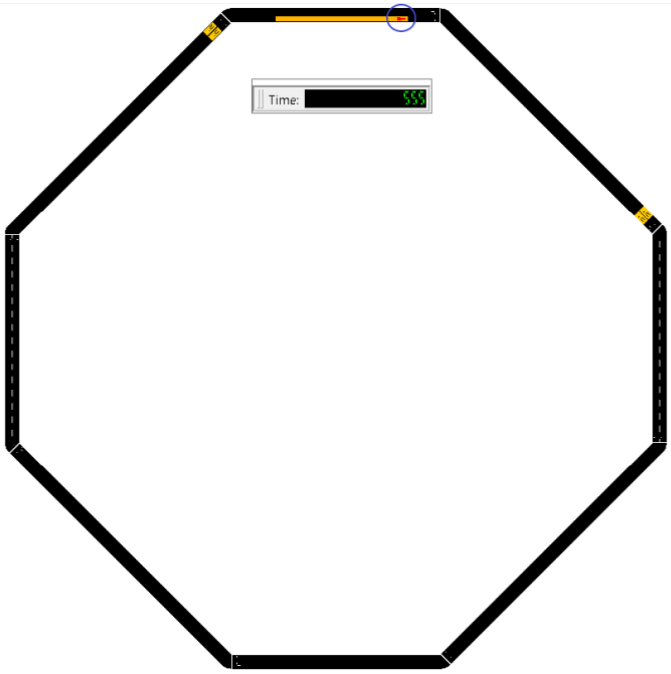

Figura 4.21 Visualización de la simulación en SUMO: Vehículo parado en la estación de carga.

En la figura 4.21 se puede observar como el vehículo se ha parado en la estación de cargar. La parada en la estación de carga se va a realizar durante 400 segundos al ser el tiempo que se le ha indicado al llamar a la función “traci.vehicle.setStop”.

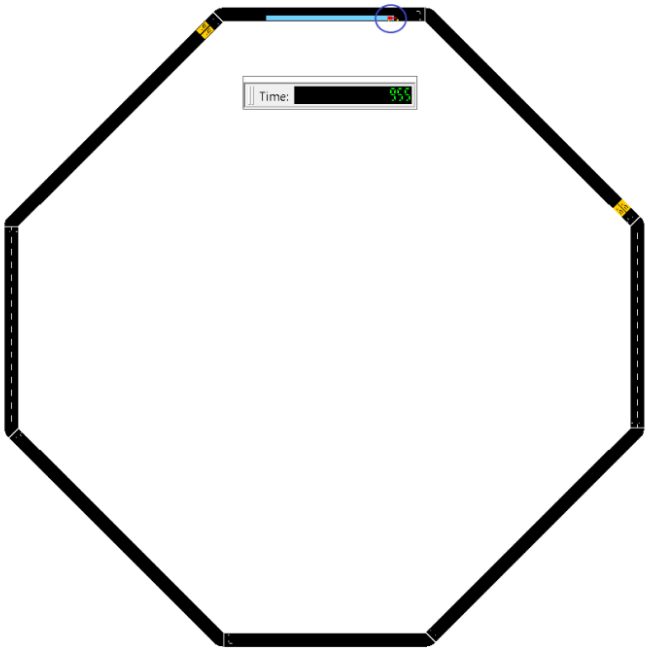

Figura 4.22 Visualización de la simulación en SUMO: Vehículo saliendo de la estación de carga.

En imagen 4.22 el vehículo está saliendo de la estación de carga ya que la simulación se encuentra en el tiempo de simulación 955 por lo que los 400 segundos de carga ya se han agotado.

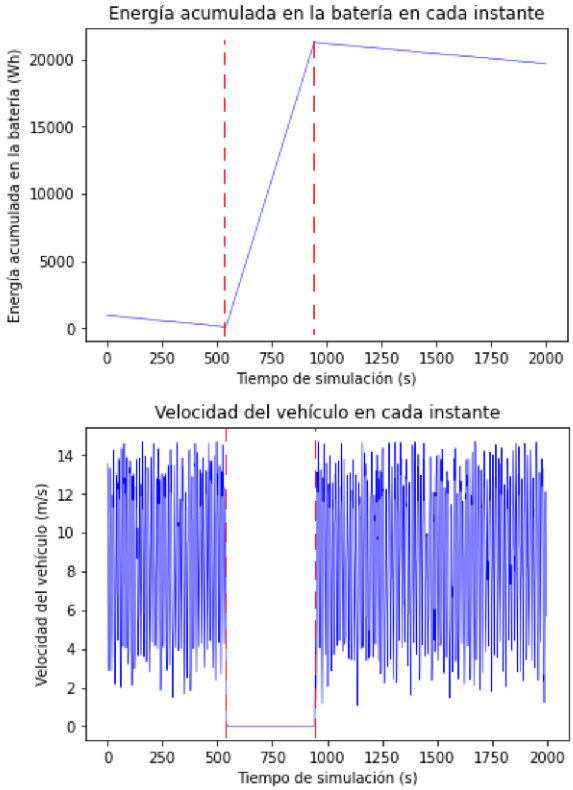

Figura 4.23 Gráfica donde se observa la velocidad y la energía del vehículo en cada instante.

Para ver que realmente funciona la simulación se han generado 2 gráficas 4.23, estas muestran la energía del vehículo en cada instante y la velocidad de este. Al pararse el vehículo en la estación de carga se puede ver como crece la gráfica de la energía de la batería, mientras que la gráfica que representa la velocidad va a pasar a valer 0 ya que el vehículo se encuentra parado mientras se realiza la carga.

Con esta simulación se ha podido comprobar que utilizando python los errores de la función "traci.vehicle.setStop" dejan de aparecer y ya se puede utilizar correctamente esta sentencia, que es de gran importancia al permitir controlar las paradas de los vehículos dentro de las estaciones de carga. Por ello a partir de ahora todas las simulaciones se van a hacer con Python.

Prueba del estacionamiento de un vehículo cerca de una estación de carga

Se va a realizar un script de Python para probar como se puede actuar cuando la estación de carga esté llena y lleguen más vehículos con necesidad de entrar. Para solucionar este problema se ha planteado colocar unas pequeñas zonas virtuales de estacionamientos de vehículos en la cercanía de las estaciones, con el fin de que el vehículo que se encuentre en espera no entorpezca la circulación del resto de vehículos.

Para mostrar el funcionamiento de la idea inicial se va a crear una simulación donde se muestran 2 vehículos en circulación por el escenario octogonal. Al quedarse ambos vehículos sin batería, uno de ellos puede entrar en la estación de carga mientras que el segundo se va a quedar estacionado antes de llegar a la estación esperando a que la estación de carga se quede libre. Antes de comenzar con la explicación del control realizado con python se van a crear los fichero necesarios para crear el entorno de simulación.

A continuación, se van a mostrar los ficheros que se han utilizado en la simulación:

- **Escenario (cricles.net.xml):** El escenario de simulación se corresponde con el escenario octogonal utilizado en ejemplos anteriores.
- **Fichero adicional (cricles2.add.xml):** Este fichero permite añadir una estación de carga y crear los rerouter, como se va a hacer la prueba para una sola estación el fichero utilizado será el mismo que se ha usado en la simulación 4.1.6.
- **Vehículos y rutas (circlesveh2.rou.xml):** En este ejemplo se van a usar un flujo de vehículo con 2 vehículos, por lo que el fichero va a presentar ligeras diferencias en relación a los utilizados anteriormente. La idea de utilizar dos vehículos se debe a que se quiere mostrar el comportamiento de los vehículos cuando la estación de carga esté llena, en este caso se va a suponer que la estación solo tiene capacidad para un vehículo.
- **Configuración (circlesveh2.sumocfg):** El fichero de configuración tendrá la estructura de siempre incluyendo los fichero de configuración de las rutas y de la red, en este caso se tiene que incluir el nuevo nombre del fichero *.rou* que se ha creado.

Como ya se ha comentado, el objetivo principal de esta simulación es crear una zona de estacionamiento cercano a la estación de carga, donde los vehículo se puedan parar a la espera de que la estación de carga se quede libre.

Para realizar esta parada previa a la parada de carga se va a utilizar la función "traci.vehicle.serStop" explicada en el apartado 4.1.5. Al usar esta sentencia hay que indicar para qué se realiza la parada, cómo se usa para estacionar hay que cambiar el valor de la variable "flags" por 1 que es el valor que indica que la parada es un estacionamiento, además se tiene que poner en la variable "edgeID" el identificador del borde de la carretera donde se quiere estacionar el vehículo. A continuación, se muestra cómo se realiza la llamada a dicha sentencia para los dos tipos de paradas utilizados:

- **Parada de carga:**

```
traci.vehicle.setStop(vehicleIDs[vehpar1],'station1',20,0,
dur_s[vehpar1],32,60,0)
```

En la anterior línea de código se puede ver como la duración introducida es una variable que depende del vehículo que se para, esta variable que indica el tiempo que debe estar el vehículo parado en la estación de carga se estima antes de realizar la llamada. Dicha variable depende de la energía que tiene el vehículo al entrar en la estación, de la potencia de carga y rendimiento de la estación y de la capacidad total de la batería.

```
eng_actB = eng_act[vehpar1,i]
 #tiempo que se tarda en cargar completamente la batería
dur_s[vehpar1] = round(((eng_max[vehpar1]-eng_actB)/pot_estac*0.95)
   *3600)
```

Sabiendo el tiempo de carga que necesita cada vehículo y el tiempo de simulación en el que el vehículo entra en la estación de carga, se consigue saber para qué tiempo de simulación se queda libre la estación de carga. Por ello se crea y se le da valor a la siguiente variable:

```
timeStaVeh[vehpar1]=i+dur_s[vehpar1]
```

La sentencia de parada se tiene que activar cuando el vehículo se encuentre en la zona de la estación de carga por lo que hay que acotar la llamada a esta función por medio de un "if".

```
 if 20 >= vehPos[j,0] < 65 and 200 < vehPos[j,1] < 300:
```

- **Parada de estacionamiento:**

```
traci.vehicle.setStop(vehicleIDs[vehpar1est],'E85',10,0,
dur_sesta[vehpar1est],1,0,0)
```

En este caso también se puede ver como la duración que se indica al realizar el estacionamiento es una variable llamada "dur_sesta" y se define para cada vehículo. Esta se estima sabiendo en que instante de tiempo el vehículo llega a la posición de estacionamiento para esperar y conociendo el tiempo de simulación en el que la estación de carga se queda libre, variable que se ha calculado anteriormente.

```
dur_sesta[vehpar1est] = float(timeStaVeh[round(vehcarga)])-i+10
```

Al igual que se tiene que acotar la llamada a la función cuando se quiere parar un vehículo en la estación de carga, también se debe hacer cuando se quiera estacionar un vehículo. Debido a esto se va a utilizar un "if" como se ha hecho anteriormente.

```
 if 2.85 <= vehPos[j,0] < 23 and 293 < vehPos[j,1] < 300:
```

Para ver cuando se está quedando sin batería el vehículo y actuar sobre él se va a utilizar la idea del ejemplo 4.1.5 y se va a añadir la activación de la variable "charge" cuando el vehículo sea de color azul. Por lo que se puede ver visualmente cuando el vehículo se debe ir a la estación de carga, lo que va a facilitar mucho la comprobación del código. En el siguiente fragmento de código se pude ver como se ha hecho la implementación:

```
if eng_act[j,i] > eng_act[j,i+1]:
  if eng_act[j,i] >= eng_act[j,1]*2/3:
    traci.vehicle.setColor(vehID,[255,0,0,255])
  if eng_act[j,1]*2/3>eng_act[j,i] and eng_act[j,i]>= eng_act[j,i]*1/3:
    traci.vehicle.setColor(vehID,[255,123,0,255])
  if 1300 > eng_act[j,i] and eng_act[j,i] > 0:
    traci.vehicle.setColor(vehID,[0,0,255,255])
    charge[j,i] = 1
else:
  traci.vehicle.setSpeed(vehID,0)
```

Resultados:

Para ver el funcionamiento del código se van a mostrar una serie de imágenes y gráficas.

A continuación se muestran imágenes de momentos importantes de la simulación:

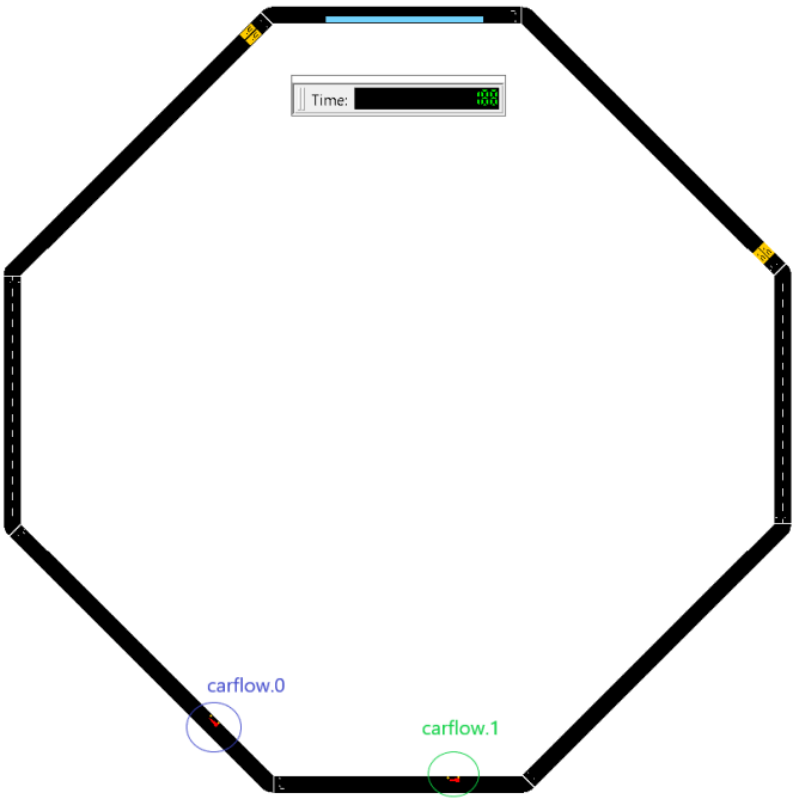

Figura 4.24 Visualización de la simulación en SUMO: Vehículos en movimiento.

En la figura 4.24 se puede ver como ambos vehículos se encuentran circulando de forma normal por el escenario, esto se debe a que los vehículos tienen suficiente batería para seguir circulando.

