UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

Maestría en

TELEMÁTICA

RPC-SO-01-NO.025-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO.

TEMA:

DESARROLLO DE UN SISTEMA DE PEAJE DINÁMICO BASADO EN IOT PARA OPTIMIZAR LA GESTIÓN DEL TRÁFICO VEHICULAR

AUTOR:

VÍCTOR ALFREDO PELCHOR ESPINOZA

DIRECTOR:

JORGE OSMANI ORDOÑEZ ORDOÑEZ

GUAYAQUIL - ECUADOR

2024

Autor(es):

Víctor Pelchor Espinoza.
Ingeniero Telemático.
Candidato a Magíster en Telemática por la Universidad Politécnica Salesiana - Sede Guayaquil.
vpelchor@est.ups.edu.ec

Dirigido por:

Jorge Osmani Ordoñez Ordoñez.
Ingeniero Electrónico.
Magíster en Telemática, Universidad Politécnica Salesiana, Ecuador.
jordonezo@ups.edu.ec

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra para fines comerciales, sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual. Se permite la libre difusión de este texto con fines académicos investigativos por cualquier medio, con la debida notificación a los autores.

DERECHOS RESERVADOS
2024 © Universidad Politécnica Salesiana.
GUAYAQUIL - ECUADOR – SUDAMÉRICA.
VÍCTOR ALFREDO PELCHOR ESPINOZA.

DESARROLLO DE UN SISTEMA DE PEAJE DINÁMICO BASADO EN IOT PARA OPTIMIZAR LA GESTIÓN DEL TRÁFICO VEHICULAR.

ÍNDICE

INDICE GENERAL III

LISTA DE TABLAS VI

LISTA DE FIGURAS VII

GLOSARIO X

ACRÓNIMOS XII

RESUMEN XIV

ABSTRACT XVI

AGRADECIMIENTOS XVIII

DEDICATORIA XIX

1. Introducción **1**
- 1.1. Antecedentes y Planteamiento 1
- 1.2. Justificación 4
- 1.3. Objetivo General 6
- 1.4. Objetivos Específicos 6

2. Marco Teórico **7**
- 2.1. Sistemas de peajes estáticos y dinámicos 7
- 2.2. Sistemas de Peajes Dinámicos 8
- 2.3. Infraestructura en peajes dinámicos 10
  - 2.3.1. Sensores de Tráfico Avanzados 10
  - 2.3.2. Comunicación Vehículo a Infraestructura V2I 10
  - 2.3.3. Sistema de posicionamiento global 11

2.3.4. Sistemas de gestión de tráfico centralizados 12
2.3.5. Sistemas de Cobro Automatizados 12
2.4. Características de un peaje dinámico IoT 13
2.5. Funcionamiento de un peaje dinámico IoT 14
2.6. Gestión del Trafico vehicular 15

3. Marco Metodológico 16
3.1. Componentes del Hardware: Dispositivos Electrónicos 17
3.2. Componentes de software 19
3.3. Parámetros del diseño 24
3.3.1. Población y muestra 24
3.3.2. Medios de recolección de datos 24
3.3.3. Procedimiento para la adquisición de datos 25
3.3.4. Análisis de los datos 25
3.3.5. Definición de los parámetros para el análisis de los datos 26
3.3.6. Valores óptimos de los sensores 26
3.3.7. Seguridad de los datos 26
3.4. Estudio de la solución IoT 27
3.5. Arquitectura Propuesta 27
3.5.1. Fase de adquisición de datos 28
3.5.2. Fase de procesamiento de datos 30
3.5.3. Fase de transmisión de datos 31
3.5.4. Fase de Almacenamiento 32
3.5.5. Fase de Simulación 32
3.6. Metodología del diseño 32
3.6.1. Diseño del sistema 33
3.7. Implementación del sistema 34
3.7.1. Creación de un grupo de recursos en IoT Central . . . 34
3.7.2. Creación de un Recurso IoT Hub 34
3.7.3. Instalación de la SDK Azure IoT 37
3.7.4. Creación de un recurso Azure Cosmos DB 38
3.7.5. Configuración de un recurso Stream Analytics Job . . 40
3.7.6. Pruebas Iniciales de Hardware 45
3.7.7. Pruebas de Comunicación y envío de datos 49
3.7.8. Instalación de SUMO y TraCI 53

4. Resultados 56
4.1. Simulación del Trafico Vehicular con SUMO 56
4.2. Análisis de los resultados obtenidos 65
4.3. Evaluación del rendimiento del sistema 67

5. Conclusiones **80**

Bibliografía **82**

LISTA DE TABLAS

3.1. **Especificaciones Técnicas Sensor HC-SR04** 29
3.2. **Especificaciones Técnicas Sensor JSN-SR04T** 29
3.3. **Ventajas y desventajas de los sensores ultrasónicos** . . 30
3.4. **Distancias máximas detectadas según ángulo efectivo de medición: HC-SR04** 47
3.5. **Distancias máximas detectadas según ángulo efectivo de medición: JSN-SR04T** 47

4.1. **Resultados Simulación con Parámetros Iniciales** 66
4.2. **Resultados Simulación con nuevos Parámetros** 69
4.3. **Análisis Comparativo de Resultados de Simulación** . . 71

LISTA DE FIGURAS

2.1. Sistema de Peaje Dinámico (Alinea, 2019). 9

3.1. La Raspberry Pi 3 Model B ofrece una amplia gama de posibilidades para ejecutar diversas aplicaciones. (UNIT Electronics , 2023). 18
3.2. HC-SR04: Pequeño módulo con dos transductores que mide distancias generando pulsos de alta frecuencia. (UNIT Electronics , 2023). 18
3.3. JSN-SR04T: Sensor compacto con una sonda de ultrasonido conectada por cable (UNIT Electronics , 2023). 19
3.4. Azure IoT Central. Fuente: (Microsoft, 2024) 21
3.5. Fases del Diseño Propuesto 28
3.6. Configuración Cliente MQTT. 31
3.7. Diseño del Sistema Peaje IoT 33
3.8. Recursos creados en un Grupo de Recursos IoT 34
3.9. Cadena de conexión Dispositivo N°1. 35
3.10. Cadena de conexión Dispositivo N°2. 35
3.11. Monitoreo Dispositivos IoT: Números de Mensajes Enviados . 36
3.12. Monitoreo Dispositivos IoT: Dispositivos Conectados y Mensajes del Dispositivos Enviados a la Nube 36
3.13. Monitoreo Dispositivos IoT: Total Dispositivos Conectados . . 37
3.14. Clonación de muestras Devices SDK Azure 37
3.15. Instalación de SDK Azure IoT 38
3.16. Recurso Azure cosmos DB 39
3.17. Base de datos y contenedores en Azure Cosmos DB 39
3.18. Funcionamiento de la Base de Datos en Azure Cosmos DB . . 40
3.19. Entrada de datos configurada en Azure Stream Analytics . . . 41
3.20. Salida de datos configurada en Azure Stream Analytics 42
3.21. Consulta N°1 de Datos en Stream Analytics 42

3.22. Consulta N°2 de Datos en Stream Analytics 43
3.23. Resultados de Consultas en Stream Analytics 43
3.24. Métricas Claves de Datos Stream Analytics 44
3.25. Métricas Claves de Datos Stream Analytics 44
3.26. Conexiones Físicas Sensor JSN-SR04T 45
3.27. Conexiones Físicas Sensor HC-SR04 46
3.28. Resultados de Pruebas Iniciales en Sensores 47
3.29. Resultados de Pruebas Iniciales en Sensores 48
3.30. Resultados de Pruebas Iniciales en Sensores 48
3.31. Resultados de Pruebas Iniciales en Sensores 49
3.32. Configuración Plug and Play desde un repositorio de modelos públicos de Azure IoT. 50
3.33. Dispositivo N°1 IoT Creado desde Azure IoT Explorer 50
3.34. Dispositivo N°2 IoT Creado desde Azure IoT Explorer 51
3.35. Datos de Telemetría enviados a la nube por Dispositivo N°1 . 51
3.36. Datos de Telemetría enviados a la nube por Dispositivo N°2 . 52
3.37. Comando ejecutado desde CLI Azure para Monitorear eventos enviados por el Dispositivo N°1 53
3.38. Comando ejecutado desde CLI Azure para Monitorear eventos enviados por el Dispositivo N°2 53
3.39. Instalación de Simulation of Urban Mobility 54
3.40. Instalación de Biblioteca SUMO Tools 55
3.41. Instalación de Traffic Control Interface 55

4.1. Configuración de Nodos en la red vial mediante archivo XML 57
4.2. Configuración de Carriles en la red vial mediante archivo XML 57
4.3. Configuración del peaje en la red vial mediante archivo XML 58
4.4. Configuración del control de trafico del peaje en la red vial mediante archivo XML 58
4.5. Configuración de las conexiones entre nodos en la red vial mediante archivo XML 59
4.6. Configuración de las rutas y demando del trafico en la red vial mediante archivo XML 59
4.7. Código de integración con datos externos mediante TraCI Python 60
4.8. XML Configurado con la Penalización de Tiempo 61
4.9. XML Configurado con la tarifa del Peaje 61
4.10. Detectores y/o semáforos desde el editor de redes viales NetEdit 62
4.11. Archivo ejecutable sumocfg de la Simulación 62
4.12. Comando para ejecutar simulación en SUMO 63

4.13. Simulation ejecutada en Simulation of Urban Mobility 63
4.14. Simulación ejecutada en Simulation of Urban Mobility 64
4.15. Simulación ejecutada en Simulation of Urban Mobility 64
4.16. Penalización de 60 segundos con semáforo en rojo por el Peaje 65
4.17. Resultados de la Simulación con Parámetros Iniciales 66
4.18. Análisis de la Simulación del sistema propuesto con los parámetros iniciales 67
4.19. Nuevos Parámetros de simulación: Tarifa 68
4.20. Nuevos Parámetros de simulación: Tráfico Vehicular 68
4.21. Resultados de la Simulation con los nuevos parámetros 69
4.22. Resultados de la simulación con los Nuevos parámetros 69
4.23. Análisis de los nuevos Parámetros de la simulación 70
4.24. Análisis Comparativo Parámetros: Tiempo promedio de viaje 72
4.25. Análisis Comparativo Parámetros: Velocidad promedio 73
4.26. Análisis Comparativo Parámetros: Tiempo de espera en peaje 74
4.27. Análisis Comparativo Parámetros: Uso de peaje 75
4.28. Análisis diferencial entre los resultados de las simulaciones: Tiempo promedio de viaje 76
4.29. Análisis diferencial entre los resultados de las simulaciones: Velocidad promedio 76
4.30. Análisis diferencial entre los resultados de las simulaciones: Tiempo de espera en peaje. 77
4.31. Análisis diferencial entre los resultados de las simulaciones: Reducción de congestión vehicular. 77
4.32. Análisis diferencial entre los resultados de las simulaciones: Uso de Peaje. 78
4.33. Análisis Comparativo Parámetros: Reducción de congestión vehicular 79

Glosario

API Conjunto de reglas y protocolos que permiten que las aplicaciones se comuniquen entre sí.

Blockchain Es una tecnología que permite registrar y compartir transacciones de forma digital y descentralizada.

CSV Formato de archivo de texto que almacena datos separados por comas.

Deep Learning Es un tipo de machine learnig orientado a aprendizaje autónomo.

Hub Centro de operaciones.

IoT Internet de las cosas.

JSON Formato de texto sencillo para el intercambio de datos.

Plug and Play Característica permite conectar y usar dispositivos externos sin necesidad de configurarlos.

Python Lenguaje de programación.

Raspberry Pi Dispositivo de hardware libre o computadora de bajo costo..

SDK Kit de desarrollo de software.

Sensor Ultrasónico Dispositivos que detectan objetos y miden distancias usando ondas de sonido..

Simulador entorno que recrea un proceso o sistema real para estudiar o comprender su funcionamiento.

SQL Lenguaje de programación para administración de base de datos.

SSL Secure Sockets Layer.

TLS Transport Layer Security.

V2I Comunicación de vehículo a infraestructura.

Wi-Fi Sistema de conexión y fidelidad inalámbrica.

XML Lenguaje de Marcado Extensible.

Acrónimos

RFID identificación por radio frecuencia

SICE Sociedad Ibérica de Construcciones Electrónicas

ITS sistema de transporte inteligente

SEP sistema electrónico de peaje

SOP sistema de operación del peaje

SSC sistema del servicio al cliente

MIT Massachusetts Institute of Technology

DSRC Comunicaciones de radio de corto alcance

DAVAO Sistema de detección de vehículos de alta ocupación

GPS sistema de posicionamiento global

V2V comunicación de vehículo a vehículo

V2I comunicación de vehículo a infraestructura

CLI Interfaz de linea de comandos

GPIO pines de entrada y salida de propósito general

I2C circuito inter-integrado

aPaaS platforma de aplicaciones como servicios

TraCI Traffic Control Interface

SUMO Simulation of Urban Mobility

MQTT Message Queuing Telemetry Transport

TIC Tecnologías de la información y la comunicación

Resumen

En los últimos años, el crecimiento urbano desorganizado ha generado un aumento significativo en la congestión vehicular, afectando la movilidad y la calidad de vida en las ciudades. Una de las principales causas de este problema es el uso de sistemas de peaje obsoletos, incapaces de adaptarse a la demanda dinámica del tránsito. Ante esta situación, países como Chile, Panamá, Israel y Corea del Sur han comenzado a implementar soluciones innovadoras basadas en tecnologías como el Internet de las Cosas (IoT) y la inteligencia artificial, orientadas a mejorar tanto la eficiencia del cobro como la gestión del tráfico. Inspirado en estas iniciativas, el presente trabajo propone el desarrollo de un sistema de peaje dinámico basado en IoT como una alternativa para optimizar el flujo vehicular. Para ello, se plantearon objetivos enfocados en el análisis del estado del arte, el diseño de la arquitectura del sistema, su simulación en un entorno controlado y la evaluación de su rendimiento. La metodología incluyó la integración de sensores ultrasónicos y una Raspberry Pi para la detección de vehículos, el envío de datos en tiempo real a Microsoft Azure IoT, y la simulación del comportamiento del sistema en el software Simulation of Urban Mobility (SUMO).

Los resultados de estas simulaciones evidenciaron una reducción del 13,6 % en el tiempo promedio de viaje (de 12,5 a 10,8 minutos) y un incremento del 14,3 % en la velocidad promedio (de 35 km/h a 40 km/h). Aunque se registró un aumento del 37,2 % en el tiempo de espera en el peaje y una disminución en su uso del 80 % al 70 %, la infraestructura vial fue utilizada de manera más eficiente, reduciendo los niveles de congestión. Estos hallazgos destacan el potencial del peaje dinámico para optimizar la fluidez del tráfico y la movilidad urbana. La integración de tecnologías IoT y herramientas de simulación se consolidan como una estrategia eficaz para enfrentar los desafíos del tránsito en ciudades de alta demanda vehicular.

Este proyecto no solo ofrece una solución técnica, sino que también establece una base sólida para futuras investigaciones sobre la aplicación del IoT para

soluciones al tráfico vehicular. Además, su carácter escalable y adaptable a diferentes escenarios operativos lo posiciona como una herramienta valiosa para abordar diversas problemáticas relacionadas con la congestión vehicular.

Palabras clave: Peaje dinámico, Internet de las cosas (IoT), gestión del tráfico, sensores ultrasónicos, Raspberry Pi, Microsoft Azure IoT, simulación.

Abstract

The rapid and unplanned expansion of urban areas in recent years has led to a significant increase in traffic congestion, adversely affecting mobility and quality of life in cities. A key contributor to this issue is the continued reliance on outdated toll systems that are unable to accommodate the dynamic nature of modern traffic demands. In response, countries such as Chile, Panama, Israel, and South Korea have begun adopting innovative approaches leveraging technologies such as the Internet of Things (IoT) and artificial intelligence to enhance tolling efficiency and traffic management.Motivated by these advancements, this study proposes the development of an IoT-based dynamic toll system as an alternative for optimizing vehicular flow. The objectives encompassed a comprehensive review of the state of the art, the architectural design of the system, simulation in a controlled environment, and performance evaluation. The methodology involved the integration of ultrasonic sensors and a Raspberry Pi for vehicle detection, real-time data transmission to Microsoft Azure IoT, and behavioral simulation of the toll system using SUMO software. Simulation outcomes indicated a 13.6 % reduction in average travel time (from 12.5 to 10.8 minutes) and a 14.3 % increase in average speed (from 35 km/h to 40 km/h). Although an increase of 37.2 % in toll waiting time and a decrease in toll usage from 80 % to 70 % were observed, the overall utilization of the road infrastructure improved, resulting in lower congestion levels. These findings underscore the potential of dynamic tolling systems to enhance traffic flow and urban mobility. Furthermore, the integration of IoT technologies and traffic simulation tools emerges as an effective strategy to address the challenges posed by high-demand urban environments. Beyond presenting a viable technical solution, this project establishes a robust foundation for future research into IoT-based traffic management strategies. Its scalability and adaptability to diverse operational scenarios position the proposed system as a versatile tool for tackling various issues associated with urban traffic congestion.

Keywords: Dynamic tolling, Internet of Things (IoT), traffic management, ultrasonic sensors, Raspberry Pi, simulation.

Agradecimientos

Agradezco a la vida por haberme dado unos padres ejemplares e incondicionales, cuyo amor, apoyo y enseñanzas han sido mi mayor fortaleza en cada etapa de este camino. Su dedicación, esfuerzo y sacrificio han sido fundamentales para que hoy pueda alcanzar este logro.

Gracias por creer en mí, incluso en los momentos más difíciles, por brindarme siempre su aliento y por enseñarme que la perseverancia y la disciplina son la clave para alcanzar cualquier meta.

A ellos, que han sido mi refugio y mi mayor inspiración, les dedico este trabajo con todo mi cariño y gratitud.

Dedicatoria

A lo largo de este camino, he aprendido que el conocimiento es un viaje de esfuerzo, dedicación y, sobre todo, de acompañamiento. Este logro no es solo mío, sino también de aquellas personas que han estado a mi lado brindándome su apoyo incondicional.

Dedico esta tesis, en primer lugar, a mis padres, quienes con su amor, paciencia y sacrificio me han guiado en cada etapa de mi vida. Finalmente, a todas aquellas personas que, de una u otra manera, han dejado una huella en este recorrido. Cada palabra de aliento, cada gesto de apoyo y cada consejo ha sido parte esencial de este logro.

Este trabajo es el resultado de un esfuerzo conjunto, y lo dedico con gratitud y profundo cariño a todos ustedes.

Capítulo 1

Introducción

1.1. Antecedentes y Planteamiento

A medida que las ciudades crecen, especialmente si lo hacen en forma no planificada, la congestión del tránsito se ha convertido en un problema difícil de solucionar. Los niveles de congestión se pueden definir como el tiempo extra promedio que un usuario típico experimenta en una ruta debido a la cantidad excesiva de vehículos que provoca un flujo de tráfico lento Irigoyen [2022]. Una de las causas de la congestión vehicular es la presencia de peajes que no han sido adecuadamente planificados o que carecen de la tecnología apropiada.

Un sistema de peaje es un tipo de infraestructura que permite a los conductores pagar por utilizar una carretera, un puente o un túnel. Estos sistemas son comunes en todo el mundo y son una forma eficaz de financiar el mantenimiento y las mejoras de la infraestructura vial Infraex [2023].

Históricamente, los peajes se implementaron para financiar infraestructuras viales, pero hoy son una fuente importante de ingresos para el mantenimiento y desarrollo de carreteras, puentes y otra infraestructura de transporte Ascencio [2020]. Al convertir el uso del automóvil en menos atractivo, los usuarios buscan otras alternativas de movilidad, como el uso de transporte público y vehículos compartidos, aportando a la mejora de la calidad del aire y generando ingresos adicionales para las ciudades. Estos ingresos pueden ser utilizados para financiar proyectos de mejora urbana y servicios públicos locales, proporcionando una fuente continua de ingresos para el desarrollo urbano del Hierro [2009].

Los sistemas de peaje antiguos usaban métodos de recolección manuales, como barreras y sistemas electrónicos. Estos métodos son propensos a errores humanos y generan retrasos en el tráfico debido a la necesidad de detenerse para realizar el pago. En el caso del peaje electrónico basado en identificación por radio frecuencia (RFID), el vehículo utiliza una etiqueta que abre automáticamente las barreras al pasar por un lector especial. Aunque más rápido, este método aún necesita mucha intervención de las personas y no se ajusta bien a los cambios del tráfico Molina [2019]. No obstante, a pesar de su mayor eficiencia, estos sistemas se basan en una estructura estática de tarifas y no tienen la habilidad para ajustarse al tráfico en tiempo real Globalvia [2012].

Los peajes dinámicos se han convertido en una herramienta importante para el control vehicular en las ciudades modernas. Más allá de su función principal, estos peajes se convierten en una alternativa para fomentar la movilidad sostenible, sino que también permiten el uso de conceptos tecnológicos revolucionarios como el Internet de las cosas IoT Urquiza Cuellar [2022].

El visionario tecnológico, Kevin Ashton, usó por primera vez el término IoT, para describir la conexión de las cosas físicas a un entorno digital a través de una red de sensores y dispositivos inteligentes que conforman una red Gokhale [2018]. El Internet de las cosas permite que los objetos se conecten y se controlen de forma remota a través de una infraestructura de red, creando oportunidades para una integración más directa del mundo físico en sistemas informáticos Gokhale [2018]. Por ejemplo, en 1982 se presentó una máquina de Coca-Cola modificada en Carnegie como el primer dispositivo conectado a Internet, capaz de informar su inventario e identificar si las bebidas cargadas estaban frías Li [2020].

Hoy en día, el IoT se ha convertido en una tecnología con múltiples aplicaciones, entre las cuales destaca la fusión de peajes con IoT. Este avance representa un hito significativo, ya que el pago por el uso de sistemas de vialidad se encuentra en la intersección de la innovación tecnológica y la movilidad urbana Orrala Rey [2023].

Investigaciones previas han explorado el potencial del IoT en el ámbito del transporte. En donde se han implementado soluciones de peaje basadas en IoT en diversas partes del mundo, como Chile, Panamá, Singapur y

Corea del Sur, Ascencio [2020], González García [2017].

La empresa multinacional, Sociedad Ibérica de Construcciones Electrónicas (SICE) implementó en la ciudad de Santiago de Chile una solución innovadora para resolver de forma íntegra el problema de congestión vehicular, la cual consta de una solución Free-Flow en toda la red vial principal de la ciudad con un entorno interoperable conformado por cinco concesiones. Esta solución se basa en un sistema central de peaje y un sistema de gestión de tráfico. El sistema de peaje se compone de un sistema electrónico de peaje (SEP), un sistema de operación del peaje (SOP) y de un sistema del servicio al cliente (SSC) González García [2017].

El SEP se encarga de almacenar y detectar toda la información de los eventos que suceden en la vía, es decir, detectar el paso de vehículos y su clasificación, mientras que el SOP opera y controla el estado del SEP, envío, recepción y tratamiento de toda esta información. Con la finalidad de ejecutar una operación válida, esto es, que sea susceptible de facturación y su posterior emisión de datos. Esta información se almacena por medio del centro de atención al cliente del Hierro [2009].

En Panamá, a través de un estudio realizado por Camarena [2018], se destacan los múltiples beneficios del IoT en el sector del transporte, en específico para el control de congestión vehicular, teniendo como referencia a RoadRunner, un prototipo de aplicación desarrollada por Jason H. Gao y Li-Shiuan Peh, investigadores del Massachusetts Institute of Technology (MIT) Gao [2014].

En Estados Unidos, RoadRunner se aprovecha de la infraestructura tecnológica disponible en este país, recopilando la información de los teléfonos cercanos a cada vehículo utilizando la conexión y comunicación de vehículo a vehículo (V2V), redes Wi-Fi y las Comunicaciones de radio de corto alcance (DSRC) para la detección y control de tráfico a gran escala dentro de una región determinada. Para gestionar el flujo vehicular, este prototipo implementa una estrategia de control que restringe el número de vehículos permitidos en áreas o carreteras con congestión. Esto se logra mediante el requisito de una ficha electrónica de entrada para cada vehículo, que puede circular y ser compartida por varios vehículos a medida que atraviesan diferentes regiones. Estas fichas electrónicas sirven como autorizaciones para acceder a las carreteras y también registran infracciones, aplicando multas por violaciones a las políticas de control de congestión.

En Israel, se implementó una solución de peaje de Indra, una empresa líder en sistema de transporte inteligente (ITS). Esta solución se destaca por ser una de las más avanzadas a nivel mundial, la cual integra capacidades de inteligencia artificial, Deep Learning, Blockchain, así como la interconexión con vehículos conectados y el Sistema de detección de vehículos de alta ocupación (DAVAO) de esta compañía. En concreto, Indra implanta 80 pórticos free-flow, que permiten el paso de los automóviles sin que tengan que reducir la velocidad, y una plataforma de back office operacional, basada en una arquitectura de microservicios, que permite la configuración de los viajes en tiempo real e integra algoritmos que permiten ajustar las tarifas en función del tráfico. La solución de peaje dinámico de Indra no solo mejora su rendimiento, sino que también aprende a medida que captura los datos en tiempo real durante su operación. Además, se caracteriza por ser escalable y altamente flexible, permitiendo adaptarse a cambios en las reglas del negocio según las necesidades del cliente Fernández [2019].

Las soluciones tecnológicas en sistemas de peaje han demostrado beneficios notables, como una gestión de peajes más eficiente, menor tiempo de espera y una mejora en la experiencia del usuario. Esta tecnología emerge como una prometedora respuesta a las limitaciones de los sistemas tradicionales, ofreciendo alternativas como los sistemas de peaje dinámico, los cuales aprovechan la infraestructura tecnológica, la conectividad entre vehículos y sensores inteligentes. Este enfoque permite la recopilación y análisis de los datos sobre el tráfico en tiempo real, demanda del peaje y condiciones de carretera, lo que, a su vez, posibilitaría ajustar de manera inteligente las tarifas según las condiciones actuales del tráfico, optimizando el tránsito vehicular y disminuyendo la congestión en las vías de transporte Ventura Navarro [2018], Leon Martinez [2018].

1.2. Justificación

En la actualidad, los sistemas de peajes tradicionales utilizados en las carreteras y autopistas presentan diversas limitaciones y desafíos en términos de eficiencia, seguridad y sostenibilidad. Estos sistemas suelen ser estáticos, por lo que pueden generar congestiones de tráfico, tiempos de espera prolongados y altos costos de infraestructura.

Un sistema de peaje dinámico debe incorporar un método de pago

inteligente que permita a los usuarios abonar la tarifa sin necesidad de detenerse para recoger un ticket o factura, prescindiendo del uso de dinero en efectivo o tarjetas de crédito/débito. Las prácticas de pago convencionales resultan incómodas para los conductores, ya que implican pérdida de tiempo que se traduce en un aumento del tráfico vehicular.