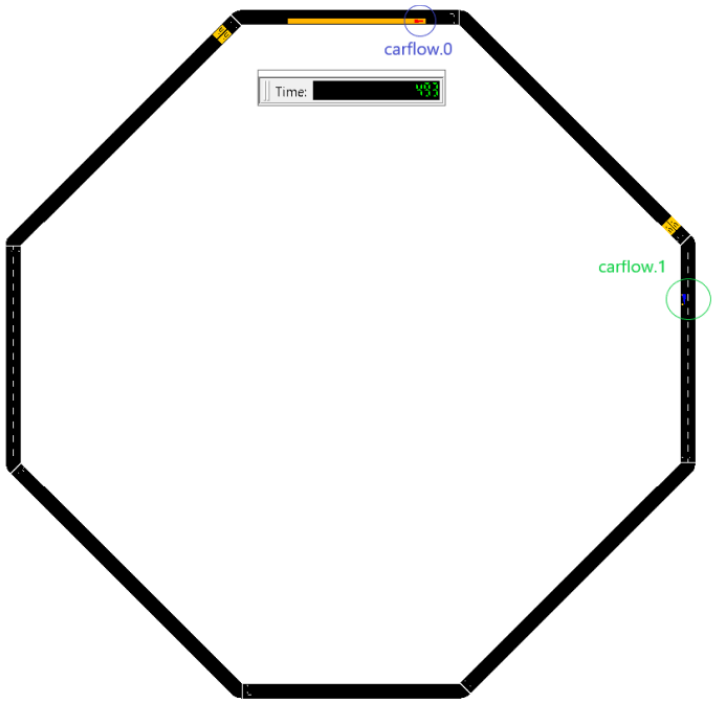

Figura 4.25 Visualización de la simulación en SUMO: Un vehículo cargando y otro en movimiento.

Se observa en la anterior imagen 4.25 que el vehículo que tiene poca carga se ha dirigido a la estación de carga mientras que, el otro sigue moviéndose para conseguir llegar a la zona de estacionamiento.

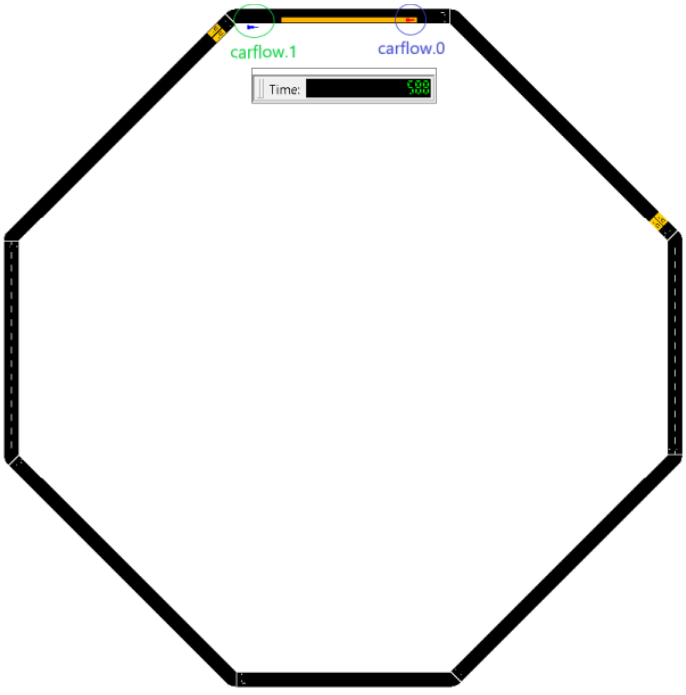

Figura 4.26 Visualización de la simulación en SUMO: Un vehículo cargando y el otro en espera.

Una vez que el vehículo pasa por la zona de estacionamiento este se para a la espera de que la estación de carga se quede libre, este suceso se puede ver en la figura 4.26.

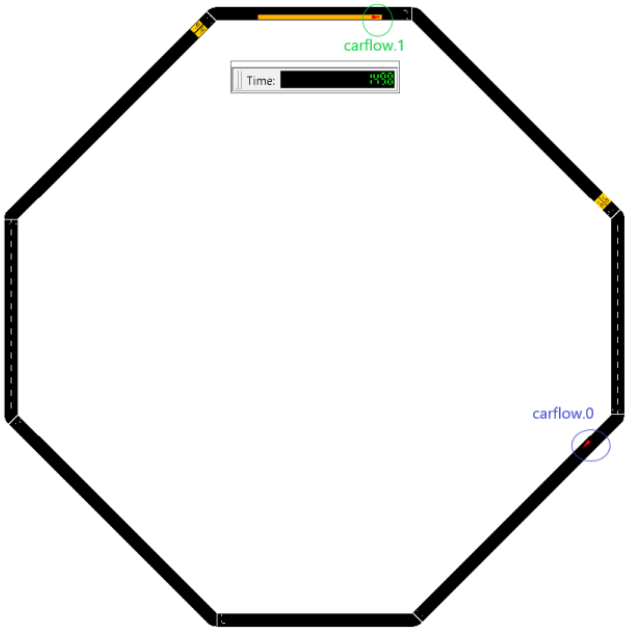

Figura 4.27 Visualización de la simulación en SUMO: Un vehículo en movimiento y otro cargando
.

En esta última imagen se puede ver como el vehículo que estaba estacionado esperando ha salido de la zona de estacionamiento y ha entrado en la zona de carga.

En las siguientes dos gráficas se puede ver la energía acumulada en la batería en cada instante y la velocidad del vehículo, estas gráficas permiten comprobar que el vehículo carflow.0 es el que ha entrado antes en la estación de carga como se muestra en la imagen 4.25. También se puede ver como este vehículo solo se para una vez ya que se puede observar en la gráfica de velocidad un solo cambio a 0.

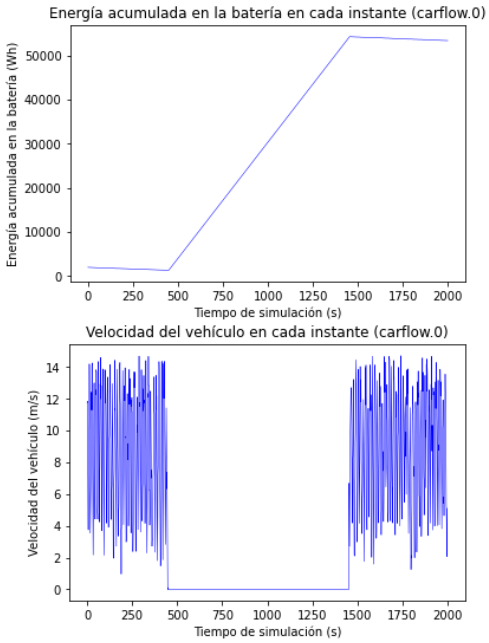

Figura 4.28 Gráfica donde se observa la velocidad y la energía del vehículo carflow.0 en cada instante.

Para terminar con el análisis de resultados de la simulación se muestran dos gráficas para el vehículo carflow.1 al igual que se ha hecho con el anterior vehículo. En la gráfica que representa la energía del vehículo se puede ver como empieza a crecer en un tiempo de simulación mucho mayor que para el vehículo carflow.0, se debe a que este vehículo ha entrado antes en la estación al estar libre, mientras que el vehículo carflow.1 ha tenido que esperar a que la estación se quede libre. También se pude ver que el vehículo carflow.1 es el vehículo que ha realizado la espera gracias a la representación de la velocidad de este, donde se puede observar como esta se hace 0 en dos intervalos de tiempo, por lo que este vehículo se para totalmente en dos ocasiones, una para esperar en el estacionamiento y otra para cargar.

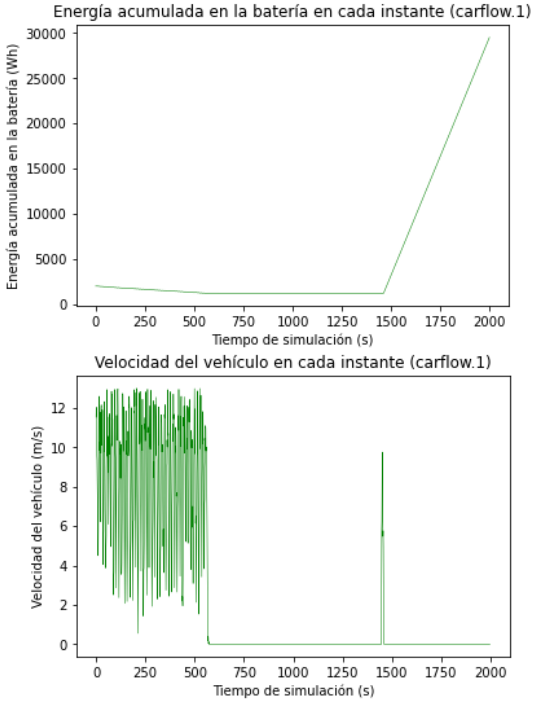

Figura 4.29 Gráfica donde se observa la velocidad y la energía del vehículo carflow.1 en cada instante.

Tras haber realizado varias pruebas de control de vehículos con Python se va a proceder a explicar la simulación final desarrollada, sobre la que se van a obtener los resultados finales del trabajo. Las diferentes pruebas mostradas anteriormente han servido para entender mejor las funciones de parada de SUMO, que son las que se van a utilizar principalmente ya que la simulación final consiste en controlar un gran número de vehículos por un escenario donde existen 8 estaciones de carga. En el siguiente punto se va a explicar el funcionamiento de la simulación final, el escenario que se ha creado para ella y se va a comentar un poco el código de control.

4.2 Simulación final

Con esta simulación se busca:

- Controlar un gran número de vehículos en un escenario cerrado en el que existen 8 estaciones de carga.
- Sacar resultados de la simulación que permitan ver que estación es más eficiente.

- Conocer la posición del vehículo y su estado en cada instante para poder actuar en función de él.
- Se monitorizarán las estaciones para ver cuando están ocupadas completamente.

4.2.1 Escenario (granadasim.net)

Para esta simulación final se va utilizar una red de carreteras un poco más compleja para que las simulaciones se puedan asemejar con la realidad. Por ello se ha tomado una red de carreteras que contiene prioridades en las intersecciones, semáforos y límites de velocidad.

Crear escenarios de simulación manualmente queda fuera de los objetivos de este trabajo, por este motivo, el escenario de simulación se ha obtenido por medio de OSM Web Wisard, aplicación que ya ha sido presentada en el apartado 3.3.2 que permite importar cualquier red de carreteras del mundo al formato deseado ".net".

El escenario que se va a utilizar se puede ver en la siguiente imagen:

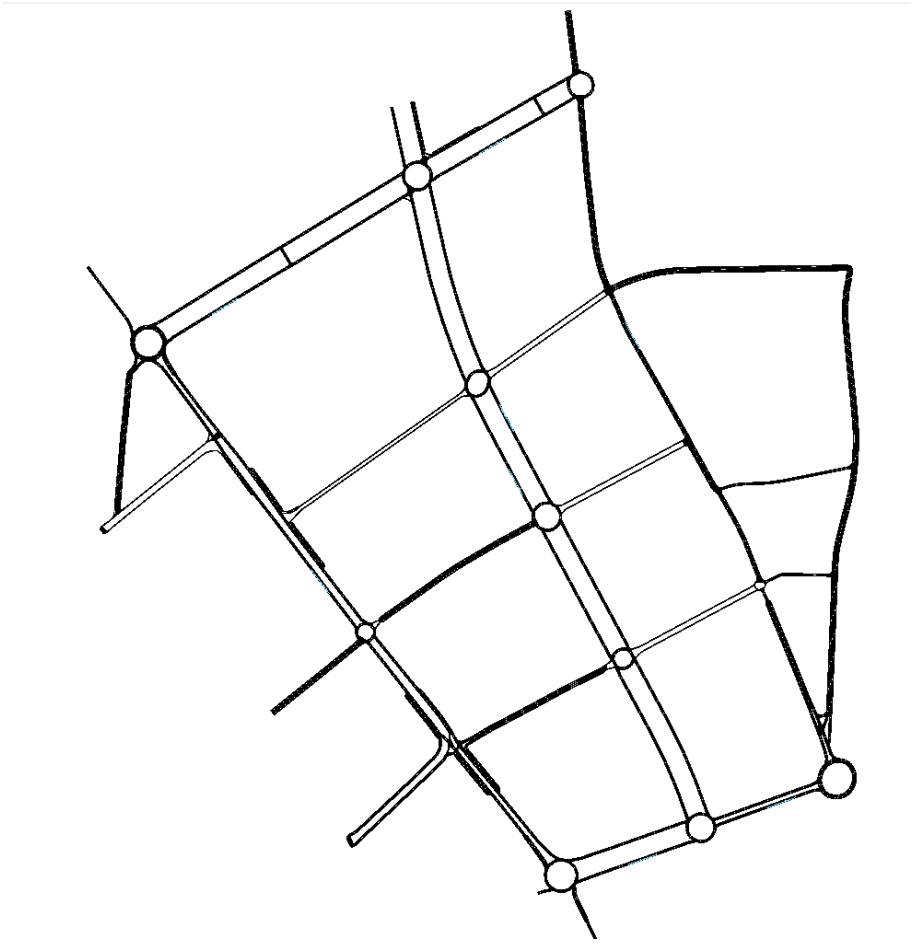

Figura 4.30 Visualización del escenario de la simulación final en NetEdit.

4.2.2 Fichero adicional (granadasim.add)

Este fichero se va a utilizar para añadir 4 redireccionamientos uno para cada ruta, ya que se van a definir 4 rutas diferentes de vehículos, para definir las 8 estaciones de cargas y los 16 detectores de áreas que se van a incluir en cada estación de carga y en su carril paralelo.

Estos detectores se van a añadir para detectar los vehículos que se encuentran en las diferentes áreas de detección con el fin de identificar vehículos que se han parado de forma errónea en las diferentes estaciones de carga.

El como definir una estación de carga y un redireccionamiento ya se ha explicado en el apartado 4.1.2, por ello solo se va a explicar como se define un detector de área.

```
<laneAreaDetector id="station1" lane="22776976#0_0" pos="100" endPos
   ="160" file="detectores.xml" freq="1"/>
```

Tabla 4.12 Parámetros para la definición de un detector de área.