La implementación de un sistema de peaje dinámico basado en IoT presenta numerosos beneficios potenciales. En primer lugar, permite una gestión más eficiente del tráfico vehicular al adaptar tarifas de peaje en función de la demanda, lo que ayuda a reducir la congestión y mejorar la fluidez del tráfico. Además, estos sistemas proporcionan datos en tiempo real sobre el tránsito y los patrones de movilidad, lo que es valioso para una buena planificación urbana.

El desarrollo de un sistema de peaje dinámico basado en el IoT implica diversos desafíos que requieren una investigación exhaustiva. Algunos de estos desafíos incluyen la identificación y seguimiento de vehículos, la integración con sistemas de navegación, el procesamiento de los datos, la seguridad y privacidad de la información, así como la aceptación y adopción de los usuarios.

Al abordar estos desafíos, los sistemas de peaje dinámico basados en el IoT tienen el potencial de mejorar la gestión del tránsito vehicular y la eficiencia general del transporte. Además, proporcionan una fuente de ingreso más sostenible para el mantenimiento y desarrollo de infraestructuras viales.

Estos tipos de sistemas basados en IoT tienen un impacto positivo en el medio ambiente al disminuir las emisiones de gases de efecto invernadero causadas por la congestión vehicular González García [2017].

Por lo tanto, existe una clara necesidad de aportar con nuevas soluciones que ayuden a mejorar la eficiencia de la gestión del tránsito, disminuir la congestión, promover la sostenibilidad, optimizar la gestión de infraestructuras viales y generar ingresos adicionales. Estos beneficios hacen que sea importante abordar el problema y explorar soluciones innovadoras basadas en la tecnología IoT.

1.3. Objetivo General

Desarrollar un sistema de peaje dinámico basado en IoT para optimizar la gestión del tráfico vehicular.

1.4. Objetivos Específicos

1. Analizar el funcionamiento y las características de IoT, así como los peajes dinámicos y la gestión del tráfico vehicular.
2. Diseñar un sistema de peaje dinámico utilizando la tecnología IoT, integrando tanto hardware como software.
3. Realizar una simulación de un sistema de peaje dinámico basado en IoT en un entorno piloto con características reales.
4. Evaluar los resultados de la simulación del peaje dinámico piloto.

Capítulo 2

Marco Teórico

Un sistema de peaje estático aplica tarifas fijas sin considerar la demanda de tráfico, basándose únicamente en factores como la distancia recorrida o el tipo de vehículo. Este enfoque puede generar congestión en horas pico, ya que no incentiva a los conductores a modificar sus horarios de viaje. En contraste, un sistema de peaje dinámico ajusta las tarifas en tiempo real según la demanda, aumentando los costos en períodos de alta congestión y reduciéndolos en momentos de menor tráfico; este mecanismo permite distribuir los viajes de manera más equilibrada a lo largo del día, contribuyendo a la reducción de la congestión y a una mayor eficiencia en la circulación vehicular.

2.1. Sistemas de peajes estáticos y dinámicos

La evolución de los peajes estáticos a los dinámicos ha sido un proceso gradual que ha estado impulsado por avances tecnológicos y la necesidad de gestionar de forma más eficiente el tráfico en las carreteras.

Inicialmente, los peajes eran estáticos, con tarifas fijas basadas en criterios como la distancia recorrida o el tipo de vehículo. Con el tiempo, se comenzaron a explorar enfoques más dinámicos para establecer tarifas de peaje, especialmente en áreas urbanas con altos niveles de congestión.

El desarrollo de tecnologías como los sistemas GPS, la comunicación móvil y los sensores de tráfico permite la implementación de sistemas de peaje dinámico que ajustan las tarifas en tiempo real según la demanda de tráfico.

Estos sistemas dinámicos han demostrado ser más efectivos para gestionar la congestión vehicular al incentivar a los conductores a viajar en momentos de menor congestión o utilizar rutas alternativas. Además, los avances en tecnología han permitido la implementación de sistemas de peaje sin barreras, que utilizan tecnología de radiofrecuencia o cámaras para detectar y cobrar peajes de forma automática, lo que ha contribuido aún más a la evolución de los peajes hacia sistemas más eficientes.

2.2. Sistemas de Peajes Dinámicos

En los sistemas inteligentes de transporte, los peajes dinámicos, representados en la Fig. 2.1, se destacan como nuevas herramientas para el control del tránsito. Estos sistemas ayudan a mitigar problemas como la congestión vehicular, los accidentes y la contaminación ambiental. A nivel mundial, estos sistemas se han convertido en una tendencia tecnológica que optimiza la movilidad, permitiendo un control eficiente de la operación y administración del tráfico y el transporte. Además, garantizan que las necesidades de movilidad de los usuarios sean atendidas de manera eficiente, económica y segura Vaca [2015].

Debido al constante avance tecnológico, estos tipos de sistemas también deben ajustarse a las nuevas exigencias y demandas para mejorar la experiencia y el servicio a los usuarios. El IoT se ha convertido en una herramienta importante para la gestión inteligente del transporte, permitiendo la conexión y comunicación entre dispositivos físicos y sistemas de información en tiempo real. En el contexto del transporte, el IoT facilita la recopilación de datos sobre el tráfico, las condiciones de la carretera y los vehículos, entre otros. Estos datos son fundamentales para la implementación de sistemas de peaje dinámico, ya que permiten establecer tarifas variables en función de la demanda y las condiciones de tráfico. Por ejemplo, mediante sensores colocados en la carretera y en los vehículos, el IoT puede proporcionar información en tiempo real sobre la densidad de tráfico, la velocidad de los vehículos y las condiciones climáticas, lo que permite ajustar las tarifas de peaje de manera dinámica para gestionar la congestión y mejorar la fluidez del tráfico Pazmiño [2019].

El concepto de los sistemas de peajes dinámicos es una aplicación directa de la tecnología IoT. A diferencia de los peajes estáticos, donde las tarifas son fijas, los peajes dinámicos utilizan la información proporcionada por el IoT para ajustar las tarifas en función de la demanda y las condiciones del tránsito vehicular. Esto significa que en horas pico o en situaciones de tráfico intenso, las tarifas pueden aumentar para desincentivar la circulación, mientras que en momentos de menor congestión, las tarifas pueden reducirse para fomentar un flujo más uniforme de vehículos Vasquez [2020].

Figura 2.1: Sistema de Peaje Dinámico (Alinea, 2019).

2.3. Infraestructuras tecnológicas implementadas en los sistemas de peajes dinámicos

Los sistemas de peajes dinámicos representan una evolución significativa para la gestión del tránsito, especialmente en entornos urbanos y metropolitanos. Estos sistemas utilizan una variedad de tecnologías e infraestructuras especializadas en recopilar datos, analizar patrones de tráfico y ajustar las tarifas de peaje de manera dinámica según la demanda y las condiciones de tráfico. Entre las infraestructuras clave implementadas en los sistemas de peajes dinámicos se incluyen sensores de tráfico avanzados, sistemas de comunicación vehículo a infraestructura, sistemas de posicionamiento global, sistemas de gestión de tráfico centralizados y sistemas de cobro electrónico sin contacto.

Estas infraestructuras trabajan en conjunto para mejorar la gestión del tráfico y la eficiencia del transporte. Además, proporcionan una experiencia de conducción más segura y fluida para los usuarios en la carretera.

2.3.1. Sensores de Tráfico Avanzados

Los sistemas de peaje dinámico suelen utilizar una variedad de sensores para recopilar datos sobre el flujo de vehículos. Estos sensores pueden incluir cámaras de vídeo, sensores ultrasónicos, magnéticos y de velocidad, entre otros. Estos sensores proporcionan información sobre la cantidad de vehículos, la velocidad a la que se desplazan y la densidad del tráfico, lo que permite al sistema ajustar las tarifas de peaje según la demanda.

2.3.2. Comunicación Vehículo a Infraestructura V2I

Los sistemas inteligentes de transporte se distinguen por el uso intensivo de tecnologías de la información y comunicación Tecnologías de la información y la comunicación (TIC) con el objetivo de mejorar la seguridad y gestión vial, además de disminuir el impacto ambiental y energético de los vehículos en las carreteras. En este contexto, diversas iniciativas y proyectos están explorando el potencial de las tecnologías de comunicaciones inalámbricas aplicadas a estos sistemas. Estos sistemas inteligentes permiten una gestión más eficiente y segura del tránsito mediante el intercambio de información en tiempo real entre vehículos V2V y entre vehículos e infraestructura V2I, lo que facilita una respuesta más ágil

y precisa ante las condiciones cambiantes del entorno vial de la Fuente [2011].

La incorporación de sistemas integrados en los vehículos ha tenido un notable crecimiento, orientándose hacia una interacción eficiente entre el hombre y la máquina. Esta integración busca no solo mejorar la comodidad del conductor y los pasajeros, sino también establecer una asistencia a la conducción que sea poco intrusiva y que, al mismo tiempo, incremente la seguridad vial. Este avance ha sido impulsado principalmente por el desarrollo de tecnologías de comunicación y protocolo diseñados para facilitar la interacción tanto entre vehículos y el conductor como entre vehículos, promoviendo así entornos de conducción más seguros y conectados Florido [2018].

La comunicación de vehículo a infraestructura (V2I) permite que los vehículos se comuniquen con la infraestructura de la carretera, como los sistemas de peaje, para proporcionar información sobre su ubicación, velocidad y destino. Esto permite a los sistemas de peaje dinámico recopilar datos precisos sobre el tráfico y ajustar las tarifas en consecuencia.

2.3.3. Sistema de posicionamiento global

El sistema de posicionamiento global (GPS) es una tecnología de localización que permite obtener la posición y ubicación en cualquier lugar del mundo, las 24 horas del día y bajo cualquier condición climática de un dispositivo. Su funcionamiento se basa en el cálculo de coordenadas tridimensionales mediante la medición de distancias a al menos tres satélites cuya ubicación es previamente conocida, lo que permite establecer la ubicación exacta del receptor en el espacio Letham [2001], Angela Ribeiro [2000].

El sistema GPS es una herramienta que se ha utilizado con gran éxito en el ámbito tecnológico. Por tal razón, se convierte en un componente importante para la gestión de los sistemas de peajes inteligentes, Roberto Rodriguez Solano [1999].

El uso del GPS en los sistemas de peajes dinámicos radica en transmitir los datos y rastrear la ubicación de los vehículos en tiempo real y determinar su ruta. Esto resulta muy útil, ya que permite establecer el valor de una tarifa en relación a la ruta tomada y la distancia recorrida por cada vehículo. La implementación de estos sistemas en las urbes es fundamental, ya que

están orientados a mejorar la eficiencia del transporte urbano y ofrecer soluciones concretas para reducir la congestión vehicular. Asimismo, al considerar factores como horarios y rutas a través de redes de comunicación, es esencial que el diseño de estos sistemas incluya mensajes claros y precisos, permitiendo que los usuarios accedan a la información de forma rápida y segura. Toalombo [2021]

2.3.4. Sistemas de gestión de tráfico centralizados

Estos sistemas recopilan y procesan los datos de tráfico recopilados por los sensores y otros dispositivos. Utilizan algoritmos sofisticados para analizar los datos y ajustar las tarifas de peaje de manera dinámica para gestionar la congestión y mejorar la fluidez del tráfico.

2.3.5. Sistemas de Cobro Automatizados

Otra de las aplicaciones a las que se han orientado los sistemas de transporte inteligentes son los sistemas de tarifa automatizada. Desde el cobro de pasajes de buses en estaciones hasta sistemas de peajes, estos sistemas han evolucionado en función de brindar mayor eficacia al momento de realizar este tipo de trámites que anteriormente se realizaban de persona a persona. La gestión de estos sistemas no solamente se ha focalizado en zonas urbanas, sino que se ha expandido también a entornos rurales como es el caso de la implementación de sistemas de peajes electrónicos inteligentes Toalombo [2021].

Por otra lado, el uso de los sistemas convencionales ha requerido la presencia de personas de forma continua, generando gastos en personal. Actualmente, estos se han ido reemplazando mediante sistemas de cobro electrónicos, automatizados e inteligentes. Los sistemas de peaje dinámico suelen utilizar sistemas de cobro electrónico sin contacto, como RFID o pagos móviles, para agilizar el proceso de pago y eliminar la necesidad de barreras físicas y minimizar las interrupciones en las vías del peaje Toalombo [2021], Urquiza Cuellar [2022].

La gestión de los sistemas de peajes inteligentes no solamente implica el reemplazo del personal al momento del cobro, sino que busca la gestión de información que se recolecta de las vías para determinar aquellos tramos que requerirán mayor atención en relación a aquellos que son menos transitados. En este sentido, se puede también regular la economía y cobros que pueden

ajustarse a tarifas de acuerdo a la época en la que funcionan los peajes, el número de autos que circulan así como el tiempo de viaje que lleva un automóvil. Todos estos datos pueden ser procesados en función de garantizar la economía de las personas que frecuentan las zonas de peaje, así como permitir el mantenimiento de las carreteras a través de la recolección necesaria de dinero que demanda de acuerdo al tráfico que abarcan en función del tiempo Francesc Robusté [2003].