Nombre	Descripción	Tipo	Valor por defecto
id	Identificador detector de área	sting	-
lane	Carril donde ucbicar la estación	string	Id carril valido
pos	Posición inicial de la estación en el carril especificado	float	0
endPos	Posición final de la estación en el carril especificado	float	-
file	Fichero de salida para guardar resultados	sting	-
freq	Frecuencia de muestreo(segundos)	int	1

Una vez definidos los detectores de áreas se pueden visualizar en la interfaz quedando de la siguiente forma:

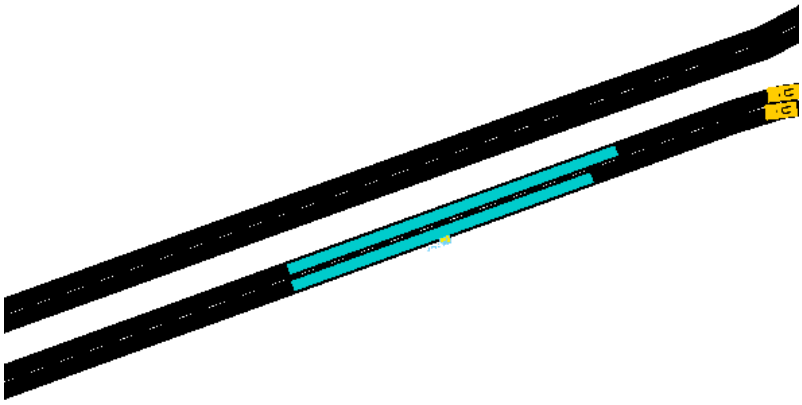

Figura 4.31 Visualización del detector de área.

En esta imagen se pueden observar dos rectángulos de color azul que se corresponden con los dos detectores de áreas que se han añadido por estación de carga. La estación de carga se encuentra tapada por el detector de área de menor longitud.

4.2.3 Propiedades de los vehículos y rutas (granadasim.rou)

En este fichero se van a definir las propiedades de los vehículos que van a aparecer en la simulación. Dicha definición del vehículo se debe hacer de la misma forma que se ha explicado anteriormente. Además de definir los vehículos se definen las 4 rutas diferentes por las que van a circular los vehículos.

No se va a incluir ningún código sobre la definición de las rutas y vehículos que se ha realizado en este fichero ya que se haría de la misma forma que se ha hecho en simulaciones anteriores 4.1.3.

A continuación se va a mostrar una imagen, donde se puede observar sobre el escenario de la simulación los diferentes recorridos que van a realizar los vehículos en función de a que ruta pertenezcan y la posición de las 8 estaciones de carga.

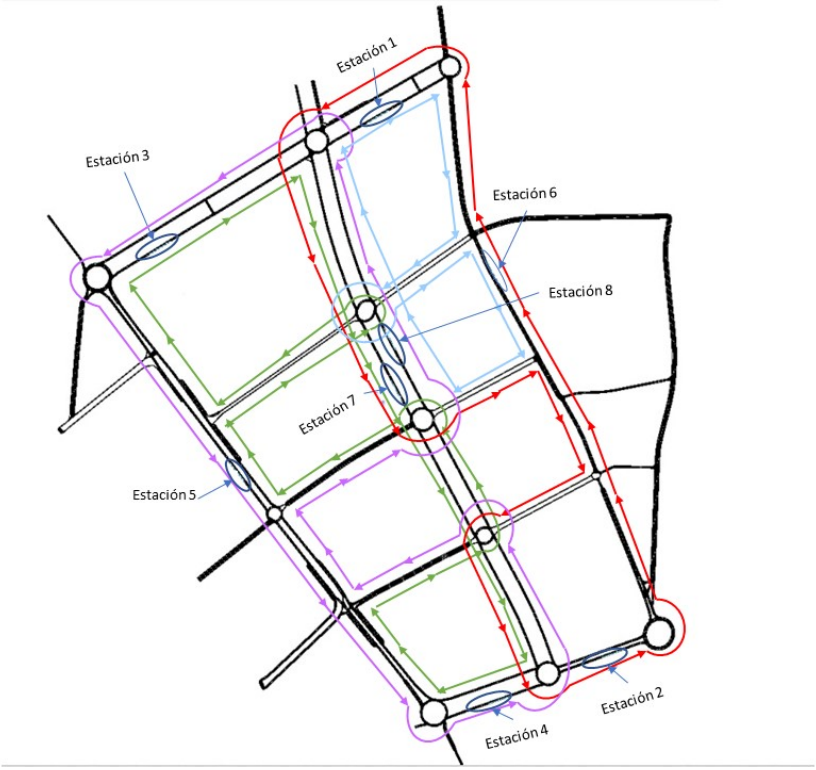

Figura 4.32 Visualización de las rutas y estaciones sobre el escenario.

4.2.4 Fichero de configuración de SUMO (granadasim.sumocfg)

En este fichero de configuración se deben introducir los ficheros necesarios “granadasim.net.xml”, “granadasim.rou.xml” y “granadasim.add.xml”. La explicación y la introducción de otros parámetros necesarios referente a este tipo de fichero se encuentran en el punto 4.1.4.

4.2.5 Control

La idea principal de esta simulación es estudiar el comportamiento de los vehículos eléctricos y ver la eficiencia de las diferentes estaciones de carga. Por ello, se debe realizar un control que permita monitorizar los diferentes vehículos de la simulación y en función de su estado actuar sobre ellos.

- **Control de las rutas y de la estación de parada.**

  La simulación va a constar de 4 rutas de vehículos y 8 estaciones de cargas a las cuales pueden acceder los vehículos en función de la ruta a la que pertenezca. En la figura 4.32 se puede ver a que estaciones de carga pueden acceder cada vehículo en función de su ruta :

  - Vehículos de la ruta de color azul $\longrightarrow$ estación 1 y estación 8.
  - Vehículos de la ruta roja $\longrightarrow$ estación 2, estación 6 y estación 7.
  - Vehículos de la ruta verde $\longrightarrow$ estación 3 y estación 7.

- Vehículos de la ruta morada $\longrightarrow$ estación 4, estación 5 y estación 8.

Además de distinguir las estaciones de carga a las que puede acceder cada vehículo, también se va a tener en cuenta en la simulación que estación de carga es la más cercana al vehículo cuando este se esté quedando sin batería.

Para ello se lee en cada instante la posición en la que está el vehículo y cuando la posición de este coincide con una estación, se le indica al vehículo cual es la próxima estación a esa, esto se hace hasta que se active la variable "charge=1" que indica que el vehículo se está quedando sin batería y debe dirigirse a la estación marcada como la más próxima. También se va teniendo en cuenta el número de vehículos que hay en cada estación y si el número es igual a la capacidad máxima de la estación, esta estación estará cerrada hasta que salga un vehículo de ella.

Antes de ver la estación más próxima se debe ver a que ruta pertenece el vehículo para saber la secuencia de estaciones de carga en las que puede parar, esto se hace detectando el identificador del vehículo y en función de este se conoce a que ruta pertenece.

Se va a mostrar un ejemplo del código que consigue realizar lo explicado, el código representado es el que se utiliza para los vehículos de la ruta azul, para el resto de rutas la idea es la misma.

```
if vehicleIDs[j].count('carflow3')==1: #ruta azul
  if 1229 <= vehPos[j,0] < 1279 and 4043 < vehPos[j,1] < 4078
  and charge[j,i]==0:
      #si ya ha pasado de la estación 1 la siguiente será la estació
         n 8
      num_station[j] = 8
      if veh_st[8] == limvehst:
          num_station[j] = 1
  if 1265 <= vehPos[j,0] < 1298 and 3503 < vehPos[j,1] < 3557
  and charge[j,i]==0:
      num_station[j] = 1
      if veh_st[1] == limvehst:
          num_station[j] = 8
```

- **Control de la energía de cada vehículo.**

  El código encargado de indicar que un vehículo se está quedando sin batería es idéntico al explicado anteriormente en el apartado 4.1.6. Se ha utilizado este código para poder ver durante la simulación cuando un vehículo debe ir a la estación de carga gracias al cambio de color que sufre el vehículo.

- **Definir la energía inicial de cada vehículo al comienzo.**

  Para poder hacer varias simulaciones y que estas sean diferentes se va a crear un código que permita definir la energía inicial del vehículo, esta energía se va a definir para cada uno de forma aleatoria para que se comporten de forma diferente, la energía inicial del vehículo se introduce en el primer paso de simulación.

```
if i==0:
    for j in range(n):
```

```
      vehIDale = vehicleIDs[j]
      #asigar valores aleatorios como energía inicial
      energia_ini=random.uniform(1400,5000)
      energia_iniT[j]=energia_ini
      traci.vehicle.setParameter(vehIDale, 'device.battery.
          actualBatteryCapacity',str(energia_ini))
      eng_act[j,i]=float(traci.vehicle.getParameter(vehIDale,'device
          .battery.actualBatteryCapacity'))
```

Al igual que se pretende hacer simulaciones diferentes también se quiere poder repetir una misma simulación, por ello se deben guardar los datos iniciales de la simulación como el número de vehículos y la energía inicial de cada uno de ellos, estos datos se recogen en un excel al terminar la ejecución del código. Este excel también recoge los resultados de la simulación como los vehículos que se han parado, en que estación se han parado y el tiempo de carga total de cada estación, estos valores van a servir para ver la eficiencia de cada estación y poder sacar conclusiones.

Para poder repetir simulaciones se va a crear un código que permita leer en el primer instante las carga iniciales de cada vehículo almacenadas en un excel. La idea de poder repetir simulaciones es para solucionar posibles errores en el código.

```
if i==0:
  for j in range(n):
    vehIDale = vehicleIDs[j]
    # cargar la energía inicial de los vehículos de un excel
    ruta_archivo = 'C:/Users/Juan Carlos/Desktop/Juan_Carlos/4º/TFG/
        sumo-1_13_0/tools/contributed/traci4matlab/examples/
        granadasimmasCar/simulaciones/nombrefichero.xlsx'
    datos = pd.read_excel(ruta_archivo, sheet_name='Valores
        simulación')
    energia_iniT1=datos.iloc[:,0]
    traci.vehicle.setParameter(vehIDale, 'device.battery.
        actualBatteryCapacity',str(datos.iloc[j,0]))
    eng_act[j,i]=float(traci.vehicle.getParameter(vehIDale,'device.
        battery.actualBatteryCapacity'))
```

- **Control de la parada en un estación de carga.**

  Para controlar la carga de los vehículos se va a realizar un código encargado de indicar el tiempo de carga necesario para cada vehículo y el número de espacios libres en la estación. Este código está particularizado para cada estación ya que este control se realiza de forma independiente para cada estación de carga, por ello a continuación solo se va a mostrar el código utilizado para el control de la estación de carga 1. Para el resto de estaciones de carga el código va a tener la misma estructura y el principio de funcionamiento es el mismo.

```
if veh_st[1] <= limvehst-1:
  if 1220 <= vehPos[j,0] < 1282 and 4040 < vehPos[j,1] < 4075 and
      parado[j]==0 and num_station[j]==1:
    vehpar1 = j
    if charge[vehpar1,i]==1 and num_station[vehpar1]==1:
```

```
      carril_actual = traci.vehicle.getLaneID(vehicleIDs[vehpar1])
      if carril_actual ==’22776976#0_0’ or carril_actual
         ==’22776976#0_1’:
        print("Coche",vehpar1,"parado estación 1")
        #se calcula el tiempo que se tarda en cargar completamente
           la batería
        veh_st[1]=veh_st[1]+1
        eng_actB = eng_act[vehpar1,i] dur_s[vehpar1]=round(((eng_max
           [vehpar1]-eng_actB)/(pot_estac[0]*0.95))*3600) traci.
           vehicle.setStop(vehicleIDs[vehpar1],’station1’,20,0,dur_
           s[vehpar1],32,80,0)
        #ya[vehpar1]==1
        acumnum_station[vehpar1,int(cuentasta[vehpar1])]=1
        acumnum_timestation[vehpar1,int(cuentasta[vehpar1])]=i
        acumnum_durstation[vehpar1,int(cuentasta[vehpar1])]=dur_s[
           vehpar1]
        cuentasta[vehpar1]=cuentasta[vehpar1]+1#nº estaciones en las
            que se ha parado
        colaesta1[t1] = vehpar1#cola de vehículos según el orden de
           entrada dentro de la estación
        t1=t1+1
        if t1==limvehst:
           t1=0
        timeStaVeh[vehpar1]=i+dur_s[vehpar1]
        charge[vehpar1,i] = False
        parado[vehpar1] = True
        print(i,":El vehículo",j,"con",eng_act[j,i],"Wh de batería
           .","Se va a cargar en la estación 1 durante",dur_s[
           vehpar1],"segundos")
```

En este código se puede ver como para realizar una parada en la estación de carga el vehículo debe estar cerca de la estación de carga, por ello se utiliza un condicional que se activa si la posición del vehículo coincide con la posición de la estación de carga. El funcionamiento de la orden de parada de los vehículos ya se ha explicado con mayor claridad en el apartado 4.1.5.

Con este código se obtiene una cuenta de los vehículos que se cargan en dicha estación y el tiempo de la carga, datos útiles a la hora de analizar resultados.

Junto a este se encuentra el código encargado de controlar la parada de espera de los vehículos antes de pararse en la estación de carga.

- **Control de la parada de espera o estacionamiento.**

  En este punto se va a presentar la estructura principal del código encargado de controlar el estacionamiento de aquellos vehículos que no pueden acceder a la estación de carga al estar completa su capacidad.

```
else: #estacionar el vehículo, espera
  if 1050 <= vehPos[j,0] < 1150 and 3990 < vehPos[j,1] < 4040 and
     parado[j]==0 and estacionado[j]==0 and num_station[j]==1:
    vehpar1est = j
    if charge[vehpar1est,i] and num_station[vehpar1est]==1:
```

```
print("Coche",vehpar1est,"va a estacionar")
#se ha introducido lo siguiente para solucionar los siguientes
    tipos de errores:
#TraCIException: stop for vehicle 'carflow1.19' on lane
   '179023414#0_0' is not downstream the current route.
carril_actual = traci.vehicle.getLaneID(vehicleIDs[vehpar1est
   ])
if carril_actual =='22776976#0_1':
  traci.vehicle.changeLane(vehicleIDs[vehpar1est], 0,5)
if carril_actual =='22776976#0_0':
  estacionado[vehpar1est] = True #float(timeStaVeh[round(
      colaesta1[p1])])-i(le quito el tiempo de simulación
      actual)+10(tiempode margen) consigo saber cuando se
      quedará libre un hueco en la estación
  dur_sesta[vehpar1est] = float(timeStaVeh[round(colaesta1[p
      1])])-i+10
  traci.vehicle.setStop(vehicleIDs[vehpar1est
      ],'22776976#0',30+p1*6,0,dur_sesta[vehpar1est],1,20,0)
  acumnum_parado[vehpar1est,int(cuentapara[vehpar1est])]=1
  acumnum_timeparado[vehpar1est,int(cuentapara[vehpar1est])]=i
  acumnum_durparado[vehpar1est,int(cuentapara[vehpar1est])]=
      dur_sesta[vehpar1est]
  cuentapara[vehpar1est]=cuentapara[vehpar1est]+1#nº
      estaciones en las que se ha parado ha esperar
  p1=p1+1
  if p1==len(colaesta1):
      p1=0
  timeStaVehesta[vehpar1est]=i+dur_sesta[vehpar1est]
```

La función encargada de mandarle la orden de estacionamiento al vehículo es la misma que se utiliza para parar el vehículo en la estación de carga, para ejecutar dicha función el vehículo debe estar cerca de la zona habilitada para realizar el estacionamiento. El funcionamiento de la orden de estacionamiento se ha explicado en el punto 4.1.5.