2.4. Características de un sistema de peaje dinámico basado en IoT

los sistemas de peajes dinámicos basados en IoT ofrecen una serie de ventajas sobre los sistemas de peajes estáticos tradicionales, incluyendo una gestión más eficiente del tráfico, una mayor flexibilidad, adaptabilidad y una mayor sostenibilidad en el transporte. Los sistemas de peajes dinámicos basados en IoT tienen varias características distintivas que los diferencian de los sistemas de peajes estáticos tradicionales. Algunas de las características clave incluyen:

- ***Tarifas variables en tiempo Real:*** Estas tarifas se ajustan dinámicamente según la demanda de tráfico, lo que permite mitigar la congestión en horas pico y distribuir mejor el flujo vehicular.
- ***Recopilación de datos en tiempo real:*** Con la incorporación de sensores IoT se realiza una recopilación continua de los datos de forma precisa sobre el tráfico, densidad de vehículos, la velocidad promedio y las condiciones del entorno vial.
- ***Análisis y gestión inteligente del Tráfico:*** Los datos recopilados se utilizan para realizar análisis inteligentes, que permiten identificar patrones de trafico, predecir situaciones de congestión y tomar decisiones en tiempo real para mejorar la fluidez vehicular.
- ***Cobro electrónico sin contacto:*** se implementan métodos de cobro electrónicos, como tecnologías RFID o pagos móviles, para aceleran el proceso de pago y eliminar la necesidad de detenerse en las estaciones de peaje.
- ***Comunicación Vehículo a Infraestructura V2I:*** Permiten la comunicación entre los vehículos y la infraestructura de la carretera,

lo que facilita la gestión del tráfico y la optimización de las rutas de viaje.

- ***Flexibilidad y adaptabilidad:*** Los sistemas de peaje dinámicos basados en IoT pueden adaptarse a cambios en las condiciones del tráfico y en los patrones de viaje, lo que los hace más flexibles que los sistemas de peajes estáticos.
- ***Mayor eficiencia y sostenibilidad:*** Al gestionar de manera más eficiente el tráfico, los sistemas de peaje dinámicos basados en IoT pueden reducir la congestión, mejorar la fluidez del tráfico y reducir las emisiones de carbono asociadas con la circulación vehicular.
- ***Interfaz de usuario:*** Ofrecen interfaces de usuario intuitivas, ya sea a través de aplicaciones móviles o páginas web, para que los usuarios puedan consultar información sobre tarifas, tráfico y rutas alternativas.
- ***Gestión Remota:*** Permiten la gestión remota del sistema, lo que facilita la configuración, supervisión y actualización de los dispositivos desde una ubicación centralizada

2.5. Funcionamiento de un sistema de peaje dinámico basado en IoT

Los sistemas de peaje dinámico basado en IoT operan mediante la recopilación y análisis de los datos en tiempo real sobre el flujo vehicular, empleando una red de sensores y dispositivos conectados. Esta infraestructura permite monitorear variables como la densidad de tráfico, velocidad promedio y condiciones de la vía. Con base en esta información, el sistema ajusta automáticamente las tarifas de peaje, promoviendo una gestión eficiente de la congestión vial y mejorando la fluidez del tránsito. El funcionamiento general de este tipo de sistema se describe de la siguiente manera:

1. ***Recopilación de Datos:*** El sistema utiliza sensores IoT, como sistemas GPS, cámaras de video vigilancia y sensores de velocidad, para recopilar datos sobre el tráfico vehicular, incluyendo la cantidad de vehículos, la velocidad a la que se desplazan y las condiciones de la carretera.

2. ***Procesamiento de Datos:*** Los datos se envían a un sistema centralizado de gestión de tráfico, donde se procesan y analizan utilizando algoritmos inteligentes. Este análisis permite identificar patrones y tendencias en el tráfico, así como predecir la demanda futura.

3. ***Ajustes de tarifas:*** El sistema ajusta dinámicamente las tarifas de peaje en tiempo real. Por ejemplo, las tarifas pueden aumentar durante las horas pico para desincentivar la circulación o reducirse en momentos de menor demanda para fomentar un flujo más uniforme de vehículos.

4. ***Comunicación de conductores:*** Los conductores son informados sobre las tarifas de peaje actuales a través de paneles de información en la carretera o aplicaciones móviles. Esto les permite tomar decisiones informadas sobre sus rutas y horarios de viaje.

5. ***Método de cobro:*** El sistema utiliza métodos de pagos atomizados, para cobrar de forma automática a los Vehículos que pasan por una estación de peaje.

2.6. Características de la gestión del tráfico vehicular.

La congestión vehicular se presenta cuando el volumen de vehículos en una vía excede su capacidad operativa, provocando una disminución considerable de la velocidad de circulación y un aumento en los tiempos de desplazamiento que hacen tiempos de viaje más largos. Algunas de las características principales de la congestión vehicular son:

- Reducción de la velocidad de los vehículos.
- Aumento en los tiempos de viaje.
- Flujo irregular de vehículos, con paradas y arranques frecuentes.
- Incremento en la contaminación del aire debido a vehículos en mal estado.
- Generación de estrés y malestar en los conductores.
- Impacto económico negativo debido a costos adicionales de combustible y pérdida de productividad.

Capítulo 3

Marco Metodológico

En este proyecto se aplicaron diferentes metodologías de investigación tales como las: descriptivas, experimentales, cualitativas y cuantitativas. Con la metodología descriptiva se realizó una revisión bibliográfica y recopilación de información desde artículos científicos, informes, conferencias y tesis relacionadas con el tema de investigación.

A través de una metodología experimental se realizaron estudios y análisis de los dispositivos IoT apropiados a utilizar en el sistema de peaje dinámico propuesto; así como también la infraestructura tecnológica a implementar. Estos estudios experimentales permitieron seleccionar el hardware y software adecuados para el diseño del sistema de peaje dinámico basado en IoT.

Con las metodologías cualitativas y cuantitativas se logró evaluar el rendimiento y eficacia del sistema propuesto bajo condiciones controladas. Así como también comparar los datos y resultados obtenidos de la simulación del sistema de peaje dinámico basado en IoT con proyectos similares, con el propósito de identificar posibles mejoras, evaluar su viabilidad y validar la efectividad del sistema propuesto. En resumen, este proyecto consiste en 3 fases, las mismas que permitieron cumplir con los objetivos planteados.

En la primera fase se recopiló información sobre los componentes que se necesitan para el desarrollo del sistema de peaje dinámico basado en IoT.

La segunda fase consistió en el diseño e implementación del sistema propuesto, el cual se basa en la detección de vehículos, adquisición y

transmisión de los datos a la plataforma IoT y el envío de esos datos al entorno de simulación.

En la tercera fase se realizó la simulación del peaje IoT en función de los datos obtenidos por los sensores.

3.1. Componentes del Hardware: Dispositivos Electrónicos

Raspberry Pi 3 Model B: En la Fig. 3.1, se muestra una Raspberry Pi que es un ordenador compacto y de bajo costo utilizado en diversos proyectos de electrónica y programación gracias a su bajo consumo de energía. Está equipada con 40 pines como puertos GPIO para conexiones de entrada y salida, así como una interfaz de circuito inter-integrado (I2C) para la gestión e interacción con sensores físicos. La Raspberry Pi actúa como la unidad central de procesamiento, encargándose de recopilar los datos de los sensores y transmitirlos a la plataforma Microsoft Azure IoT.

Sensor Ultrasónico HC-SR04: El HC-SR04 de la Fig. 3.2, es un Sensor Ultrasónico de bajo costo y alta precisión utilizado para medir distancias. Es ampliamente utilizado en proyectos de robótica y sistemas de medición de distancia. Es similar al JSN-SR04T, emite una señal de sonido de alta frecuencia y mide el tiempo que tarda en regresar después de rebotar en un objeto. Su rango de medición es de 0.02 m a 4 m operando en una frecuencia de 40KHz.

Sensor Ultrasónico JSN-SR04T: El JSN-SR04T de la Fig. 3.3, es un Sensor Ultrasónico especialmente diseñado para ser resistente al agua, lo que lo hace ideal para aplicaciones en exteriores y entornos húmedos. Utiliza ondas ultrasónicas para medir la distancia entre el sensor y un objeto: emite una señal de sonido de alta frecuencia que se refleja en el objeto y regresa al sensor. El tiempo que tarda en regresar dicha señal se utiliza para calcular la distancia. Tiene un rango de medición de 0.2 m a 6 m y opera a una frecuencia de 40 kHz, lo que lo convierte en una opción adecuada para aplicaciones de medición en exteriores.

Figura 3.1: La Raspberry Pi 3 Model B ofrece una amplia gama de posibilidades para ejecutar diversas aplicaciones. (UNIT Electronics , 2023).

Figura 3.2: HC-SR04: Pequeño módulo con dos transductores que mide distancias generando pulsos de alta frecuencia. (UNIT Electronics , 2023).

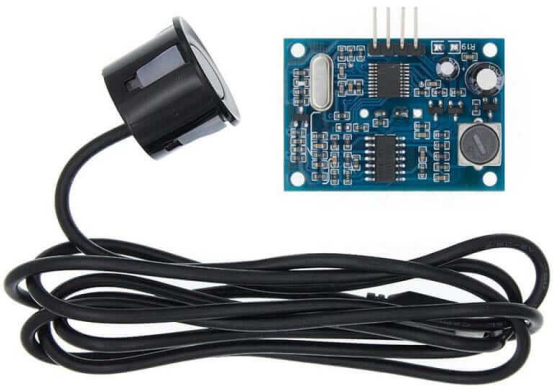

Figura 3.3: JSN-SR04T: Sensor compacto con una sonda de ultrasonido conectada por cable (UNIT Electronics , 2023).

3.2. Componentes de software

En esta sección se describen los componentes de software que fueron utilizados para el diseño del sistema propuesto, incluyendo aquellos responsables del envío, almacenamiento y recepción de los datos generados por los sensores.

Raspberry Pi OS: Raspbian es un sistema operativo basado en Debian optimizado para la Raspberry Pi, entre sus características cuenta con el soporte para programación en Python y otros lenguajes. Este sistema operativo cuenta con una versión Desktop y una versión Lite, básicamente la versión Desktop cuenta con una interfaz gráfica muy amigable con el usuario, mientras que la versión Lite es un sistema operativo muy ligero con un conjunto mínimo de paquetes que cuenta con una Interfaz de linea de comandos (CLI).

Biblioteca RPi.GPIO: Esta librería proporciona un módulo de Python para controlar los puertos GPIO en una Raspberry Pi. Estos son pines de entrada y salida de propósito general (GPIO) que sirven para interactuar con un entorno físico a través de sensores.

Comunicación I2C: Una comunicación I2C es un protocolo de comunicaciones que se usa como interfaz para hardware como sensores. Esta interfaz es necesaria para interactuar con los sensores físicos. Por defecto, la interfaz I2C viene deshabilitada, por lo que es necesario habilitar esta interfaz para lograr la interacción de los sensores con la Raspberry Pi a través de la SDK basada en Python.

Azure IoT Central: es una platforma de aplicaciones como servicios (aPaaS), que facilita el desarrollo de soluciones IoT al proporcionar un conjunto de API preconfiguradas. Esta plataforma permite crear, conectar y administrar grandes cantidades de dispositivos de forma escalable. A través de su interfaz web, los usuarios pueden integrar fácilmente dispositivos, supervisar su estado, configurar reglas automatizadas y administrar tanto los dispositivos como los datos durante todo su ciclo de vida, Microsoft [2024d].

Azure IoT Central se utiliza para almacenar y visualizar los datos recolectados por los sensores ultrasónicos que miden la distancia de los vehículos entre las estaciones del peaje. Estos datos se envían desde la Raspberry Pi a Azure IoT Central, donde se analizan y se usan para tomar decisiones sobre la gestión del tráfico vehicular.

La Fig.3.4 muestra un esquema de los elementos que conforman la arquitectura de una solución en Azure IoT Central.

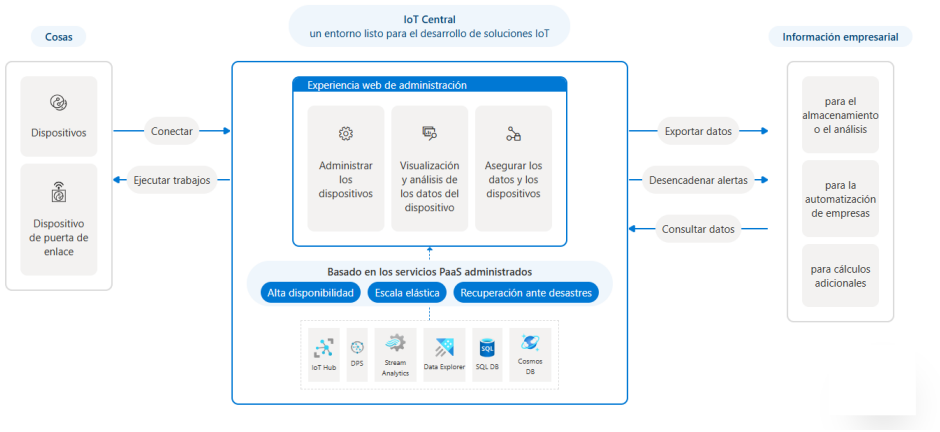

Figura 3.4: Azure IoT Central. Fuente: (Microsoft, 2024)

Azure IoT Device SDK: Una Azure IoT SDK permite crear aplicaciones que se ejecutan en dispositivos IoT utilizando el cliente del dispositivo o el cliente del módulo. Estas aplicaciones envían telemetría a un IoT Hub y, opcionalmente, reciben mensajes, trabajos, métodos o actualizaciones gemelas desde su centro de IoT. Puede usar estos SDK para crear aplicaciones de dispositivos que usen modelos de Azure IoT Plug and Play para anunciar sus capacidades en aplicaciones habilitadas. Microsoft [2024a].

Las SDK instaladas en dispositivos IoT contienen un código que facilita la creación de aplicaciones, las cuales se conectan y son administradas por los servicios de Azure IoT Hub. Estas SDK pueden ejecutarse en un dispositivo informático general basado en microprocesadores, como una Raspberry Pi. Estas admiten el desarrollo en lenguajes de programación modernos, incluidos C, NodeJS, Python y Java Microsoft [2024a].

Las SDK básicamente, son APIs que facilitan la integración y la comunicación entre los dispositivos y la nube. Sobre una Raspberry Pi se instaló una SDK de Azure IoT basada en Python, la cual permite enviar los datos de los sensores a Azure IoT Central a través de Azure IoT Hub. La SDK facilita la autenticación, el envío de datos y la gestión de la conectividad, asegurando una transmisión de datos segura y eficiente.

Azure IoT HUB: IoT Hub es un servicio que permite una comunicación segura y confiable entre las aplicaciones de internet de las cosas y los dispositivos que gestiona. Azure IoT Hub actúa como una solución de backend en la nube que facilita la conexión con prácticamente cualquier tipo de dispositivo. Además, permite extender la solución desde la nube hasta el borde, gracias a funciones como la autenticación individual por dispositivo y la gestión centralizada Microsoft [2024c].

IoT Hub aprovecha los datos de telemetría enviados por los dispositivos hacia la nube para monitorear su estado y establecer rutas de mensajes hacia otros servicios de Azure, todo esto sin necesidad de una programación adicional. También permite el envío de mensajes desde la nube hacia los dispositivos Microsoft [2024c].

Azure Stream Analytics: es un servicio diseñado para el análisis de datos en tiempo real, que permite procesar grandes volúmenes de información en movimiento. Utiliza consultas basadas en SQL para transformar y examinar datos instantáneamente, facilitando su integración con otros servicios de Azure para el almacenamiento y uso de los resultados Microsoft [2024e].

Este recurso permitió analizar en tiempo real los datos del tráfico enviados a Azure IoT central. Lo que facilitó la detección de patrones, anomalías y eventos importantes «como situaciones de congestión», generando alertas e informes instantáneos que contribuyen a una gestión del tráfico más eficiente y oportuna.

Azure Cosmos DB: es un servicio de base de datos no relacional totalmente administrado y distribuido globalmente, diseñado para ofrecer alta escalabilidad y baja latencia. Admite diversos modelos de datos, como documentos, gráficos y tablas, y permite almacenar documentos JSON nativos con esquemas flexibles. Además, los datos se indexan automáticamente, los cuales pueden ser consultados mediante una consulta SQL Microsoft [2024b].

Azure Cosmos DB se utilizó para almacenar los datos históricos del tráfico vehicular recolectados por los sensores. Gracias a su escalabilidad y capacidad para realizar consultas avanzadas, permitió llevar a cabo análisis detallados y extraer información importante sobre los patrones de tráfico en distintos períodos de tiempo.

Traffic Control Interface: TraCI, es una interfaz que permite establecer comunicación en tiempo real entre un simulador de tráfico y aplicaciones externas. A través de un protocolo basado en TCP, facilita el intercambio de datos y comandos, lo que permite controlar y monitorear el comportamiento del simulador desde otros programas SUMO [2024b].

Traffic Control Interface (TraCI) se utiliza para enviar los datos en tiempo real desde la Raspberry Pi a SUMO, permitiendo que el simulador de tráfico utilice estos datos para actualizar y controlar la simulación de tráfico urbano. Esto permite evaluar el impacto de diferentes estrategias de peaje dinámico y optimizar la gestión del tráfico.

Simulation Urban Mobility: SUMO, es un simulador de código abierto diseñado para representar y analizar el comportamiento del tránsito vehicular en entornos urbanos. Permite realizar simulaciones detalladas de vehículos individuales y redes viales completas, y se puede integrar con herramientas externas mediante la interfaz de TraCI SUMO [2024a].

se utilizó SUMO para simular el tráfico vehicular y evaluar el impacto de diversas estrategias de peaje dinámico, utilizando los datos recolectados por los sensores. Esta simulación permitió visualizar el efecto del sistema sobre el flujo vehicular y facilitó la optimización de decisiones de gestión antes de una posible implementación en entornos reales.

3.3. Parámetros del diseño

Para el diseño del sistema propuesto se consideraron varios parámetros clave como: técnicos, funcionales, metodológicos, de seguridad y de escalabilidad.

Técnicamente, se utilizan sensores ultrasónicos para medir distancias, y una Raspberry Pi 3 como plataforma de hardware. Funcionalmente, los datos se recolectan y se envían a Azure IoT Central a través de Azure IoT Hub utilizando el Azure IoT Device SDK, con análisis en tiempo real mediante Azure Stream Analytics y almacenamiento en Azure Cosmos DB. Metodológicamente, se tomó una muestra representativa de vehículos y se realiza la recolección de datos durante un mes, con análisis y validación posterior. En términos de seguridad, se implementa encriptación TLS/SSL y autenticación mediante claves de Azure IoT Central.

Finalmente, el sistema está diseñado para ser escalable, permitiendo la incorporación de más sensores en diferentes ubicaciones sin comprometer el rendimiento, lo que facilita la expansión del sistema para cubrir áreas de tráfico más amplias o aumentar la precisión de la medición.

3.3.1. Población y muestra

La población considerada para este proyecto incluye a todos los vehículos y conductores que circulan en un lugar determinado. Para realizar un estudio representativo, se selecciona una muestra de aproximadamente 50 vehículos y 1 punto de tráfico estratégico. Esta muestra se elige para asegurar la representatividad de diferentes tipos de vehículos, horarios y áreas de tráfico, permitiendo así una evaluación precisa del impacto del sistema de peaje dinámico sobre el flujo vehicular. La recolección de datos se realiza durante un mes, midiendo las distancias de los vehículos cada segundo y enviando estos datos a Azure IoT Central para su análisis. Este enfoque permitió obtener información detallada y relevante que puede ser utilizada para optimizar la gestión del tráfico vehicular en la ciudad.

3.3.2. Medios de recolección de datos

Se utilizaron 2 clases de sensores para la recolección de datos que resultaron ser instrumentos clave para adquirir los datos. Los principales son los sensores ultrasónicos JSN-SR04T y HC-SR04, que realizaron la

detección y cálculo de la distancia de los vehículos. Estos sensores están conectados a una Raspberry Pi, que actúa como plataforma de procesamiento y comunicación. La Raspberry Pi ejecuta scripts en Python que utilizan la biblioteca RPi.GPIO para interactuar con los sensores y el Azure IoT Device SDK para enviar los datos recolectados a Azure IoT Central a través de Azure IoT Hub.

3.3.3. Procedimiento para la adquisición de datos

Este procedimiento implica 2 etapas. Primero, se identifican y seleccionan ubicaciones estratégicas para maximizar la precisión y cobertura de los sensores ultrasónicos. Segundo, se instalan los sensores, asegurándose de que estén orientados correctamente para medir la distancia de los vehículos que pasan. Los datos se recolectan continuamente, con mediciones realizadas cada segundo y enviadas en intervalos de cinco segundos para minimizar el tráfico de red. Azure Stream Analytics procesa los datos en tiempo real para detectar patrones y eventos significativos, mientras que Azure Cosmos DB almacena los datos históricos para análisis posteriores.

3.3.4. Análisis de los datos

Ya recolectados los datos, se inicia el proceso del análisis de estos datos, esencial para optimizar la gestión del tráfico vehicular. Estos datos se transmiten y son procesados en tiempo real mediante Azure Stream Analytics para identificar patrones.

Los datos históricos se almacenan en Azure cosmos DB, lo que permite un análisis detallado a largo plazo y el uso de técnicas predictivas para anticipar futuros patrones de tráfico. La integración con el simulador SUMO mediante la interfaz TraCI permite simular diferentes escenarios de tráfico y evaluar estrategias de peaje dinámico, ajustando tarifas y horarios de peaje según los resultados de las simulaciones.

Los Dashboards en Azure IoT Central proporcionan visualizaciones intuitivas de los datos en tiempo real, facilitando el monitoreo continuo del sistema. Reportes periódicos resumen los hallazgos y ofrecen recomendaciones basadas en los insights obtenidos. Además, se implementa un ciclo de retroalimentación que utiliza los datos analizados para realizar ajustes y optimizaciones constantes en el sistema de peaje dinámico, asegurando una mejora continua en la gestión del tráfico vehicular.

3.3.5. Definición de los parámetros para el análisis de los datos

Se establecieron parámetros importantes para asegurar un análisis de datos efectivo y preciso. Estos parámetros guían la fase de recolección, procesamiento y evaluación de los datos, incluyendo la frecuencia de muestreo de los sensores «mediciones cada segundo y transmisiones cada cinco segundos», umbrales de distancia para registrar datos relevantes. Estos datos se almacenan en Azure Cosmos DB y se integran con SUMO para simular un peaje dinámico, estableciendo indicadores de rendimiento como el flujo vehicular, tiempo de espera y eficiencia del peaje,

3.3.6. Valores óptimos de los sensores

Determinar los valores óptimos de los sensores ultrasónicos es vital para dar fiabilidad a los datos adquiridos. Estos sensores deben estar calibrados para medir distancias en un rango de 0.2 a 5 metros, con una precisión de ± 1 cm, aunque en condiciones abiertas los sensores lograron medir distancias de 7 hasta 12 metros. La frecuencia de muestreo se establece en una lectura por segundo para captar cambios rápidos en el tráfico vehicular. Además, es esencial que los sensores operen en condiciones ambientales estables, evitando interferencias causadas por lluvia o polvo. La instalación debe garantizar un ángulo de medición correcto para evitar lecturas falsas, y los datos recolectados se deben filtrar para eliminar valores atípicos, calculando promedios de múltiples lecturas para obtener resultados precisos y consistentes.

3.3.7. Seguridad de los datos

La seguridad de los datos es un aspecto fundamental para preservar la integridad y privacidad de la información recolectada. Para ello, se implementaron múltiples capas de protección, comenzando con la encriptación de los datos mediante protocolos como TLS, que aseguran la comunicación entre sensores, la Raspberry Pi y Azure IoT Central. Además, se emplearon credenciales seguras y mecanismos de autenticación basados en certificados digitales, garantizando que únicamente los dispositivos autorizados puedan transmitir datos hacia la nube. La información almacenada en Azure Cosmos DB está protegida con cifrado avanzado y políticas de acceso restringidas, permitiendo el acceso exclusivamente a usuarios y aplicaciones autorizadas. Adicionalmente, se ejecutan auditorías de seguridad de forma periódica para detectar actividades anómalas o

intentos de acceso no autorizado, asegurando así la confidencialidad, integridad y disponibilidad de los datos en todo momento.

3.4. Estudio de la solución IoT para optimizar el trafico vehicular

El estudio de esta solución se centra en integrar tecnologías avanzadas para mejorar la gestión y fluidez del tránsito; utilizando sensores ultrasónicos, con los cuales se monitorea en tiempo real el flujo de los vehículos. Los datos recolectados se envían a Azure IoT Central a través de Azure IoT Hub, donde se procesan y analizan mediante Azure Stream Analytics. Este proceso permite detectar patrones de tráfico y eventos significativos, facilitando la toma de decisiones sobre el ajuste de tarifas y horarios de peaje. Además, los datos almacenados en Azure Cosmos DB permiten realizar análisis predictivos y modelado de escenarios futuros. La integración con el simulador SUMO a través de TraCI permite evaluar diferentes estrategias de peaje en un entorno controlado, optimizando la gestión del tráfico en función de los resultados obtenidos. Este enfoque holístico y basado en datos asegura una gestión del tráfico más eficiente, reduciendo la congestión y mejorando la experiencia de los usuarios.

3.5. Arquitectura Propuesta

La arquitectura del sistema de peaje dinámico basado en IoT está diseñada para optimizar la gestión del tráfico vehicular mediante la integración de hardware y software libre. Utiliza sensores ultrasónicos conectados a una Raspberry Pi para detectar vehículos en tiempo real, enviando los datos a la nube a través de Azure IoT Hub. Azure IoT Central centraliza la gestión y monitoreo de dispositivos, mientras que Azure Stream Analytics analiza el tráfico en tiempo real. Los datos históricos se almacenan en Azure Cosmos DB para análisis predictivo. Además, la integración con SUMO y TraCI permite simular escenarios y evaluar estrategias de peaje.

En la Fig 3.5, se muestra a detalle la arquitectura tecnológica en la que se basó el diseño del sistema propuesto, que básicamente se resume en 5 fases. Esta arquitectura modular y escalable asegura una gestión del tráfico eficiente y permite ajustes y optimizaciones continuas basadas en los datos recolectados y analizados.