Al igual que el código de parada en la estación de carga, este código también recoge datos útiles a la hora de analizar los resultados como el número de vehículos que se han parado a esperar y el tiempo que estos están parados.

Durante la programación se han encontrado varios avisos de errores que nos mostraba la interfaz de SUMO, estos errores provocaban que la simulación se parase por completo. Tras varias pruebas se han conseguido solucionar los errores, a continuación se muestran las diferentes soluciones que se le ha dado a cada uno de ellos.

- **Errores advertidos por la interfaz de SUMO:** Durante la programación se han dado dos avisos de error distinto, se va a explicar como se ha conseguido solucionar el error y a que se deben.
  - **Downstream.**

    Este error ya se ha explicado en simulaciones anteriores 4.1.5 pero en este caso el error se debe a algo diferente, este aviso se daba en algunas ocasiones cuando se ordenaba la función de estacionamiento del vehículo (traci.vehicle.setStop).

El error se daba cuando el vehículo que se quería estacionar pasaba por la zona de estacionamiento pero por el carril contrario, lo que provocaba que saltase el aviso ya que el carril de parada era el derecho y el vehículo pasaba por el izquierdo. Para solucionarlo se introdujo dentro del código encargado del estacionamiento del vehículo las siguientes líneas:

```
#se ha introducido lo siguiente el siguiente error:
#TraCIException: stop for vehicle 'carflow1.19' on lane
    '179023414#0_0' is not downstream the current route.
carril_actual = traci.vehicle.getLaneID(vehicleIDs[vehpar1est])
if carril_actual =='22776976#0_1':
  traci.vehicle.changeLane(vehicleIDs[vehpar1est], 0,5)
```

Con este código se lee el carril por el que circula el vehículo (traci.vehicle.getLaneID) antes de ordenar el estacionamiento y si el carril por el que circula es el contrario de donde se sitúa la estación de carga, se le ordena al vehículo que se cambie de carril (traci.vehicle.changeLane). Normalmente las estaciones de carga se sitúan en el lado derecho en el sentido de la marcha, el identificador de este carril siempre va a terminar en 0, por ello se comprueba si el vehículo está circulando por el carril cuyo identificador termina en 1 y si es así se cambia de carril al vehículo para que pase a circular por el carril cuyo identificador termina en 0.

– **Too close to brake.**

Este error se daba cuando el vehículo iba demasiado rápido para realizar la parada en la estación de carga, se pensó para solucionarlo en cambiar la velocidad del vehículo en la cercanía de las estaciones. Esto no se puede hacer ya que al cambiar la velocidad del vehículo en un instante, este pasará a tener velocidad constante en toda la simulación y el modelo de velocidad de estos no puede ser constante ya que depende del tráfico y de la carretera por la que esté circulando.

Tras varias pruebas se pensó en cambiar la velocidad máxima permitida en las carreteras donde ocurría dicho error, de esta forma no se cambiaba directamente la velocidad del vehículo y se consigue disminuir la velocidad de este en la cercanía de las estaciones de carga. Para cambiar la velocidad máxima del carril se han introducido las siguientes instrucciones al inicio del programa.

```
# #estación5
traci.lane.setMaxSpeed('826110528#0_0', 13.89)
traci.lane.setMaxSpeed('826110528#0_1', 13.89)
```

Con esto se consigue cambiar la velocidad máxima (traci.lane.setMaxSpeed) de los dos carriles que hay antes de llegar a la estación de carga número 5, solo se cambia porque el error se daba con algunos vehículos que intentaban pararse en la estación 5. La velocidad máxima que se le ha dado es de 13.89 m/s.

Tras explicar los errores que se han encontrado al ejecutar el código con diferentes datos iniciales, también se han encontrado problemas de tráfico en los que la vía se quedaba obstruida al haber un vehículo parado en la estación de carga y otro parado de forma paralela.

- **Saturación de la red debido al tráfico.**

  La obstrucción se debe a que dos vehículos llegan a la vez para cargarse, uno de ellos consigue entrar en la estación y comienza a cargarse mientras que el otro se queda parado de manera

incorrecta al lado del vehículo que se ha parado correctamente. En la siguiente imagen se puede ver lo que se ha comentado:

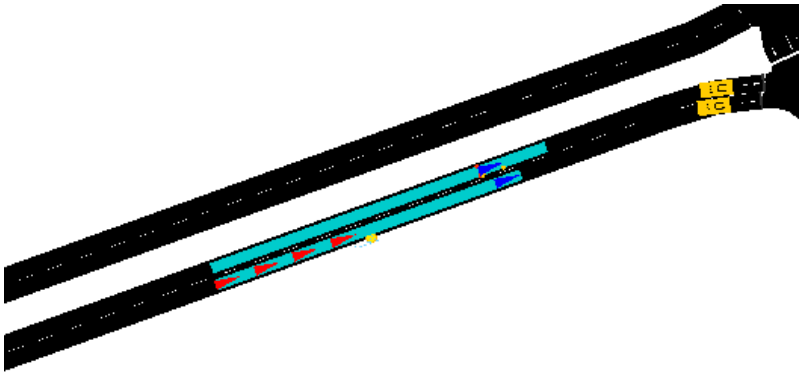

Figura 4.33 Visualización de la obstrucción de la vía.

Para solucionarlo se han utilizado detectores de áreas, que permiten identificar que vehículos han pasado por el área definida y con ello ver si se ha producido una obstrucción. Para ello se usa la función (traci.lanearea.getLastStepVehicleIDs) cuyo argumento de entrada es el identificador del detector de área.

La idea es identificar la obstrucción y hacer que el vehículo mal aparcado se vuelva a poner en movimiento, pero al enviarle la orden este la ignora. Al ver lo que ocurre se decide poner en movimiento el vehículo que está bien aparcado para que el vehículo mal aparcado pueda acceder a la estación de carga y de esta forma se consigue eliminar la obstrucción.

Existen obstrucciones más grandes en las que hay que poner en movimiento más de un vehículo que se encuentre bien aparcado para poder resolver el atasco. En la siguiente imagen se puede ver un ejemplo de ese tipo de atasco.

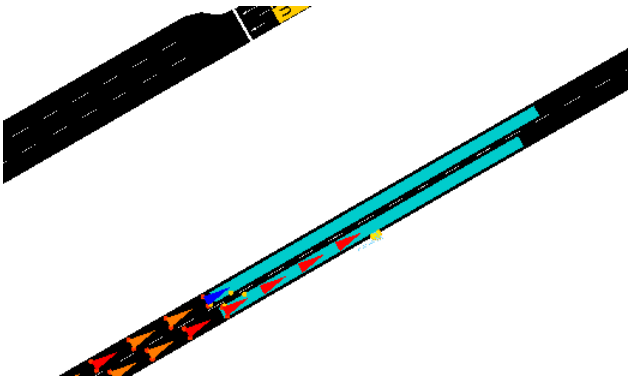

Figura 4.34 Visualización de la obstrucción de la vía.

En estos atascos primero se busca el vehículo más cercano al vehículo que se encuentra mal aparcado, tras encontrarlo se busca cada uno de los vehículos que se encuentran a una distancia determinada del vehículo cercano. Al tener todos los vehículos encontrados se les mandan la orden de ponerse en movimiento para que estos dejen la estación de carga libre y pueda entrar el vehículo que provoca la obstrucción.

5 Resultados

En este capítulo se van a explicar los diferentes resultados obtenidos al ejecutar el programa de la simulación final. En dicha simulación se van a plantear diferentes situaciones iniciales con el fin de poder sacar conclusiones. Para obtener los resultados y poder analizarlos de forma visual se va a crear un excel con Python donde se muestran los datos más importantes de la simulación y con mayor interés.

El objetivo que se busca al realizar diferentes simulaciones, es el de obtener datos suficientes para conocer el comportamiento de los vehículos con el control realizado y de analizar la eficiencia de las estaciones de carga.

Las diferentes casuísticas que se van a analizar se van a obtener del control explicado en el apartado 4.2. Se van a conseguir diferentes situaciones gracias al control que se tiene sobre el número de vehículo, la energía inicial de la batería de estos, la potencia de carga de las diferentes estaciones de carga y el tiempo de simulación. Cambiando estos parámetros de forma independiente se pueden obtener situaciones totalmente diferentes para poder analizar.

A continuación, se van a explicar como se van a analizar los datos de la simulación y las diferentes situaciones que se han creado y su análisis.

5.1 Ficheros de análisis

Para poder obtener datos de las simulaciones se ha creado una función de Python que permite crear un excel con datos útiles para realizar el análisis. Este excel se va a generar al final de cada simulación y va a recoger los datos iniciales de cada simulación y los datos obtenidos al simular.

- **Primera página.**

  En el excel se recoge en la primera página los vehículos que se han parado en cada estación por ciclo, se considera ciclo a cada una de las paradas que han hecho los vehículos para cargarse, si un vehículo se ha parado en una misma simulación dos veces para cargarse de manera completa se representará en el excel dos ciclos. Tras representar un ciclo en cada columna se representa en la ultima columna el total, que es el número total de vehículos que se han cargado por estación. En las últimas columnas aparece el tiempo de carga total de la estación y el tiempo medio de carga por vehículo de cada estación.

Id estación	Veh (ciclo 1)	Veh totales	Tiempo carga total	Tiempo medio carga
1	2	2	2011	1005,5
2	10	10	10070	1007
3	13	13	13178	1013,692308
4	6	6	6031	1005,166667
5	7	7	7056	1008
6	7	7	7064	1009,142857
7	7	7	7053	1007,571429
8	21	21	21145	1006,904762

Figura 5.1 Representación de ejemplo de los datos en el excel.

De la misma forma que se lleva el recuento de vehículos que se paran en cada una de las estaciones también se lleva la cuenta del número de vehículos que se paran en la cercanía de las estaciones de cargas, a la espera de que la estación deje un hueco libre de capacidad.

- **Segunda página.**

  En esta página se pone un resumen general que sirve para depurar el código ya que recoge para cada vehículo el número de estación donde se ha parado a cargar, a esperar y el tiempo que ha durado su carga o espera. Esta página no es de gran importancia para realizar el análisis de las simulaciones.

- **Tercera página.**

  En la primera columna de esta página se guarda la cantidad de energía de las baterías que tienen cada vehículo al principio de la simulación, esto se hace para poder repetir simulaciones idénticas con los mismos datos de partida. Junto a esto aparecen otros datos de partida de la simulación como el número de vehículos que se introducen por ruta, la energía que es capaz de almacenar la batería de un vehículo, la potencia de carga de las estaciones de carga y el tiempo de simulación.

  Finalmente en esta página del excel aparecen los resultados que se van a analizar en cada simulación entre los que se encuentran, los vehículos que podría parar en cada estación como máximo en un caso ideal, una tasa de acceso referido a cada estación de carga, un valor de eficiencia calculado parada cada estación y el tiempo total de cada estación.

Id estación	Veh max	Veh max esperando	Tasa acceso	Eficiencia (veh/tasa acce)	Eficiencia(tiempo/veh)
1	20	13	0,25	8	1005,5
2	20	13	0,25	40	1189
3	20	13	0,25	52	1718,942308
4	20	13	0,25	24	1005,166667
5	20	13	0,25	28	1008
6	20	13	0,25	28	1009,142857
7	40	33	0,5	14	1007,571429
8	40	33	0,5	42	1515,904762

Figura 5.2 Representación de ejemplo de los datos en el excel.

En la segunda columna de la tabla de la imagen, aparece el número de vehículos diferentes que pueden pasar por cada estación como máximo (Veh max). Esto se puede calcular sabiendo el número de vehículos que se introducen por ruta y las estaciones que recorren cada ruta. Las estaciones que recorre cada ruta se sabe a priori ya que se ha definido al configurar la red 4.2.5.

En la tercera columna se puede ver el número máximo que podrían estar esperando a que la estación de carga se quede libre, este dato no será de gran ayuda ya que solo es la diferencia

entre los valores de la segunda columna y la capacidad máxima de cada estación, valor que se considera constante en cada una de las simulaciones.

En la columna siguiente se define un valor llamado tasa de acceso, este es el resultado de una fracción entre el número de vehículos máximo que puede parar en una determinada estación (Veh max) entre el número de vehículos totales que hay en circulación.

$$tas_{acc} = \frac{Veh_{max}}{Veh_{totales}} \tag{5.1}$$

En la quinta columna se muestra un valor que hace referencia a la eficiencia de la estación desde el punto de vista de los vehículos cargados, que se calcula en función del número de vehículos que se paran a cargarse en la estación y la tasa de acceso.

$$Eficiencia = \frac{Veh_{cargados}}{tas_{acc}} \tag{5.2}$$

En la última columna se presenta otro tipo de parámetro que hace referencia al tiempo total que los vehículos se encuentran parados en la estación o cercanía. Este valor se define en función de la suma de todos los tiempo de carga de la estación entre los vehículos que se han cargado en cada estación más la suma del tiempo medio que los vehículos se paran en la cercanía de las estaciones.

$$Tiempo_{total} = \frac{tiempo_{carga}}{Veh_{cargados}} + \frac{tiempo_{parado}}{Veh_{parados}} \tag{5.3}$$

Al sumarle el término del tiempo que los vehículos están parados esperando a que se quede algún hueco libre en la estación, el valor del tiempo total aumenta considerablemente en aquellas estaciones en las que hay algún vehículo esperando y este aumento se refleja de forma negativa ya que cuanto más tiempo estén parados los vehículos en la estación peor será la estación.

Una vez presentado el excel del que se obtienen los datos a analizar se van a proceder a explicar los diferentes análisis realizados. Con esto se busca conocer los límites del control realizado y del escenario que se ha utilizado, también se quiere ver cuál es el comportamiento de cada una de las estaciones de carga y de los vehículos.