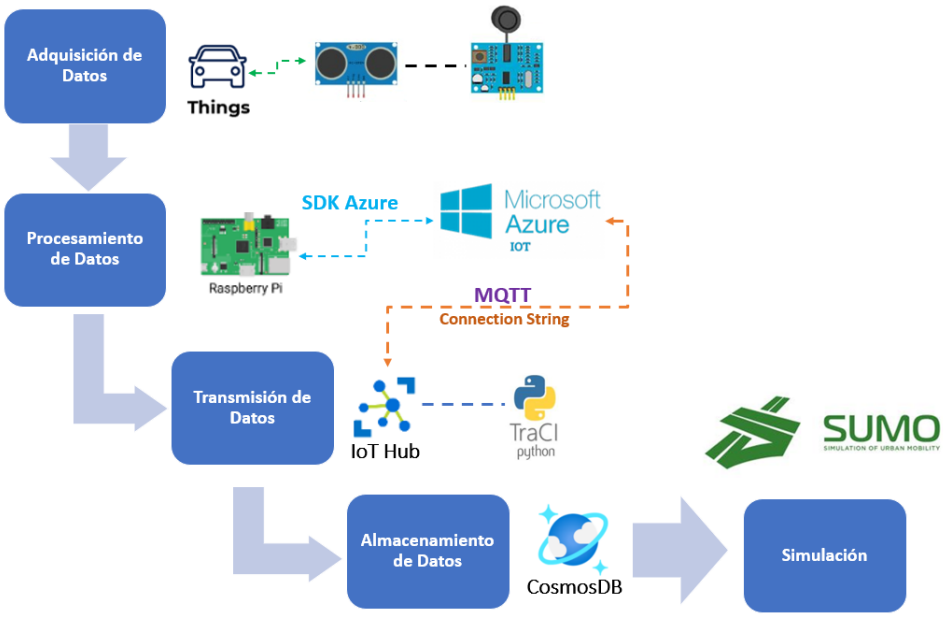

Figura 3.5: Fases del Diseño Propuesto

3.5.1. Fase de adquisición de datos

Para la fase de adquisición de datos se utilizan sensores ultrasónicos para detectar vehículos y calcular la distancia en tiempo real. Estos sensores están conectados a una Raspberry Pi, que realiza las mediciones con una frecuencia de muestreo de una lectura por segundo, asegurando precisión y fiabilidad en la detección.

Del análisis realizado a los sensores ultrasónicos JSN-SR04T y HC-SR04 se tuvieron en cuenta especificaciones técnicas que se puedan ajustar al diseño del sistema, así como también algunos aspectos importantes tales como:

- Precio.
- Disponibilidad.
- Ventajas.
- Desventajas.

En las tablas: 3.1, 3.2, y 3.3 se muestran las especificaciones técnicas de estos dispositivos, ventajas, desventajas y precio.

Cuadro 3.1: **Especificaciones Técnicas Sensor HC-SR04**

Característica	**Especificación**
Tipo	ultrasónico
Modelo	HC-SR04
Dimensiones	45 mm x 20 mm x 15 mm
Peso	9 gramos
Voltaje de operación	+5V DC
Corriente de trabajo	30mA
Rango de medición	2 cm a 4m
Frecuencia de trabajo	40KHz
Ángulo de medición efectivo	$<15^{o}$
Ángulo de medición	30^{o}
Precisión	± 3 mm
Pines	Vcc, Trigger, Echo, GND

Cuadro 3.2: **Especificaciones Técnicas Sensor JSN-SR04T**

Característica	**Especificación**
Tipo	ultrasónico
Modelo	JSN SR04T
Dimensiones	40 mm x 28 mm
Voltaje de Operación	+5V DC
Corriente de trabajo	30mA
Rango de medición	2 cm a 6m
Frecuencia de trabajo	40KHz
Ángulo de medición efectivo	$<50^{o}$
Ángulo de medición	75^{o}
Precisión	3mm
Longuitud del cable	2.5m
Temperatura de trabajo	-10^{o}C hasta 70^{o}C
Pines	Vcc, Trigger, Echo, GND

Cuadro 3.3: **Ventajas y desventajas de los sensores ultrasónicos**

Dispositivo	Ventaja	Desventaja	Precio
JSN-SR04T	Ideal para zona abiertas y exteriores, resistente al agua.	Su cable es propenso a daños debido a su exposición.	$15,00
HC-SR04	Sensor de alta precisión	Su detección efectiva depende del ángulo de operación.	$3,50

La elección de estos sensores se basó en su alta fiabilidad en la precisión de los datos y de su alta disponibilidad en el mercado, además de ajustarse al presupuesto destinado para el desarrollo del sistema propuesto.

3.5.2. Fase de procesamiento de datos

Los datos recolectados por los sensores se procesan localmente en la Raspberry Pi para filtrar lecturas anómalas y calcular promedios. Este preprocesamiento mejora la calidad de los datos antes de su transmisión a la nube. El envío de datos a Azure IoT Central desde la Raspberry Pi se realiza típicamente utilizando el protocolo Message Queuing Telemetry Transport (MQTT). Este protocolo de mensajería es ligero y eficiente, ideal para dispositivos IoT debido a su bajo consumo de ancho de banda y capacidad de operar en redes inestables. La ventaja del protocolo MQTT es que soporta encriptación TLS para asegurar la privacidad y seguridad de los datos en tránsito.

Se creó un cliente MQTT que se conecta a Azure IoT Central utilizando una cadena de conexión generada por el dispositivo IoT desde el recurso IoT Hub. Luego, se define una función para enviar y empaquetar los datos en un mensaje MQTT y los envía a Azure IoT Central.

En la fig 3.6, se muestra la configuración del cliente MQTT que realiza el envío de los datos a la plataforma IoT.

```python
from azure.iot.device import IoTHubDeviceClient, Message

# Conexión a Azure IoT Central
connection_string = "HostName={your-hub-name}.azure-devices.net;DeviceId={your-device-
client = IoTHubDeviceClient.create_from_connection_string(connection_string)

# Función para enviar datos
def send_data_to_azure(payload):
    message = Message(payload)
    client.send_message(message)
    print("Mensaje enviado: ", payload)

# Ejemplo de envío de datos
payload = '{"distance": 2.5}'
send_data_to_azure(payload)
```

Figura 3.6: Configuración Cliente MQTT.

3.5.3. Fase de transmisión de datos

Los datos procesados se envían desde la Raspberry Pi a Azure IoT Central a través de Azure IoT Hub. La transmisión se realiza cada cinco segundos para equilibrar la carga de la red y mantener actualizaciones constantes.

En la fase de transmisión de datos, Raspberry Pi procesa los datos recolectados por los sensores ultrasónicos y los envía a Azure IoT Hub utilizando el protocolo MQTT. Esta transmisión se realiza en intervalos de cinco segundos para asegurar una actualización constante de la información.

Azure IoT Hub actúa como un puente, recibiendo los datos y enviándolos a Azure IoT Central, donde se monitorean y analizan en tiempo real. Este proceso asegura que los datos recolectados sean accesibles de manera inmediata para la toma de decisiones y análisis posteriores. Simultáneamente, se utiliza la interfaz TraCI para transmitir datos al Simulador SUMO, permitiendo la evaluación de diferentes escenarios de tráfico y estrategias de peaje dinámico.

3.5.4. Fase de Almacenamiento

Base de Datos: Azure Cosmos DB

Para La fase de almacenamiento de los datos se utilizó Azure cosmos DB para guardar los datos históricos recolectados por los sensores. Se establecieron las fuentes de datos, tanto de entrada como de salida, así como una consulta que permitió visualizar los datos obtenidos y depurados.

Azure Cosmos DB es una base de datos No SQL distribuida globalmente que ofrece alta disponibilidad y baja latencia, ideal para manejar grandes volúmenes de datos IoT. Los datos se estructuran en colecciones y se organizan para facilitar consultas y análisis eficientes. Esta fase permite el acceso a datos históricos para realizar análisis predictivos y modelado de escenarios futuros, proporcionando una base sólida para decisiones basadas en datos.

3.5.5. Fase de Simulación

En la fase de simulación, se utilizó SUMO como entorno para modelar y evaluar diferentes escenarios de tráfico. La interfaz TraCI permite la comunicación entre el Simulador y los datos obtenidos. Los datos de tráfico se transmiten a SUMO, donde se configuran escenarios de tráfico y se evalúan las estrategias de peaje dinámico. Los resultados de la simulación se analizan para optimizar las estrategias de gestión del tráfico, ajustando los parámetros en función de los resultados obtenidos para mejorar la eficiencia y reducir la congestión vehicular.

3.6. Metodología del diseño

La metodología aplicada para el diseño del sistema propuesto sigue un enfoque estructurado e iterativo, comenzando con el análisis de requisitos y la definición de objetivos claros. Se diseña una arquitectura integrada de hardware y software, seleccionando tecnologías adecuadas como MQTT para la transmisión de los datos a la plataforma IoT y TraCI para el envío de estos datos al entorno de simulación. En la fase de desarrollo, los sensores ultrasónicos se integran con la Raspberry Pi, y se configuran recursos de Azure IoT Central para la gestión y almacenamiento de datos. Se realizan pruebas unitarias e integrales para validar la funcionalidad del sistema, seguidas por la simulación en SUMO para optimizar las estrategias de peaje.

Finalmente, el sistema se despliega en un entorno real, con monitoreo y mantenimiento continuo para asegurar su rendimiento y escalabilidad.

3.6.1. Diseño del sistema

El diseño del sistema propuesto consiste en el uso de una arquitectura basada en hardware y software libre, además de una plataforma IoT modular que permite el análisis y la visualización de los datos.

En el diagrama de la Fig. 3.7, la Raspberry Pi se comunica con los sensores a través de la comunicación Serial-UART para obtener la distancia detectada. Luego, utiliza TraCI para enviar esta información a la simulación en SUMO, donde puede utilizarse para controlar el tráfico en la simulación. Al mismo tiempo, la Raspberry Pi ejecuta una SDK basada en Python que permite el cálculo y envío de los datos de la distancia hacia Azure IoT Central a través de Azure IoT Hub, utilizando MQTT para la comunicación. Esta integración permitirá controlar el tráfico en la simulación de SUMO en función de los datos obtenidos en tiempo real y también almacenar y visualizar estos datos en Azure IoT Central para su posterior análisis.

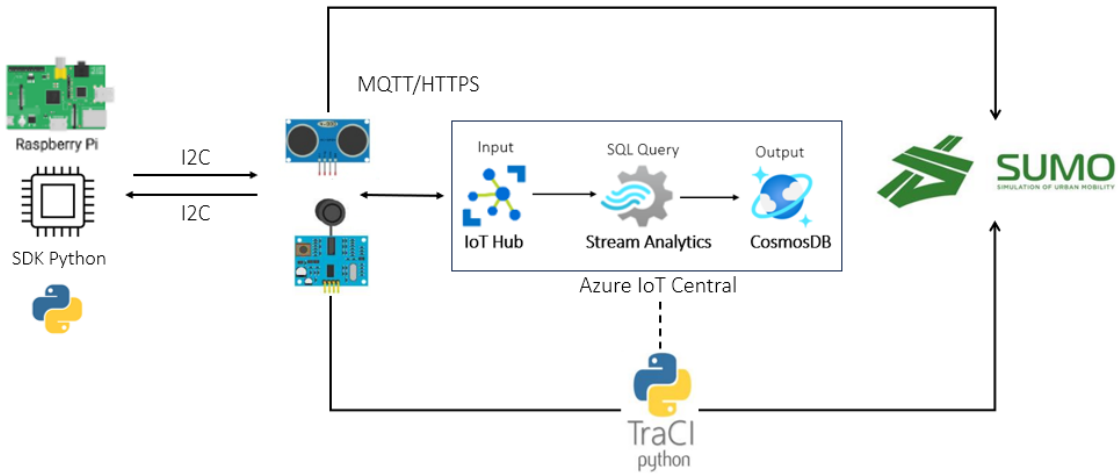

Figura 3.7: Diseño del Sistema Peaje IoT

3.7. Implementación del sistema

3.7.1. Creación de un grupo de recursos en IoT Central

La Plataforma IoT se basa en 3 recursos de Azure IoT Central que permiten administrar, agregar dispositivos, analizar y almacenar los datos; estos recursos son: IoT Hub, Azure Cosmos DB y Stream Analytics Job. Los recursos deben estar agregados a un grupo de recursos en Azure IoT Central para que funcione como una solución única.

Ya creado el grupo de recursos en Azure IoT, se pueden crear los recursos anteriormente mencionados para la ejecución de este proyecto. En la Fig. 3.8, se muestra el grupo de recursos con sus respectivos recursos IoT creados.

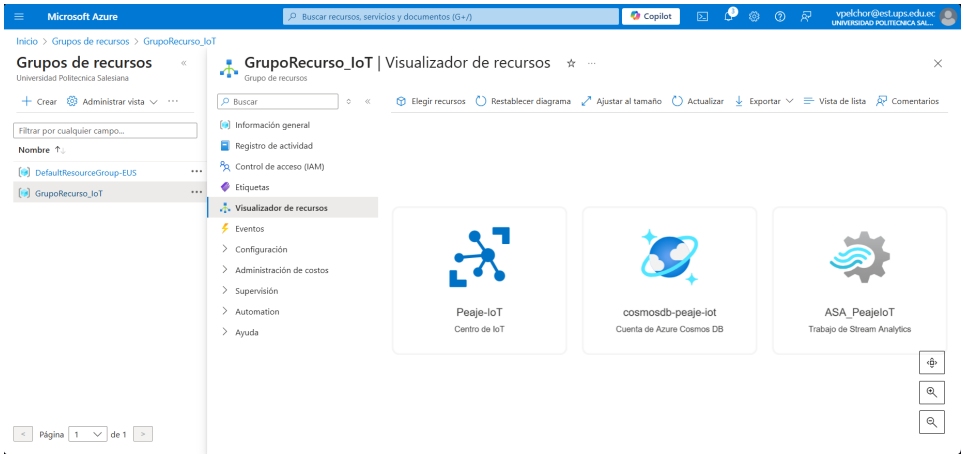

Figura 3.8: Recursos creados en un Grupo de Recursos IoT

3.7.2. Creación de un Recurso IoT Hub

IoT Hub permite crear, administrar, configurar, agregar y conectar dispositivos IoT a la solución IoT propuesta.

La conexión de estos dispositivos se realiza a través de una cadena de conexión encriptada obtenida desde IoT Hub. Esta cadena de conexión permite almacenar y enviar los datos desde una Raspberry Pi hacia la plataforma cloud usando el SDK de Azure IoT. En la Fig. 3.9 y Fig. 3.10, se muestra la cadena de conexión encriptada usada sobre las Raspberry Pi.

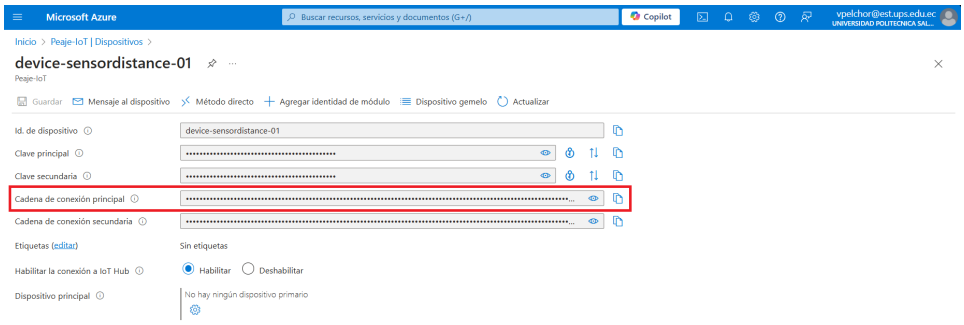

Figura 3.9: Cadena de conexión Dispositivo N°1.

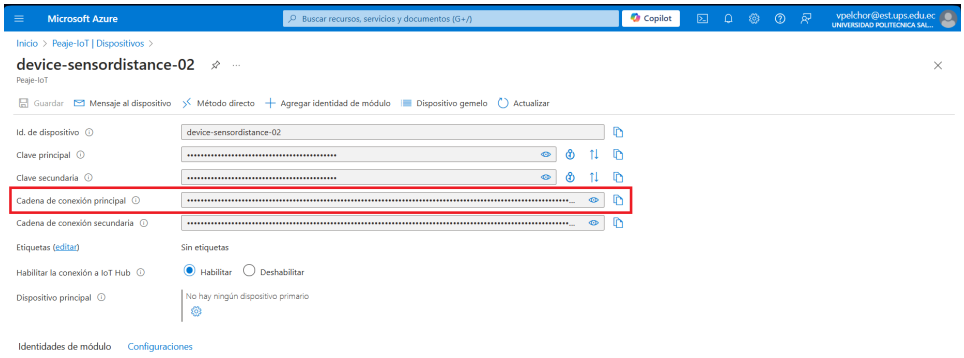

Figura 3.10: Cadena de conexión Dispositivo N°2.

Una vez creado el recurso Centro IoT y haber conectado los dispositivos a la plataforma Azure IoT Central, desde el Centro IoT se puede monitorear nuestros dispositivos a través de paneles personalizados. Al tener una suscripción gratuita, solo se dispone de 5 paneles que servirán para el monitoreo de los dispositivos que conforman el sistema de peaje dinámico, Fig. 3.11, Fig. 3.12 y Fig. 3.13.

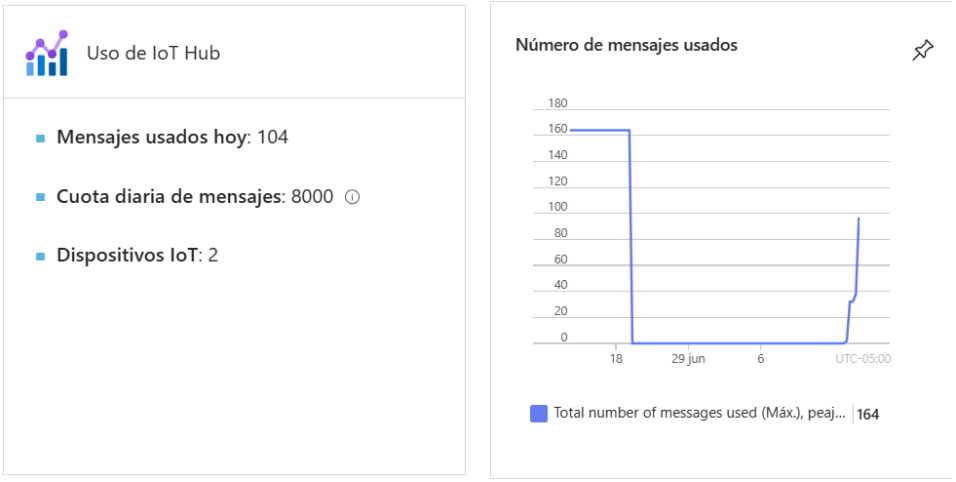

Figura 3.11: Monitoreo Dispositivos IoT: Números de Mensajes Enviados

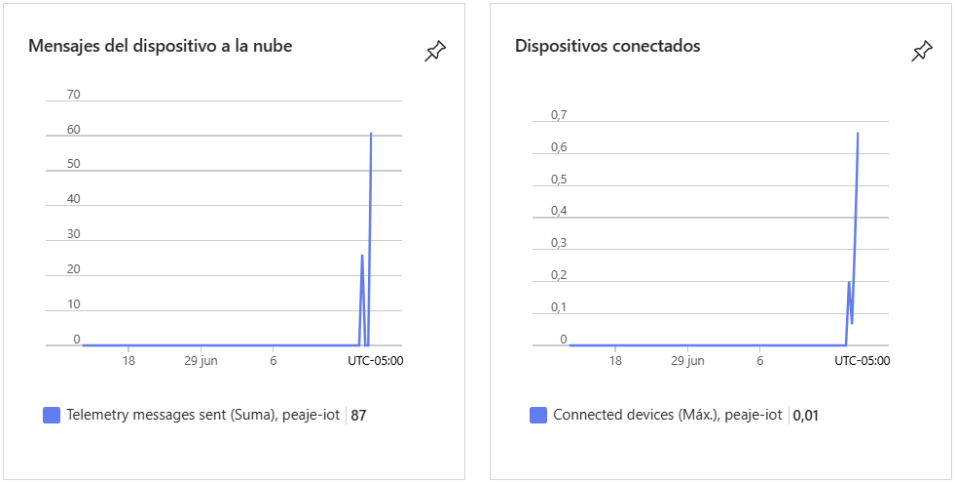

Figura 3.12: Monitoreo Dispositivos IoT: Dispositivos Conectados y Mensajes del Dispositivos Enviados a la Nube

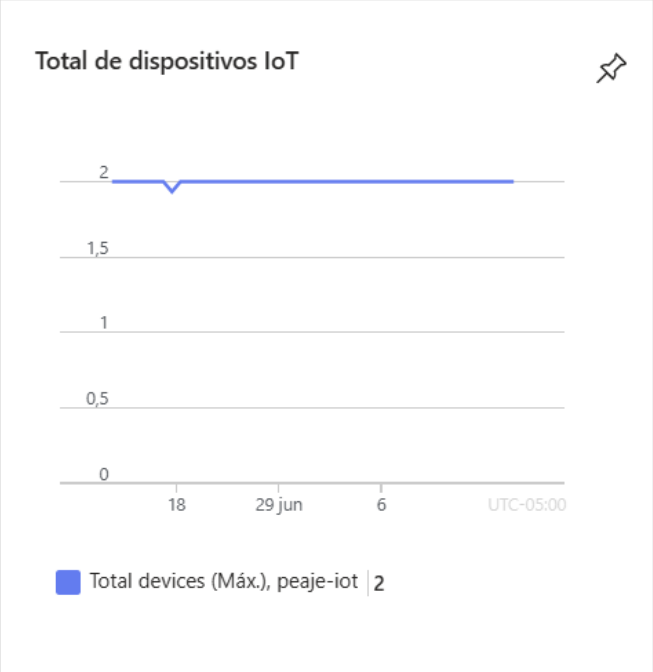

Figura 3.13: Monitoreo Dispositivos IoT: Total Dispositivos Conectados

3.7.3. Instalación de la SDK Azure IoT

Se clonó una muestra de las SDK de python soportadas para Microsoft Azure IoT sobre las Raspberries, Fig. 3.14. Posteriormente, se instaló la API de Azure IoT que permitirá enviar los datos desde un dispositivo al centro de IoT mediante comunicación MQTT, Fig. 3.15.

```
Consola
git clone --branch v2 https://github.com/Azure/azure-iot-sdk-python
```

Figura 3.14: Clonación de muestras Devices SDK Azure

```
Consola

pip3 install azure-iot-device
```

Figura 3.15: Instalación de SDK Azure IoT

Elaboración del código Python

Se trabajó en el desarrollo de un código en Python que permite calcular la distancia y que estos datos sean enviados a Azure IoT a través de una función sobre el cliente MQTT. Se configuran los puertos GPIO para que realicen la conexión con los sensores; adicionalmente, se establece una variable de entorno para que los dispositivos se puedan conectar a Azure IoT. Como valor de la variable, se usa la cadena de conexión de los sensores creados en IoT Hub, como se muestra en la Fig. 3.9 y Fig. 3.10.

Ejecución del código Python

La ejecución del código en Python se realizará de forma automática, es decir, que no será necesario ejecutarlo por comandos desde un terminal. Para ejecutar automáticamente el archivo .py al iniciar la Raspberry Pi, se puede configurar un servicio de systemd.

3.7.4. Creación de un recurso Azure Cosmos DB

Un Recurso Azure Cosmos DB permite almacenar los datos obtenidos. Este modelo de API se basa en una base de datos SQL usando contenedores y permite realizar consultas SQL a la base de datos.

Para este recurso se debe configurar una cuenta de Cosmos DB y se debe estar suscrito al grupo de recursos creado para la solución IoT del sistema.

En la fig. 3.16 y fig. 3.17 se puede observar desde el panel central la información general del recurso de Azure Cosmos DB creado, así como la base de datos y los contenedores creados.

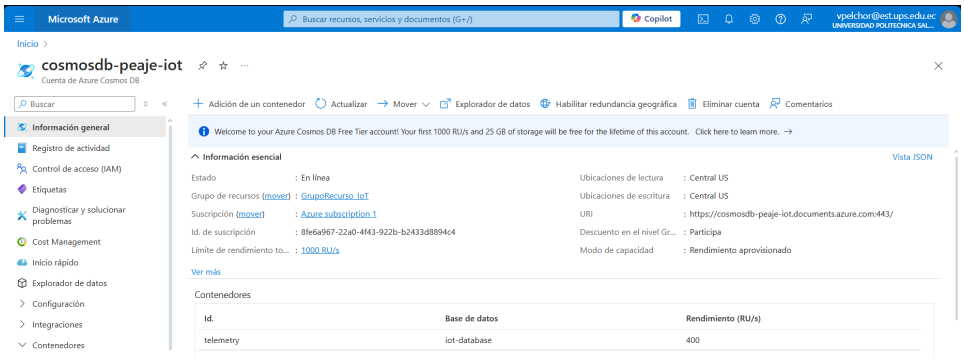

Figura 3.16: Recurso Azure cosmos DB

Configuración de una Base de Datos y Contenedores

Una vez creada la cuenta de Cosmos DB, se debe configurar la base de datos así como crear un contenedor de datos en donde se almacenarán los datos obtenidos desde la Raspberry Pi. Ya creado el recurso de Azure Cosmos DB se podrán visualizar las solicitudes de almacenamiento a la base de datos así como también el rendimiento de esta misma, Fig. 3.18.

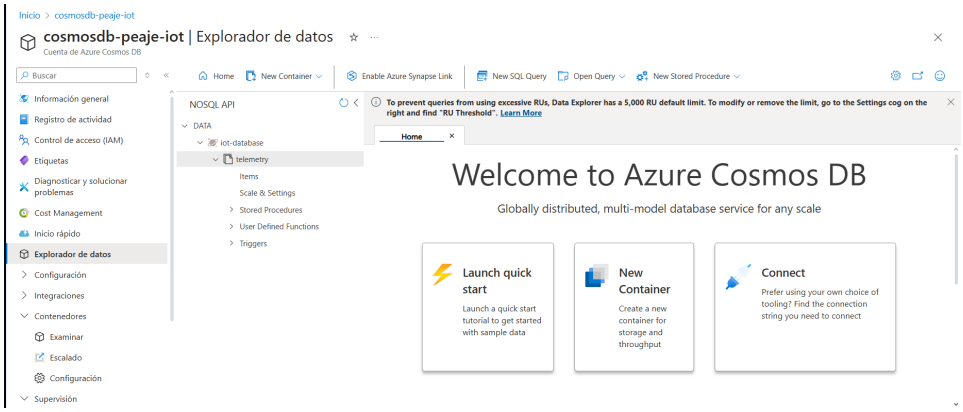

Figura 3.17: Base de datos y contenedores en Azure Cosmos DB

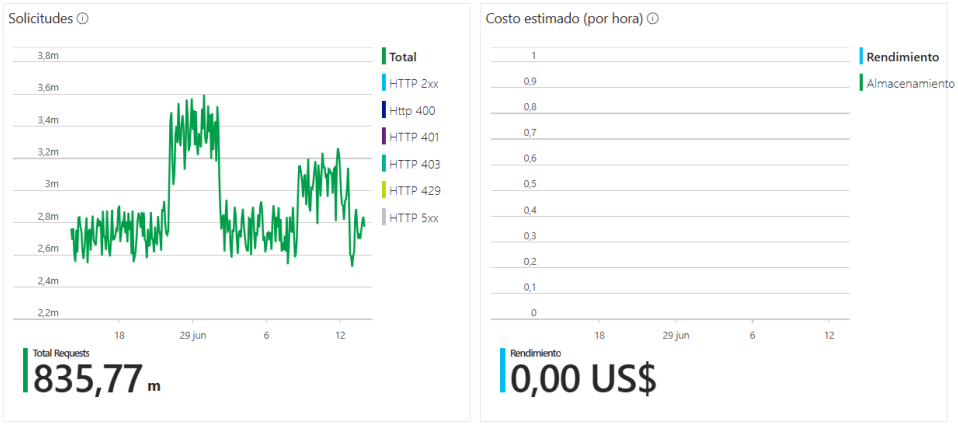

Figura 3.18: Funcionamiento de la Base de Datos en Azure Cosmos DB

3.7.5. Configuración de un recurso Stream Analytics Job

Azure Stream Analytics es un servicio que permite analizar y procesar grandes cantidades de datos en movimiento. También ayuda a la detección de anomalías, identificar comportamientos inusuales en los datos de sensores IoT. Adicionalmente, cuenta con mantenimiento predictivo que predice fallos en equipos antes de que ocurran, mediante el análisis continuo de los datos de sensores.