5.2 Primer análisis: Número máximo de vehículos en circulación

Se van a realizar múltiples simulaciones en la que se va a aumentar el número de vehículos en circulación de 100 en 100, tras ellos se van a analizar los resultados generados. Con esto se busca hallar el límite de la red en cuanto al número de vehículos máximos que puede soportar, al aumentar los vehículos en circulación se van a presentar en la simulación mayor número de atascos ya que existen mayor número de vehículos que se quedan sin batería antes de llegar a la estación de cagar.

- **Características de la simulación**

  Para agilizar la ejecución del código se va a definir la energía inicial de la batería de cada vehículo como un número aleatorio entre dos valores, este valor aleatorio se va a definir en

el mismo código de Python evitando el uso de excel ya que al usarlo se ralentiza mucho la ejecución del código. Al definir cada vez que se ejecute el código una nueva carga inicial el comportamiento de un mismmo vehículo va a ser diferente para cada simulación.

Además de definir la carga inicial de cada vehículos también se define el límite inferior de carga de la batería para que un vehículo se vaya a la estación de carga que es de 1300 Wh. En la siguiente tabla se representa un resumen de los parámetros que se mantienen constantes en las simulaciones realizadas para este análisis.

Tabla 5.1 Resumen: Parámetro constantes.

Tiempo simulación (s)	Capacidad batería veh (Wh)
3000	60000
Potencia estación (W)	Capacidad estación (nº veh)
200000	7
Batería inicial (Wh)	Energía mínima para cargar (Wh)
1400−5000	1300

- **Resultados de la simulación**

  Se van a mostrar diferentes valores de cada simulación que van a servir para analizar cada una de las simulaciones, entre los valores que se van mostrar se encuentran el número de vehículos que se quedan sin batería y el número de atascos. Con estos datos se puede ver cuantos atascos se han producido, ya que cada vehículo que se quede sin batería va a causar un atasco al quedarse parado sobre la vía, que los vehículos se queden sin batería se debe a que las estaciones al existir muchos vehículos en circulación se encuentren saturadas.

Tabla 5.2 Resumen: Parámetro constantes.

Nº veh circulando	100 veh	200 veh	300 veh	400 veh	500 veh
Nº veh sin baterría	0	5	5	4	3

Como se puede ver en la tabla en todas las simulaciones excepto la de 100 vehículos existen vehículos que se quedan sin batería por lo que las simulaciones de más de 100 vehículos tendrán muchos más atascos. Al tener más atascos en la simulación los vehículos se quedarán sin entrar en la estación y los resultados de la simulaciones serán menos exactos.

Por lo comentado anteriormente se decide realizar simulaciones donde el número de vehículos que estén circulando este entorno a 100 vehículos, por ello en el siguiente análisis se van a realizar simulaciones entre 40 y 180 vehículos.

5.3 Segundo análisis: Estación más usada y más lenta (1)

Se van a realizar un conjunto de simulaciones con las que se busca conocer cuál es la estación de carga más utilizada por los vehículos y cuál es la estación de caga más lenta.

Para obtener estos resultados sobre el escenario se van a realizar varias simulaciones en la que se dejaran constantes ciertos parámetros y solo se van a cambiar el número de vehículos que se introducen por ruta.

– **Características de la simulación**

La carga inicial de la batería de cada vehículo va a ser un número aleatorio entre 1400 y 5000, esto se hace para que cada vehículo tenga una carga inicial distinta y que su comportamiento también lo sea. Este valor aleatorio se va a generar en un excel con el fin de que las cargas iniciales no cambien de una simulación a otra. Por lo que si se ejecuta una simulación de 40 vehículos y luego una de 80 vehículos los 40 primeros vehículos de la segunda simulación tendrá la misma carga inicial que en la primera simulación de 40 vehículos.

Para las diferentes simulaciones que se van a ejecutar los vehículos se deben dirigir a la estación de carga cuando la energía de la batería sea inferior a 1300 Wh, este valor se pone para los vehículos tengan la batería necesaria para llegar a la estación de carga asignada.
Otros parámetros que se mantienen constantes durante toda la simulación y son de interés se van a representar en la siguiente tabla.

Tabla 5.3 Resumen: Parámetro constantes.

Tiempo simulación (s)	Capacidad batería veh (Wh)
3000	60000
Potencia estación (W)	Capacidad estación (nº veh)
200000	7
Batería inicial (Wh)	Energía mínima para cargar (Wh)
1400−5000	1300

Por tanto todas las simulaciones tendrán los mismos parámetros anteriores, solo se va a cambiar el número de vehículos por ruta que se van a aumentar de 5 vehículos en 5 vehículos para cada ruta. Para este primer análisis se han realizado 8 simulaciones en las que se ha ido aumentando el número desde 40 vehículos hasta 180 vehículos.
Se ha simulado hasta 180 vehículos porque se ha observado que al aumentar el número de vehículos más allá de los 180, en las simulaciones cada vez se producían más atascos y los resultados seguían comportándose del mismo modo, por lo que no aportaba grandes cambios al objetivo de este primer análisis. A esto se le añade que las simulaciones cada vez eran más pesadas y por lo tanto más lentas ya que el proceso de coger los datos de carga inicial de un excel ralentizaba demasiado la ejecución.

– **Resultados de la simulación**

A continuación, se van a mostrar los resultados obtenidos al realiza varias simulaciones, en las simulaciones como ya se ha indicado solo se ha modificado el número de vehículos por ruta. El objetivo de esta simulación es ver que estación es la más usada y el tiempo medio de carga y espera para un solo primer ciclo de carga.

En la siguiente gráfica se representan la eficiencia para cada estación y para cada simulación que se ha realizado, para distinguir que línea pertenece a una simulación u otra solo hay que ver la leyenda que hay junto a la gráfica.

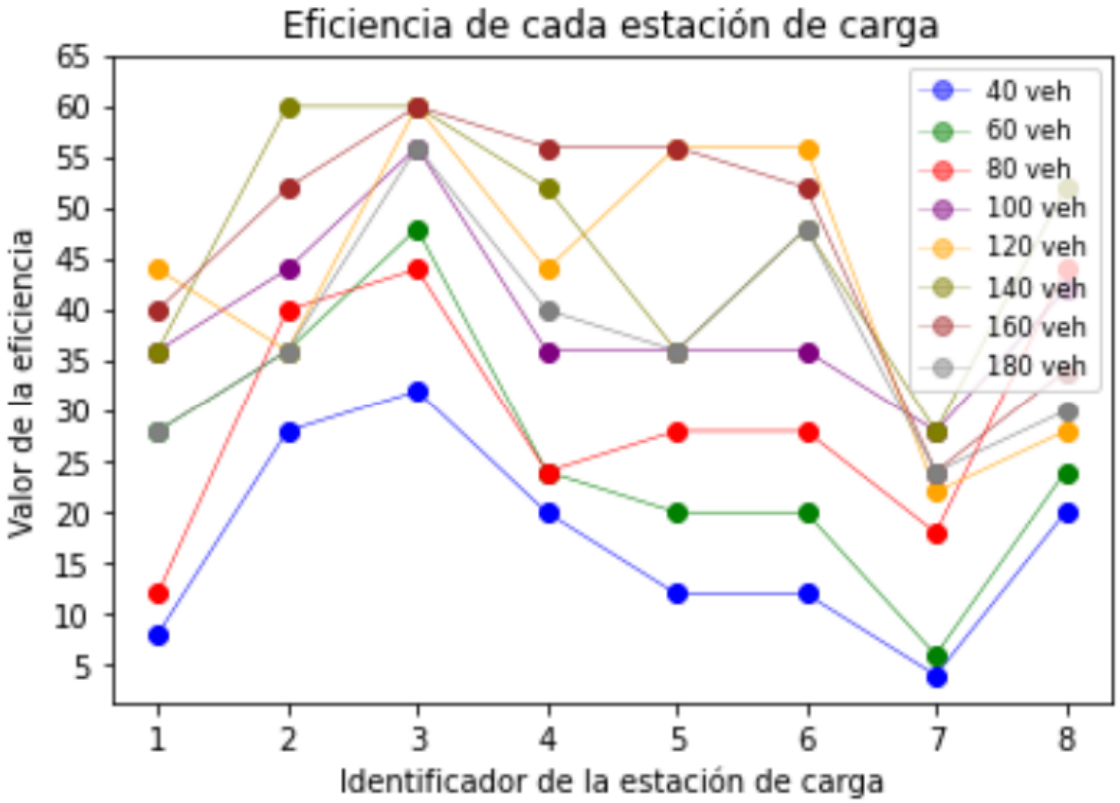

Figura 5.3 Gráfica que representa la eficiencia de cada estación de carga para cada simulación.

En esta gráfica se representa los valores de eficiencia 5.2 termino ya explicado. Cuanto mayor sea la eficiencia de la estación, mayor la relación entre los vehículos que se cargan en la estación y los vehículos que pasan por ella, en este caso para cada simulación se ha representado en la gráfica una línea de un color determinado.

Si se observa la gráfica 5.3 la estación de carga que suele ser más eficiente es la estación 3 al ser en todas las simulaciones la que tiene mayor valor de eficiencia, esto se puede deber a que el alcance de esta es mayor ya que todas las estaciones en estas simulaciones tienen la misma capacidad y misma potencia. El alcance se puede ver de manera visual al ver donde están situadas cada estación de carga en el escenario de la simulación 4.32.

Si se observa la figura referenciada se puede ver como el alcance mayor es el de la estación 3 ya que el alcance se puede definir como los kilómetros de vías que dan acceso a la estación, en este caso a todas las estaciones se puede acceder solo por una vía principal por lo que el alcance es mayor en aquella estación donde la vía de acceso tenga mayor longitud. Esto se puede ver visualmente como se ha dicho anteriormente viendo la distancia que hay en las diferentes rutas entre la estación por la que ha pasado el vehículo y la estación próxima por la que pasa. En este caso la estación que está antes y en el mismo sentido de circulación de la estación 3 es la estación 7 y es la mayor distancia que hay entre 2 estaciones de una misma ruta.

Otro aspecto que se puede apreciar en la gráfica es que el valor total de la eficiencia va aumentando de una simulación a otra, esto se puede ver mejor de forma numérica representando dicha eficiencia total en una tabla 5.4. La eficiencia total es la suma de las eficiencias de cada estación para una misma simulación.

Tabla 5.4 Eficiencia total.

Nº veh circulando	40 veh	60 veh	80 veh	100 veh
Eficiencia Total	136	226	238	314
Nº veh circulando	120 veh	140 veh	160 veh	180
Eficiencia Total	346	372	364	298

Este aumento de la eficiencia total se debe a que el número de vehículos en las estaciones de carga también aumenta al haber más vehículos en circulación. Al observar la simulación de 160 vehículos se observa que la eficiencia total deja de aumentar con respecto a la simulación anterior por lo que hay menos vehículos que se han parado a cargarse, por lo que existen vehículos que no han llegado a la estación de carga.
Los vehículos pueden no llegar a la estación de cagar porque se han quedado sin batería antes de llegar o porque se han quedado atascados debido al tráfico. Otra opción sería que el vehículo no llegase al mínimo de batería por falta de tiempo, suceso que no se puede dar si el vehículo está circulando de forman normal ya que la energía inicial dada a cada vehículo debe gastarse antes de que pasen los 3000 segundo de simulación.

Los atascos nombrados anteriormente suelen darse porque existen vehículos que por el tráfico se quedan sin batería antes de llegar a la estación de carga, provocando atascos al quedarse completamente parados en la vía.

En la siguiente gráfica se van a representar los distintos tiempos características de cada estación en cada simulación. Se va a representar el tiempo medio de espera para cada estación y el tiempo total para cada estación, valor que ya se ha definido anteriormente 5.3.

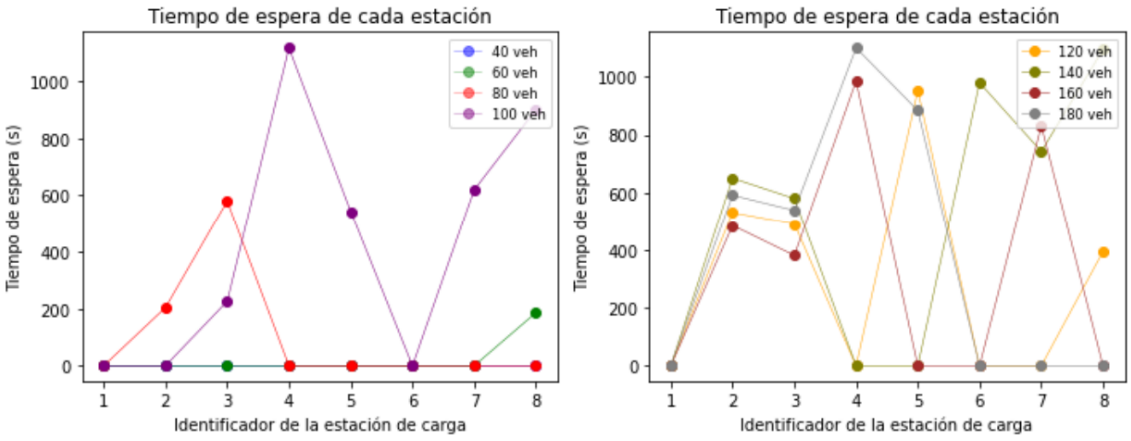

Figura 5.4 Gráfica que representa el tiempo de espera medio de cada estación de carga para cada simulación.

Se puede observar en la gráfica 5.4, que para la mayoría de la estaciones en las diferentes simulaciones el tiempo de espera es 0, esto se debe a que la estación de carga en

determinadas simulaciones no llega a llenarse por completo y los vehículos que llegan pueden ir a cargarse directamente sin parar en la zona de espera.

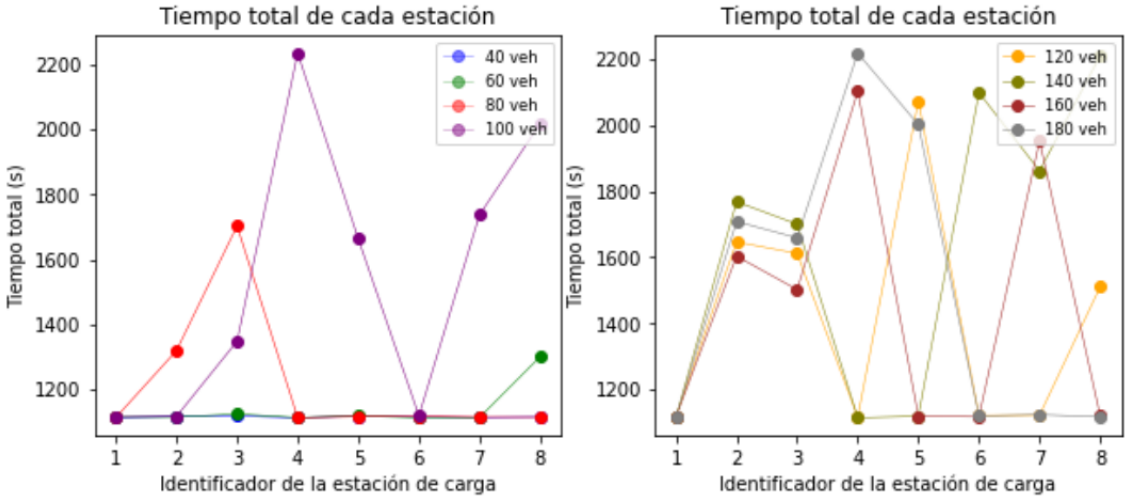

Figura 5.5 Gráfica que representa el tiempo total de cada estación de carga para cada simulación.

Como ya se ha explicado el tiempo total se define como la suma del tiempo medio de carga más el tiempo medio de espera de cada estación. En la gráfica 5.5 representada se puede observar que el tiempo total en la mayoría de las estaciones tiene un valor entorno a 1100 segundos, cuando el tiempo total tiene estos valores se entiende que no existen ninguna parada de espera en la cercanía de la estación por lo que todo el tiempo total representa el tiempo medio de carga de la estación en esos casos.

El tiempo de carga medio de cada estación será muy similar para cada estación y simulación ya que todas las estaciones tienen la misma potencia de carga. La pequeña diferencia que hay entre esos tiempos se debe a que cada vehículo llega con una carga distinta a la estación, esta carga debe ser menor al mínimo de batería establecido que es 1300 Wh para todos los vehículos.

El tiempo mínimo que puede estar un vehículo cargando en la estación hasta conseguir la carga completa se puede calcular analíticamente de la siguiente forma. Los datos necesarios se pueden ver en esta tabla 5.3.

La energía mínima que se va a cargar a un vehículo se da cuando la variable que indica que está el vehículo en mínimos de batería se activa justo al llegar a la estación, por lo que la batería que queda por cargar es la diferencia entre la capacidad máxima de la batería y la energía mínima para cargar.

$$Energia_{min} = Cap_{bateria} - Bateria_{min} = 60000 - 1300 = 58700 \quad \text{Wh} \tag{5.4}$$

Para poder calcular el tiempo que tarda la estación en cargar esa energía mínima se debe conocer la potencia de la estación y la eficiencia de cargar que es 0,95.

$$t_{carga} = \frac{Energia_{min}}{P_{estacin} * Eficiencia_{carga}} = \frac{58700}{200000 * 0.95} = 0.309 = 1112 \quad \text{segundos} \tag{5.5}$$

El tiempo mínimo que va a estar un vehículo parado en la estación de carga es de 1112 segundos, por ello se decía anteriormente que en la mayoría de las simulaciones le

tiempo total para cada estación estaba entorno a 1100 segundos.

Si se observa la gráfica 5.5 se ve que el tiempo total aumenta considerablemente en aquellas estaciones donde hay vehículos parados esperando a que se quede algún hueco libre, la única estación cuyo tiempo total no aumenta debido al tiempo de espera de los vehículos es la estación número 1. Por lo que esta estación será la más eficiente en términos de tiempo ya que los vehículos que paran en esta estación están menos tiempo parados en dicha estación o en su cercanía.

5.4 Tercer análisis: Estación más usada y más lenta (2)

Con este análisis se busca encontrar cuál es la estación más usada y a su vez la más lenta, la idea es la misma que ya se ha presentado en el análisis anterior 5.3. En este caso en lugar de cambiar el número de vehículos que están en circulación para que cada simulación sea diferente, la variable que cambia es la carga inicial de cada vehículo que se va a tomar de manera aleatoria por lo que cada simulación va a ser totalmente diferente.

En este análisis a diferencia del anterior se busca que los vehículos tengan más de un ciclo de carga y descarga, para conseguir esto se ha decidido reducir mucho la capacidad de la batería del vehículo para que la descarga total del vehículo no sea tan lenta. También se ha cambiado el límite de batería mínima que se ha puesto en un 20 % de la capacidad total de la batería.

– **Características de la simulación**

Como ya se ha dicho se va a disminuir de manera considerable la capacidad máxima de los vehículos con el fin de que se den más ciclos de carga y descarga en una sola simulación. Para obtener más valores con los que comparar resultados se ha decidido hacer 8 simulaciones distintas, 4 de ellas con una capacidad máxima de batería de 5000 Wh y las otras con una capacidad máxima de 10000 Wh.

En estas simulaciones se va a definir la carga inicial de cada vehículo como ya se ha dicho por medio de un número aleatorio definido directamente por Python, dependiendo de la máxima capacidad de la batería del vehículo de la simulación se va a definir entre diferentes valores.

Tabla 5.5 Resumen: Parámetro constantes simulación capacidad 10000 Wh.

Tiempo simulación (s)	Capacidad batería veh (Wh)
10000	10000
Potencia estación (W)	Capacidad estación (nº veh)
200000	7
Batería inicial (Wh)	Energía mínima para cargar (Wh)
1400−10000	2000
Vehículos en circulación	
100	

Tabla 5.6 Resumen: Parámetro constantes simulación capacidad 5000 Wh.

Tiempo simulación (s)	Capacidad batería veh (Wh)
10000	5000
Potencia estación (W)	Capacidad estación (nº veh)
200000	7
Batería inicial (Wh)	Energía mínima para cargar (Wh)
1000−5000	1000
Vehículos en circulación	
100	

El rango de la carga inicial se ha tomado mayor con la idea de que los vehículos se queden sin batería en instantes distintos lo que hace que la simulación sea más realista y por ello las estaciones de carga tienen menos probabilidad de que se llenen al completo.

– **Resultados de la simulación**

A continuación, se van a representar los datos obtenidos en las diferentes simulaciones. El objetivo principal de la simulación es ver cual suele ser la estación de carga más usada y el tiempo de de carga y espera para simulaciones donde haya más de un ciclo de carga y descarga.

Para obtener los valores de cada simulación se han hecho muchas simulaciones y se han tomado aquellas donde no se han producido grandes atascos, ya que existen algunas simulaciones en las que aparecen atascos y el recorrido de los vehículos de alguna de las rutas se ve alterado por lo que los resultados obtenidos dejan de ser reales.

Se va a comenzar representando la eficiencia de cada estación de carga para cada una de las simulaciones, diferenciando entre las simulaciones donde la capacidad de la batería del vehículo es de 10000 Wh y de 5000 Wh.

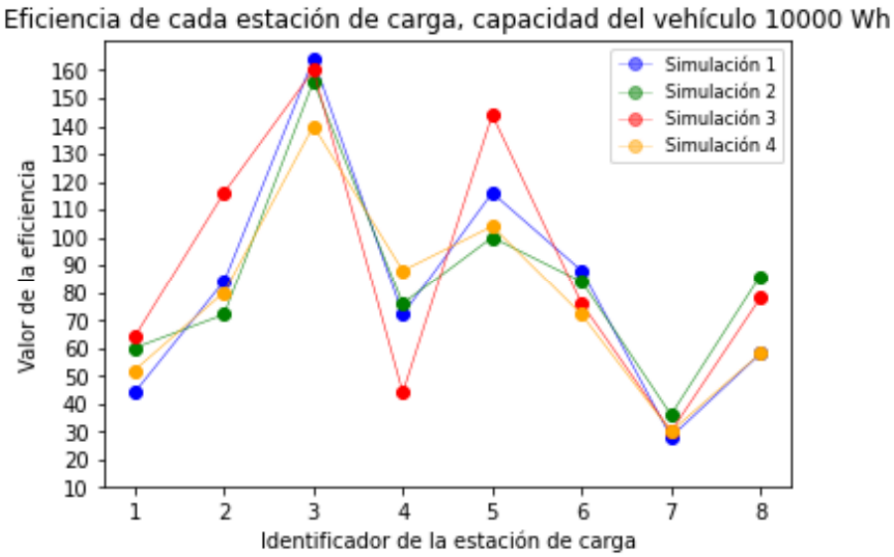

Figura 5.6 Gráfica que representa la eficiencia de cada estación de carga para las simulaciones de 10000 Wh de capacidad.

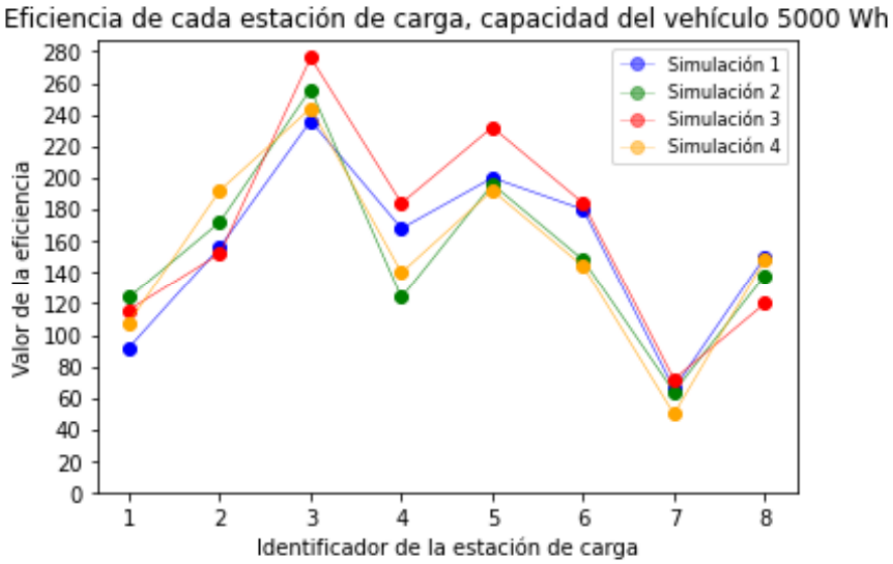

Figura 5.7 Gráfica que representa la eficiencia de cada estación de carga para las simulaciones de 5000 Wh de capacidad.

En la gráfica 5.6 se puede ver como la mayor eficiencia en cada una de las simulaciones se da para la estación 3 al igual que sucede en la gráfica 5.7. Por tanto, con estas simulaciones confirmamos que la estación 3 es la más eficiente del escenario, hecho que ya se ha dado en los análisis de las simulaciones anteriores 5.3.

La segunda estación más usada en la gráfica 5.6 también se puede ver con claridad, ya que el valor de eficiencia es el segundo mayor con una gran diferencia sobre el resto. Por lo que se puede decir con toda seguridad que en condiciones normales de funcionamiento la segunda estación de carga más usada es la número 5. Esto también ocurre para la gráfica 5.7, aunque se puede ver con menos claridad en unas de las cuatros simulaciones, ya que para la simulación 4 la eficiencia tiene el mismo valor tanto para la estación 2 como para la 5. A pesar de esto se puede decir que la segunda estación más usada es la número 5 porque en las otras 3 simulaciones si se puede observar con claridad.

Tabla 5.7 Eficiencia total de cada estación y simulación.

Simulaciones de 10000 Wh de capacidad				
Eficiencia Total	654	670	712	624
Simulaciones de 5000 Wh de capacidad				
Eficiencia Total	1248	1222	1336	1218

La tabla 5.7 representa la suma total de eficiencia por simulación, valor denominado como eficiencia total, este valor no es de gran importancia para este análisis. Se puede ver como la variación de este en las diferentes simulaciones se traduce en que en cada una de ellas el número de vehículos cargados varía.

A continuación se van a representar los tiempos de espera y totales por estación y simulación por separado, dependiendo de si la simulación es para vehículos con capacidad de 10000 Wh o de 5000 Wh. Primero se va a mostrar los resultados de tiempo para la simulación cuya capacidad es 10000 Wh.

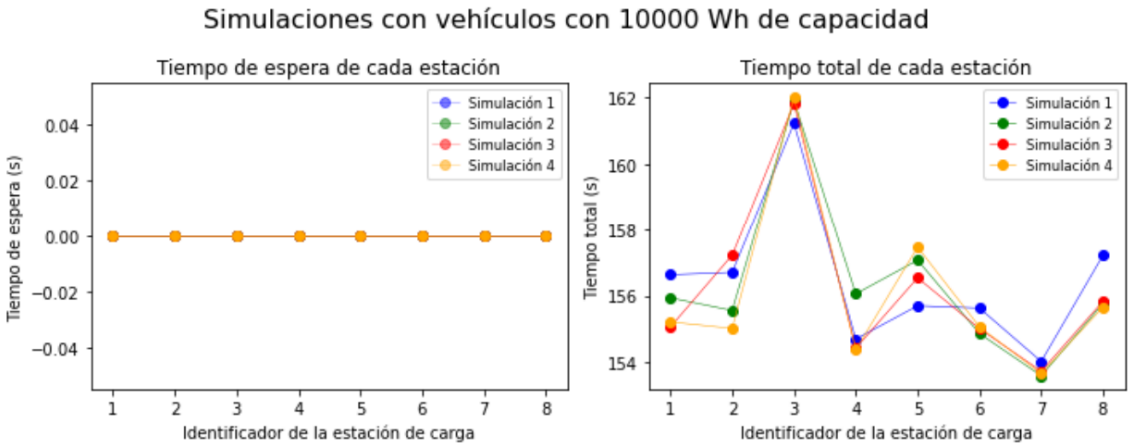

Figura 5.8 Gráfica que representan el tiempo de espera y total de cada estación de carga para las simulaciones de 10000 Wh de capacidad.

El tiempo de carga medio de cada estación va a ser muy similar ya que depende de dos factores como se comentó anteriormente, depende de la carga con la que lleguen los vehículos a la estación de carga y de la potencia de carga de cada estación, en este caso todas las estaciones tienen la misma potencia de carga 200 kW.

Para estas simulaciones donde la capacidad máxima del vehículo es de 10000 Wh y el límite de la batería para cargar es de 2000 Wh se puede calcular el tiempo mínimo de carga que está relacionado con la mínima carga que se le va a realizar a un vehículo, para ello se presenta los cálculos realizados para obtener dicho tiempo.

$$Energia_{min} = Cap_{bateria} - Bateria_{min} = 10000 - 2000 = 8000 \quad \text{Wh} \tag{5.6}$$

Para realizar este cálculo la eficiencia de cargar es 0.95.

$$t_{carga} = \frac{Energia_{min}}{P_{estacion} * Eficiencia_{carga}} = \frac{8000}{200000 * 0.95} = 0.0421 = 151.57 \quad \text{segundos} \tag{5.7}$$

El tiempo mínimo de carga de cada estación es 151.57 segundos, por lo que si se observa la tabla se puede ver que todos los tiempo de carga son mayores a este ya que los vehículos suelen llegar a la estación de carga con una energía inferior a la mínima de 2000 Wh.

También se puede observar en la gráfica 5.8 que todos los tiempos totales son parecidos en todas las simulaciones y estaciones, esto se debe a que en estas simulaciones no hay vehículos parados en la cercanía de la estaciones esperando a que alguna de estas se queden libre, por lo que el tiempo de espera en todas las estaciones es 0. Esto se da porque la potencia de carga de la estación es muy alta en relación a la carga que hay que cargar por vehículo, lo que provoca que la parada del vehículo en la estación de carga sea muy pequeña, por lo que no van a coincidir un gran número de vehículos en una misma estación para que esta se quede completamente llena.

A continuación se van a mostrar los tiempos totales y de espera de cada estación para las simulaciones realizadas donde la capacidad máxima del vehículos es de 5000 Wh.

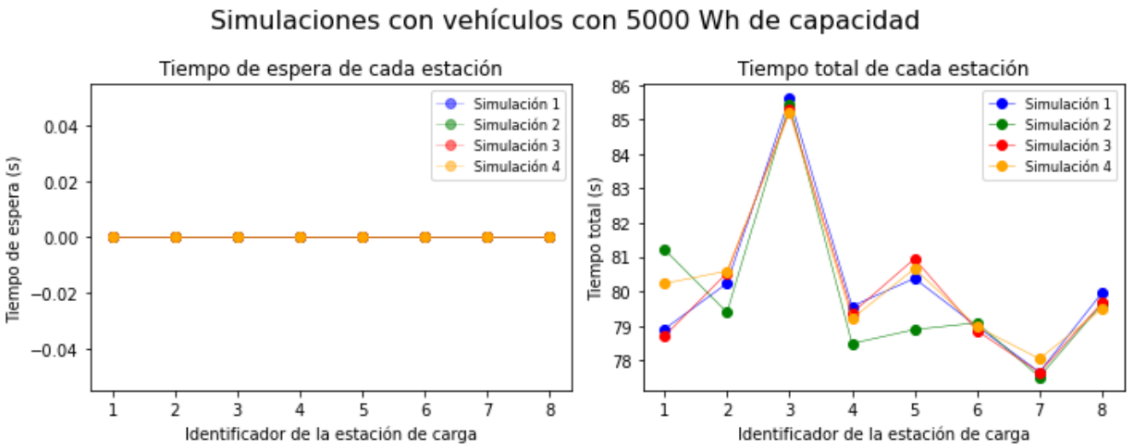

Figura 5.9 Gráfica que representan el tiempo de espera y total de cada estación de carga para las simulaciones de 5000 Wh de capacidad.

Sabiendo que en estas simulaciones los vehículos tienen una capacidad máxima de 5000 Wh y el límite para que estos se dirijan a las estaciones de carga es de 1000 Wh, se puede calcular al igual que se ha hecho anteriormente el tiempo mínimo de carga que necesita cada vehículo. Para hacer este cálculo se debe conocer también la potencia que es 200 kW y la eficiencia de la estación que es de 0.95.

A continuación se muestran los cálculos realizados para obtener el tiempo mínimo de carga.

$$Energia_{min} = Cap_{bateria} - Bateria_{min} = 5000 - 1000 = 4000 \quad \text{Wh} \tag{5.8}$$

$$t_{carga} = \frac{Energia_{min}}{P_{estacion} * Eficiencia_{carga}} = \frac{4000}{200000 * 0.95} = 0.02105 = 75.8 \quad \text{segundos} \tag{5.9}$$

El tiempo de carga mínimo por estación es 75.8 segundos. Se puede ver en la gráfica 5.9 como los tiempos totales son mayores que este tiempo calculado de forma analítica, ya que normalmente los vehículos van a necesitar más tiempo de carga al llegar con una carga menor de 1000 Wh.

El tiempo total representado en las simulaciones de capacidad de 5000 Wh tienen en cuenta también el tiempo de espera de cada vehículo por estación y este es 0 al igual que ocurre en las simulaciones de capacidad de 10000 Wh. El tiempo de espera es nulo debido a que ninguna estación de carga llega a llenarse con 7 vehículos a la vez, esto se debe como se explicó en la anterior simulación a que al ser tan pequeño el tiempo de carga que necesita el vehículo en relación con el tiempo total de simulación no le da tiempo a la estación de carga a llenarse.

5.5 Tercer análisis: Estación más usada y más lenta (3)

Se pretende realizar varias simulaciones en las que se cambie la potencia de carga de la estación con el fin de ver los cambios que se producen con respecto a una simulación de referencia. La idea es ver los cambios que se producen en los resultados obtenidos y ver si sigue siendo la estación de carga más usada la misma que en las simulaciones anteriores.

Para esto se va a tomar como referencia la carga inicial de cada vehículo de una simulación anterior para que el resto de simulaciones sean idénticas, solo se va a cambiar de las simulaciones la potencia de carga de cada estación, el resto de valores serán idénticos a los valores de la simulación de referencia.

– **Características de la simulación**

Las diferentes simulaciones como ya se ha comentado, va a ser idéntica a una simulación de referencia. Como referencia se ha tomado la primera simulación realizada para el análisis anterior, cuyo valores característicos son los siguientes:

Tabla 5.8 Resumen: Parámetro constantes simulación capacidad 10000 Wh.

Tiempo simulación (s)	Capacidad batería veh (Wh)
10000	10000
Capacidad estación (nº veh)	Energía mínima para cargar (Wh)
7	2000
Vehículos en circulación	
100	

En esta tabla no se ha representado la batería inicial ya que en este caso no se va a definir para cada simulación de forma aleatoria, si no que va a ser dato ya que se toma la misma carga inicial de la simulación de referencia para el resto de simulaciones.

Además de no aparecer la carga inicial, tampoco aparece el valor de la potencia de la estación de carga ya que este valor va a ser diferente para cada simulación, al ser esta la variable que se va a ir cambiando para cada simulación. El valor de potencia de cada estación se va a representar junto a los resultados para saber que valor se le ha dado para cada estación y simulación.

– **Resultados**

Se van a representar los resultados obtenidos al realizar varias simulaciones en las que se ha ido variando la potencia de la estación.

El objetivo de estas simulaciones es ver el cambio que se produce en las eficiencias de cada estación y en el tiempo total por estación, cuando se va variando la potencia de carga de cada estación. Para ello se van a realizar 6 simulaciones diferentes entre las que se va cambiar la potencia de carga, se va a hacer una primera simulación donde todas las estaciones tengan la nueva potencia de carga , 4 simulaciones donde se ha decidido cambiar la potencia de cagar por ruta, es decir todas las estaciones que estén contenidas en una ruta van a tener la nueva potencia de carga y una simulación donde la estación 3 que es la de mayor eficiencia se la única que tenga una potencia de carga nueva.

∗ **La nueva potencia de carga es 100 kW**

En esta primera tabla representada se muestran los valores de potencia de carga que va a tomar cada estación para cada simulación. Esto se hace para que en la

gráfica que se va a representar más adelante se pueda relacionar la leyenda con la característica de la simulación.

Tabla 5.9 Características de cada simulación.

	Sim 1	Sim 2	Sim 3	Sim 4
Estación	Potencia (W)	Potencia (W)	Potencia (W)	Potencia (W)
1	200000	100000	200000	200000
2	200000	100000	200000	200000
3	200000	100000	100000	100000
4	200000	100000	200000	200000
5	200000	100000	200000	200000
6	200000	100000	200000	200000
7	200000	100000	100000	200000
8	200000	100000	200000	200000

	Sim 5	Sim 6	Sim 7	
Estación	Potencia (W)	Potencia (W)	Potencia (W)	
1	200000	100000	200000	
2	200000	200000	100000	
3	200000	200000	200000	
4	100000	200000	200000	
5	100000	200000	200000	
6	200000	200000	100000	
7	200000	200000	100000	
8	100000	100000	200000	

A continuación, se va a mostrar una gráfica que representa la eficiencia de cada estación para cada una de las 7 simulaciones que se han realizado. Para entender que diferencias existen entre una simulación y otra hay que ver la tabla 5.11 con la que se relaciona el nombre de la leyenda de la gráfica y las características de la simulación.

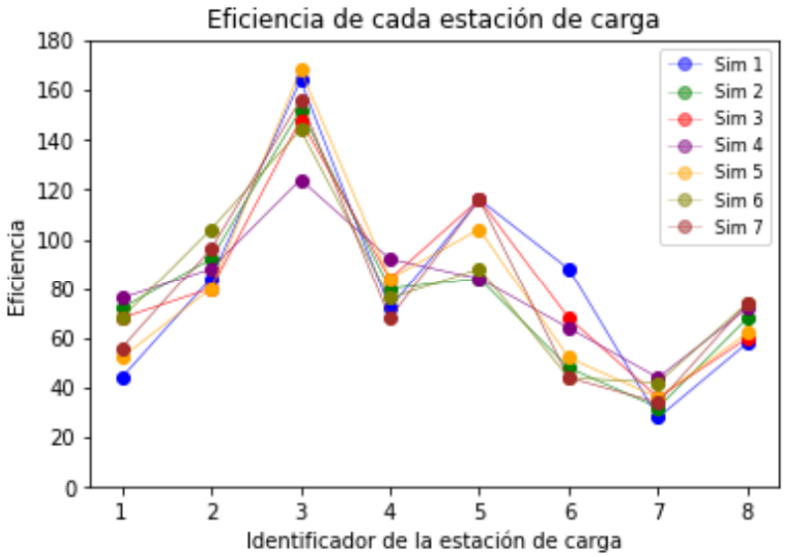

Figura 5.10 Gráfica que representan la eficiencia de cada estación de carga para las simulaciones.

A pesar de haber modificado la potencia de carga de las estaciones, el comportamiento de la red es similar y la estación de carga más usada sigue siendo la estación de carga número 3. La segunda estación de carga más usada varía de un análisis a otro. En el análisis anterior 5.4 la segunda estación de carga más usada era la número 5, mientras que en este análisis la segunda estación de carga más usada en algunas simulaciones es la estación de carga número 2 y en otras la número 5.

Tabla 5.10 Eficiencia total para cada simulación.

	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5	Sim 6	Sim 7
Eficiencia Total	654	628	660	644	638	640	644

En la tabla 5.10 se puede ver la eficiencia total de cada simulación que al ser las simulaciones parecidas no va a variar mucho la eficiencia total de una simulación a otra ya que las cargas iniciales de los vehículos son idénticas de una simulación a otra.

En la siguiente gráfica se muestran los tiempos totales de cada estación y simulación. Se ha divido la representación en dos gráficas distintas para facilitar la visualización.

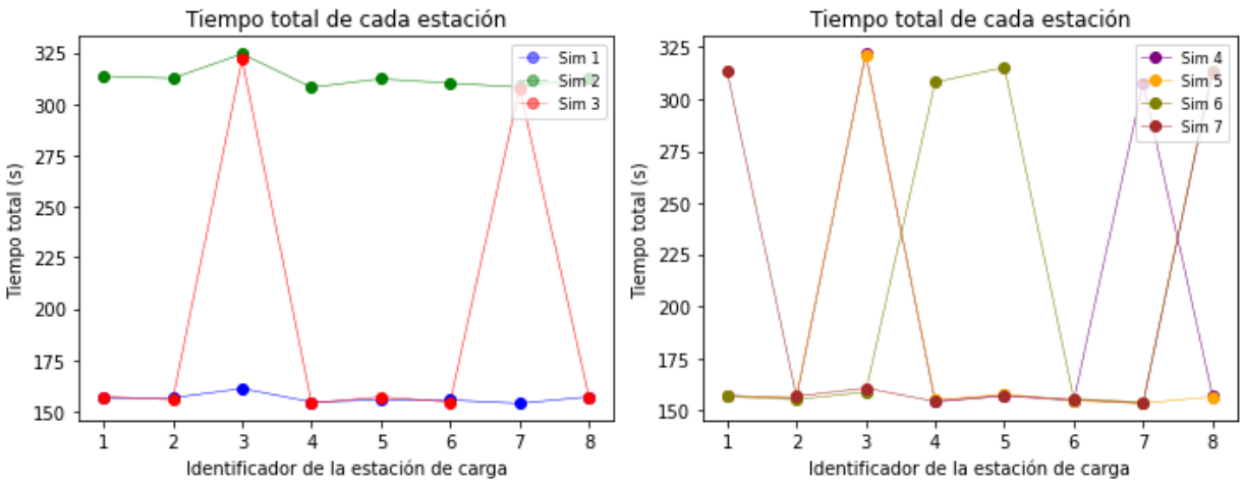

Figura 5.11 Gráficas que representan el tiempo total de cada estación de carga para las simulaciones.

Se puede ver como los tiempos varían entre dos valores distintos, esto se debe a que en algunas simulaciones hay dos potencias de carga diferentes, se van a mostrar los cálculos del tiempo mínimo de carga para cargar la mínima carga necesaria.

Para las estaciones de carga con una potencia de 200 kW y una energía mínima a cargar de 8000 Wh el tiempo de carga mínimo necesario es de 151.57 segundos, valor que se ha calculado analíticamente anteriormente en el punto 5.7. Si se observan las estaciones con dicha potencia se puede ver que el valor del tiempo total está entorno a 155 segundos, por lo que no se aleja mucho del valor de tiempo mínimo que se ha calculado. Esto se debe a que las estaciones en estas simulaciones no se llenan al completo y el tiempo de espera es nulo.

Para las estaciones de carga con una potencia de 100 kW se sabe que al igual que en las de 200 kW la energía mínima que se debe cargar al vehículo es de 8000 Wh,

sabiendo esto y la eficiencia de la estación que siempre es de 0.95 se puede calcular el tiempo mínimo de carga de forma analítica:

$$t_{carga} = \frac{Energia_{min}}{P_{estacion} * Eficiencia_{carga}} = \frac{8000}{100000 * 0.95} = 0.08421 = 303.16 \quad \text{segundos} \tag{5.10}$$

Como se puede ver en la tabla el tiempo total de las estaciones con potencia de 100 kW está entorno 315 segundos, valores ligeramente superiores al tiempo mínimo de carga calculado que es 303.16 segundos. El tiempo total no es mucho mayor que el tiempo mínimo debido a que el tiempo de espera es nulo en las estaciones con potencia de 100 kW como ya ocurría para las estaciones de 200 kW.

∗ **La nueva potencia de carga es 50 kW**

Antes de comenzar representando las gráficas de los resultados se va a representar una tabla con la potencia que va a tomar cada estación en cada simulación, para poder identificar fácilmente las características de cada simulación.

Tabla 5.11 Características de cada simulación.

	Sim 1	Sim 2	Sim 3	Sim 4
Estación	Potencia (W)	Potencia (W)	Potencia (W)	Potencia (W)
1	200000	50000	200000	200000
2	200000	50000	200000	200000
3	200000	50000	50000	50000
4	200000	50000	200000	200000
5	200000	50000	200000	200000
6	200000	50000	200000	200000
7	200000	50000	50000	200000
8	200000	50000	200000	200000

	Sim 5	Sim 6	Sim 7	
Estación	Potencia (W)	Potencia (W)	Potencia (W)	
1	200000	50000	200000	
2	200000	200000	50000	
3	200000	200000	200000	
4	50000	200000	200000	
5	50000	200000	200000	
6	200000	200000	50000	
7	200000	200000	50000	
8	50000	50000	200000	

Se van a representar los resultados obtenidos para cada simulación al variar la potencia de carga entre 200 kW y 50 kW. En la gráfica **??** se va a representar la eficiencia de cada estación para cada simulación diferente.

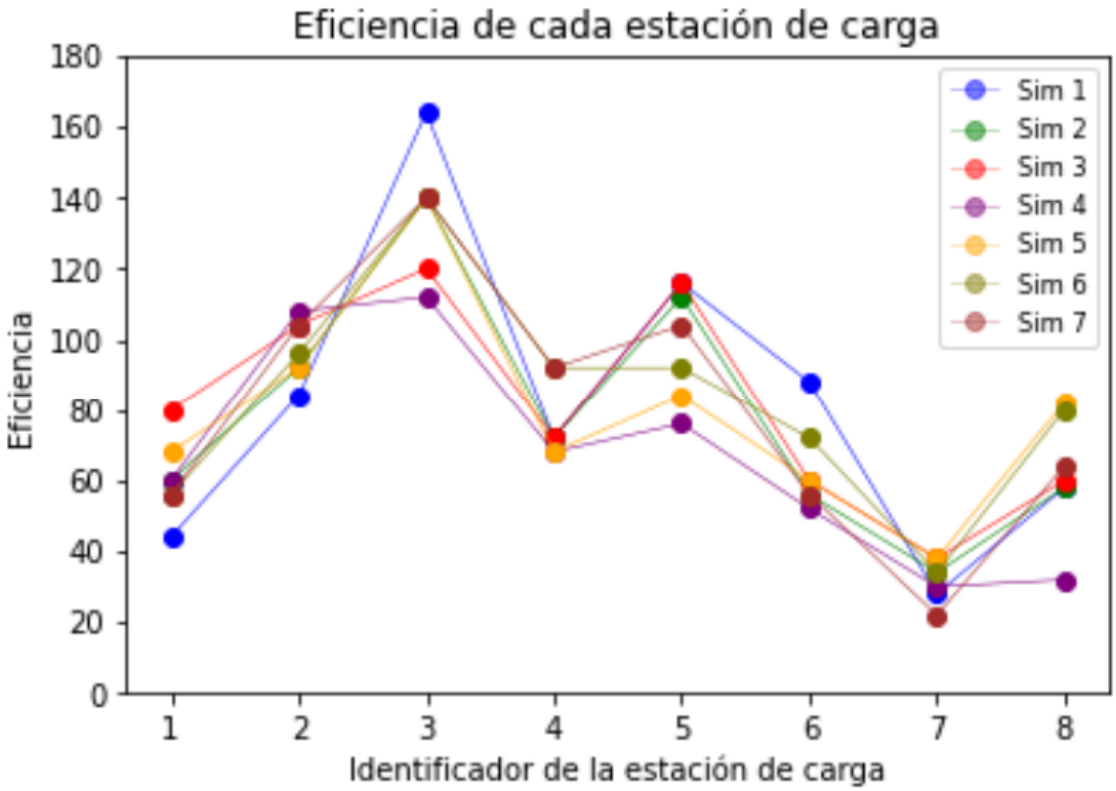

Figura 5.12 Gráficas que representan la eficiencia de cada estación de carga para las simulaciones.

A pesar de variar la potencia de carga de 200 kW a 50 kW la estación más eficiente sigue siendo la estación número 3, pero se observa que cuando hay estaciones de carga con una potencia de 200 kW y en la misma simulación hay estaciones con una potencia de 50 kW entre las que se incluye la estación de carga número 3, la eficiencia de esta disminuye.

También se puede ver que la segunda estación de carga más eficiente al igual que en las simulaciones anteriores 5.5, es en algunos casos la estación de carga número 2 y en otros casos la estación de carga número 5.

Tabla 5.12 Eficiencia total para cada simulación.

	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5	Sim 6	Sim 7
Eficiencia Total	654	624	650	538	632	662	638

Si echamos un vistazo a la tabla 5.12, la eficiencia total representada debe ser muy parecida en todas las simulaciones ya que las simulaciones son idénticas en términos de carga inicial y de número de vehículos en circulación. Por ello en la mayoría de los resultados representados para cada simulación la eficiencia total está en torno a 630, excepto para una de las simulaciones cuyo valor está en torno a 530, esta disminución considerable de la eficiencia total se debe a que en la simulación puede que haya existido un gran atasco, lo que provoca que muchos vehículos se queden parados y no se les gaste la batería, por lo que hay vehículos

que en otras simulaciones debían ir a cargarse y en esta no se dirigen a cargarse porque se quedan parados debido a un atasco.

A continuación se van a mostrar una gráfica con los tiempo total de cada estación de carga para cada simulación:

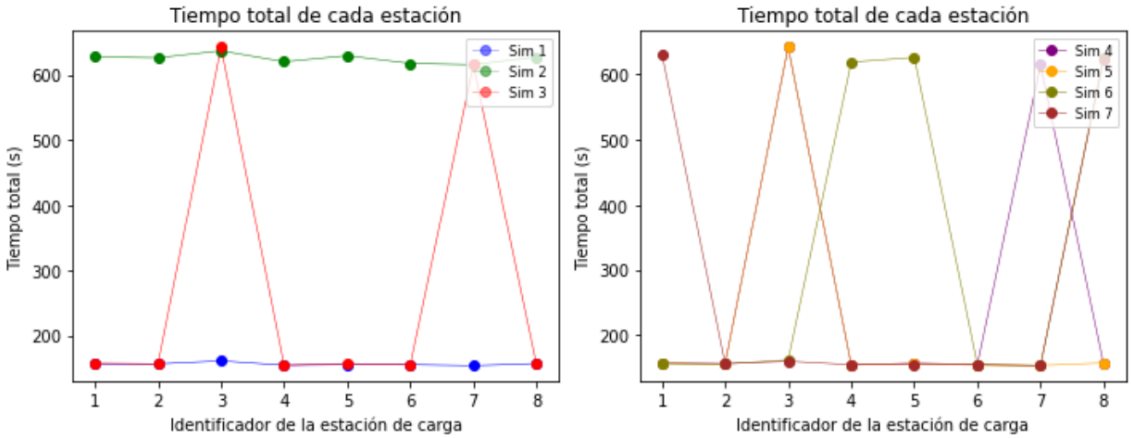

Figura 5.13 Gráficas que representan el tiempo total de cada estación de carga para las simulaciones.

Al igual que en el caso anterior en esta gráfica 5.13 se pueden ver 2 valores de tiempo total muy diferenciados, un valor entorno a 150 segundos y otro valor entorno a 630 segundos. Estos valores se deben en estas simulaciones íntegramente al tiempo de carga medio en cada estación, ya que el tiempo de espera para cada estación es nulo porque las estaciones no llegan a llenarse por completo.

Como ya se ha explicado anteriormente se puede calcular de forma analítica el tiempo de carga mínimo, para las estaciones con una potencia de 200 kW y una energía mínima a cargar de 8000 Wh, el tiempo de carga mínimo es de 151.57 segundos valor ya calculado anteriormente en el punto 5.7. Para comprobar que el tiempo mínimo de carga es correcto, se puede observar en la gráfica 5.13 que los tiempos de carga, que en este caso es igual al tiempo total para las estaciones con una potencia de 200 kW son ligeramente mayores a 150 segundos.

El tiempo mínimo de carga para las estaciones con una potencia de 50 kW y una energía mínima a cargar igual a la anterior de 8000 Wh, tiene un valor de 606.32 segundos, el cálculo se muestra a continuación.

$$t_{carga} = \frac{Energia_{min}}{P_{estacion} * Eficiencia_{carga}} = \frac{8000}{50000 * 0.95} = 0.1684 = 606.32 \quad \text{segundos} \tag{5.11}$$

Al igual que para el anterior tiempo mínimo de carga para las estaciones con una potencia de 200 kW, en las estaciones con potencia de 50 kW el tiempo total debe ser mayor que el tiempo mínimo de carga ya que la mayoría de los vehículos llegan a la estación con una energía a cargar mayor de 8000 Wh por lo que el tiempo de carga aumenta.

6 Conclusiones y trabajos futuros

6.1 Introducción

En este último capítulo se presentarán las conclusiones fundamentales obtenidas a partir de las simulaciones y el análisis de datos realizado en los apartados anteriores, referidos con la movilidad de vehículos eléctricos en entornos urbanos. Estas conclusiones proporcionarán una visión global de los resultados más relevantes que se han obtenido sobre el comportamiento de los vehículos y las estaciones de carga. Además de mostrar las conclusiones, se van a sugerir consideraciones importantes para futuras líneas de investigación.

6.2 Conclusiones

Como se ha podido observar, la eficiencia de una estación de carga está directamente relacionada con su rango de incidencia. En este sentido, el análisis de los resultados obtenidos muestran que la estación de carga más eficiente es la número 3. Esto se debe a su mayor alcance, permitiéndole atender a un mayor número de vehículos en comparación con otras estaciones. Atendiendo a los tiempos de espera de las diferentes estaciones, se puede concluir que las estaciones con un alcance más amplio atraen a un mayor número de vehículos, y por lo tanto, generan una reducción global de los tiempos de espera de cada vehículo, a la vez que una reducción del tiempo total de la estación.

Hay que tener en consideración el análisis de los tiempos de espera por cada estación, debido a que el tiempo que los vehículos esperan cuando una estación está llena es un factor crítico para la satisfacción de los usuarios. Además, la elección del usuario por una estación de carga se ve afectado por el tiempo de carga, relacionado a su vez, con la velocidad/potencia de carga.

Se puede optimizar la red de estaciones de carga basándonos en los resultados obtenidos, con el fin de garantizar una cobertura adecuada y minimizar los tiempos de espera, mejorando el reparto de las estaciones de carga a lo largo de toda la red de carreteras.

6.3 Trabajos futuros

Con el fin de optimizar el control de movilidad de los vehículos eléctricos, sería de gran ayuda que los vehículos supieran a priori el tiempo de espera en cada estación del recorrido,

con el objetivo de dirigirse a la estación de carga donde este tiempo sea menor. Esto ayudaría a disminuir los tiempos de espera y la saturación de las estaciones de carga.

Además, conociendo el gasto de batería por cada kilómetro recorrido por el vehículo, unido a la posibilidad de saber la distancia que hay en cada instante entre el vehículo y la estación de carga siguiente, se puede conocer a que estación de carga podría llegar el vehículo antes de acabar la batería por completo.

A la vez que se podría optimizar el control de las simulaciones, también se podrían mejorar el escenario de simulación y la definición de los vehículos y su itinerario. Para ello sería necesario implementar un código que permitiese generar diferentes trayectos cíclicos a partir de un escenario dado. Estos cambios harían más simples la configuración de los escenarios de simulación.

En un futuro, la optimización de las simulaciones pasa por la implantación de lo anteriormente comentado, reduciendo los tiempos de espera y distribuyendo de manera más eficiente los vehículos en las diferentes estaciones.

Índice de Figuras

Índice de Tablas

Bibliografía

[1] Javier Gómara (1), *Cinco millones de vehículos eléctricos: la cifra que españa necesita para 2030*, Abr 2022.

[2] Volvo Car Corporation (1), *Especificaciones del ex30*, 2023.

[3] Javier Gómara (2), *El gobierno publicará un detallado mapa de puntos de recarga para coches eléctricos*, Feb 2022.

[4] Volvo Car Corporation (2), *Especificaciones del xc60*, 2023.

[5] Andres Acosta, *Traci4matlab*, Mar 2015.

[6] BBVA, *Diferencias entre un híbrido y un híbrido enchufable*, Ene 2020.

[7] Alberto Pardo Calvo, *C4r: Generación de modelos de movilidad para redes de vehículos a partir de mapas reales*, 2011.

[8] Qingqi Pei y Ning Zhang Chen Chen, Yanan Jin, *A connectivity-aware intersection-based routing in vanets*, EURASIP Journal on Wireless Communications and Networking (2014).

[9] Clicars, *Fcev, ¿qué es y qué significan estas siglas?*, Dic 2021.

[10] Gobierno de España, *Perte para el desarrollo del vehículo eléctrico y conectado*, 2021.

[11] D. Tomás de francisco Aparicio, *Simulación de tráfico en ciudades inteligentes con sumo*, 2018.

[12] Carlos Bustillo de Rio, *Iniciativas necesarias para fomentar el uso de vehículos eléctricos*, EL PAÍS (2022).

[13] German Aerospace Center (DLR), *Traci*, 2023.

[14] Manu Granda, *Esta es la distribución de los puntos de carga para coches eléctricos en españa*, EL PAÍS (2022).

[15] IDAE, *Programa moves iii*, Abr 2021.

[16] Viviana Estefany Ríos Ocampo, *Estado del arte de los vehículos eléctricos y su posible implementación en colombia*, 2017.

[17] Abigail Orús, *Número anual de vehículos eléctricos matriculados en españa 2013-2022*, Ene 2023.

[18] Laura Bieker-Walz Jakob Erdmann Yun-Pang Flötteröd Robert Hilbrich Leonhard Lücken Johannes Rummel Peter Wagner Pablo Alvarez Lopez, Michael Behrisch and Evamarie Wießner, *Microscopic traffic simulation using sumo*, Mar 2018.

[19] Ministerio para la Transición Ecológica y el Reto Demográfico, *Real decreto 266/2021*, Abr 2021.

[20] SmartWallboxes, *Guía de la infraestructura de recarga de vehículos eléctricos (irve)*, Feb 2022.

[21] Christoph Sommer, *The open source vehicular network simulation framework*, Mar 2020.

[22] S.Santillán, *Diferencias entre un híbrido y un híbrido enchufable*, Ene 2020.

[23] Juan Antonio Zamora y Edgar Peralta, *Estado del arte en carga estacionaria y dinámica de vehículos eléctricos*, Revista de Tecnología e Innovación (2016).