Configuración de entradas y salidas de Datos

Configurar las entradas y salidas de datos en un trabajo de Azure Stream Analytics es esencial para definir de dónde provienen los datos y a dónde deben ir después de ser procesados y cómo los datos fluyen a través del sistema. Las entradas especifican las fuentes de datos que serán procesadas, mientras que las salidas definen los destinos para los datos procesados. Esta configuración permite que Azure Stream Analytics procese los datos de manera eficiente, proporcionando información valiosa y oportuna para la toma de decisiones.

Configuración de una entrada de datos

Las entradas definen las fuentes de datos en tiempo real que el trabajo de Stream Analytics va a procesar. Configurar las entradas permite a Azure

Stream Analytics saber de dónde leer los datos.
Para este proyecto y caso puntual se definió como fuente de datos el centro IoT creado para este sistema.

En la Fig. 3.19, se muestra la entrada de datos creada en Azure Stream Analytics.

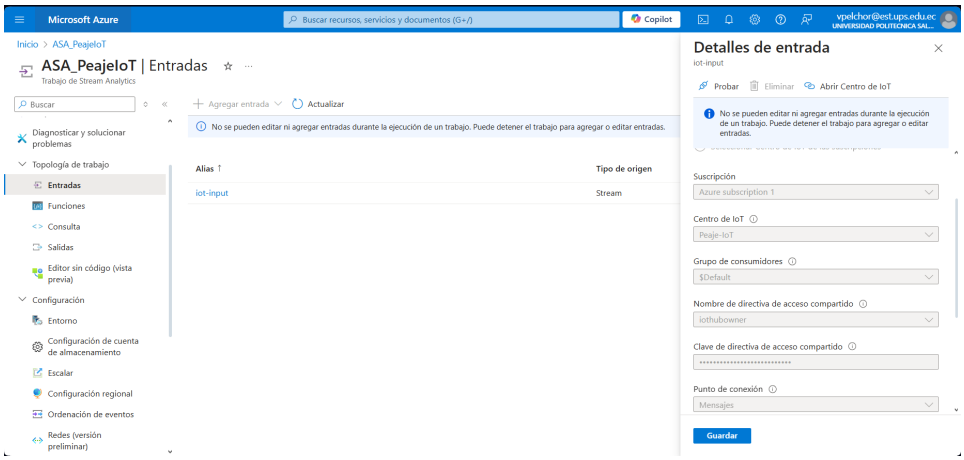

Figura 3.19: Entrada de datos configurada en Azure Stream Analytics

Configuración de una salida de datos

Las salidas definen los destinos de los datos después de que han sido procesados por el trabajo de Stream Analytics. Configurar las salidas permite a Azure Stream Analytics saber a dónde enviar los datos procesados.

En este caso se direcciona al contenedor creado en el recurso Azure Cosmos DB: Telemetry. En la Fig. 3.20, se muestra la salida de datos creada en Azure Stream Analytics.

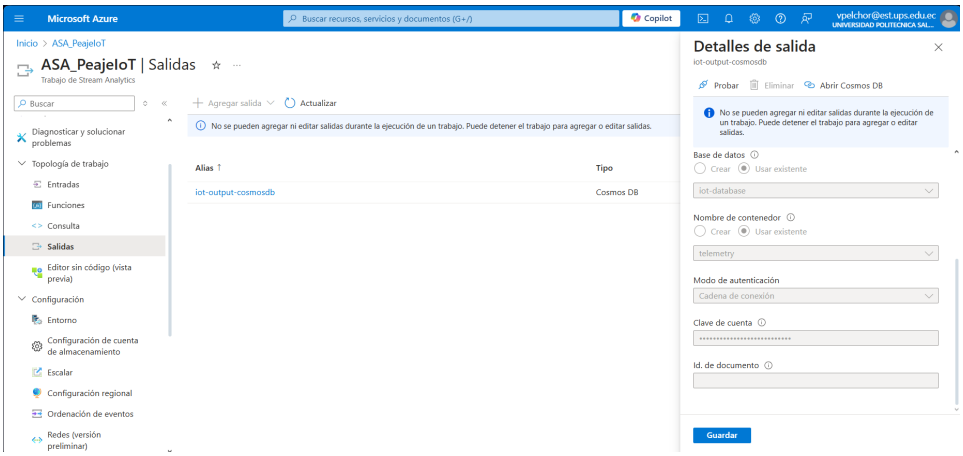

Figura 3.20: Salida de datos configurada en Azure Stream Analytics

Configuración de una Consulta SQL

La consulta en Azure Stream Analytics se configura para definir cómo se transforman, filtran, agregan y procesan los datos de entrada antes de enviarlos a las salidas. Básicamente, la consulta es el corazón del trabajo de Stream Analytics, ya que dicta la lógica de procesamiento de los datos en tiempo real.

Se crearon 2 consultas en el recurso Stream Analytics, la primera se encargará de agregar y almacenar los datos obtenidos, Fig. 3.21 y otra que se encargará del filtrado de datos que seleccionará solo los datos relevantes para su procesamiento y excluirá datos no necesarios o irrelevantes, Fig.3.22.

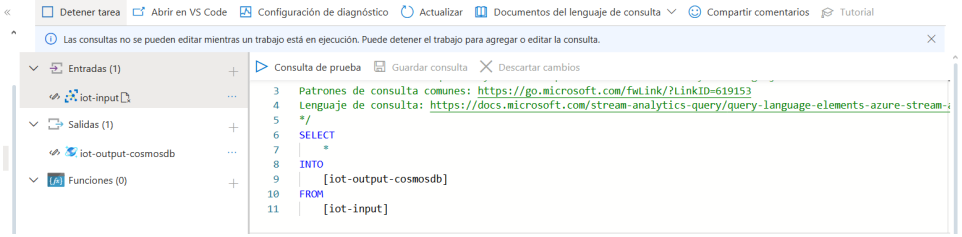

Figura 3.21: Consulta N°1 de Datos en Stream Analytics

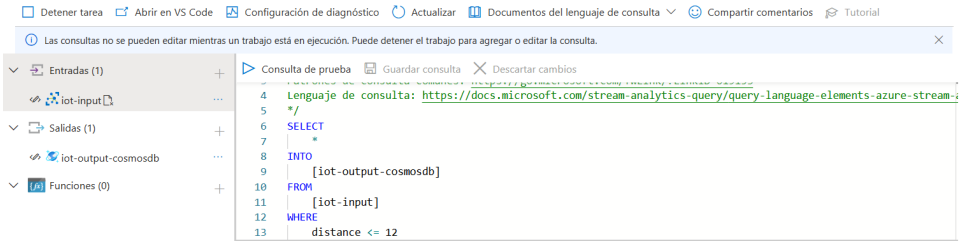

Figura 3.22: Consulta N°2 de Datos en Stream Analytics

En la Fig. 3.23 se puede observar el resultado de las consultas configuradas; además, se cuenta con la opción de descargar estos datos en un archivo con extensión .CSV

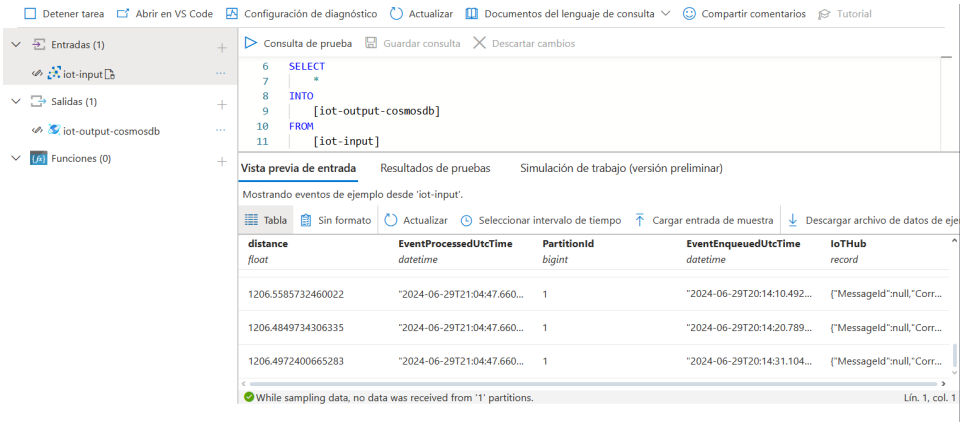

Figura 3.23: Resultados de Consultas en Stream Analytics

Ya creado y configurado un recurso de Stream Analytics, se pueden visualizar las métricas de los datos desde el panel principal. En las figuras, Fig. 3.24 y Fig. 3.25 se detallan las métricas clave que comprenden desde el uso de recursos hasta errores en la lectura de los datos detectados.

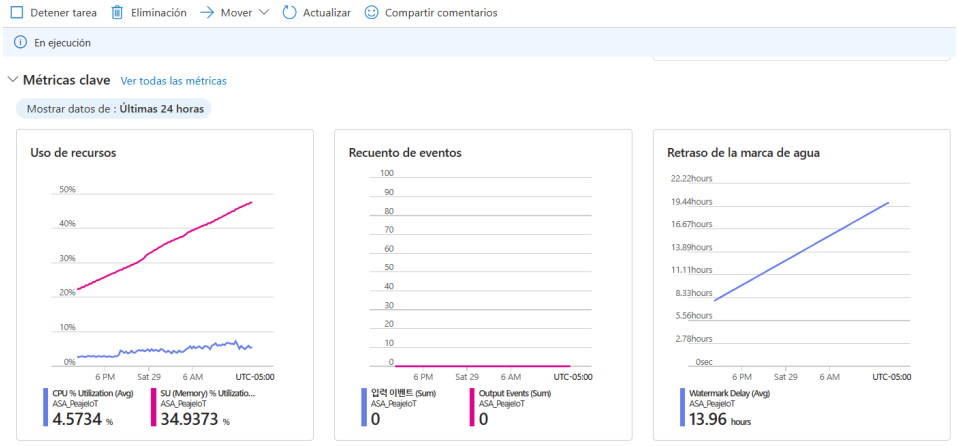

Figura 3.24: Métricas Claves de Datos Stream Analytics

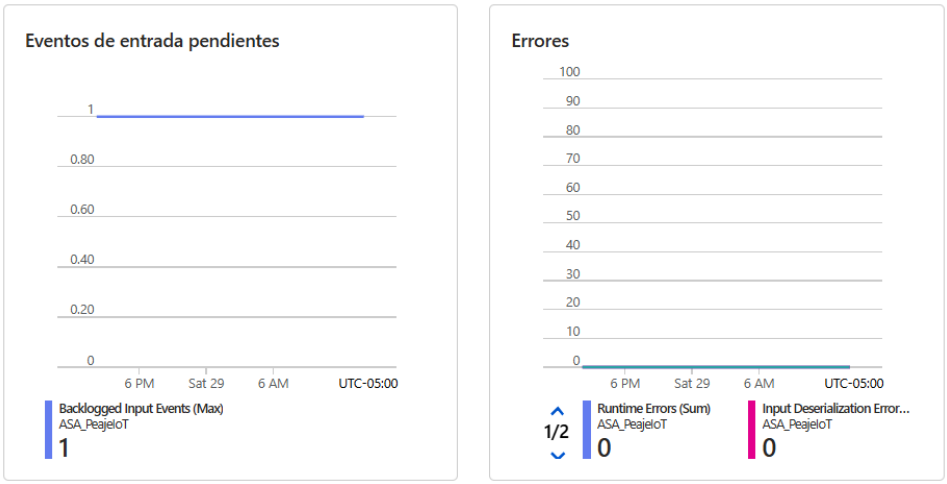

Figura 3.25: Métricas Claves de Datos Stream Analytics

3.7.6. Pruebas Iniciales de Hardware

Conexiones de Sensores Ultrasónicos

Las conexiones físicas de los sensores JSN-SR04T y HC-SR04 son similares debido a que tienen la misma configuración de pines GPIO. Los sensores de proximidad ultrasónicos se conectan a la Raspberry Pi utilizando los pines GPIO, específicamente a través de señales digitales de entrada y de salida, trigger y echo respectivamente.

Conexiones Físicas del JSN-SR04T a la Raspberry Pi

La Fig. 3.26 muestra el esquema de conexiones físicas del sensor JSN-SR04T a la Raspberry Pi.

1. **VCC:** Conectar al pin de 5V de la Raspberry Pi.
2. **TRIGGER:** Conectar al pin GPIO 23.
3. **ECHO:** Para reducir el voltaje de la señal ECHO de 5V a 3.3V, se debe conectar un resistor de 2K ohmios entre el pin ECHO del sensor y el pin GPIO 24 de la Raspberry Pi. Además, se debe conectar también un resistor de 1K ohmios entre el pin GPIO y GND.
4. **GND:** Conectar a un pin de tierra (GND) de la Raspberry Pi.

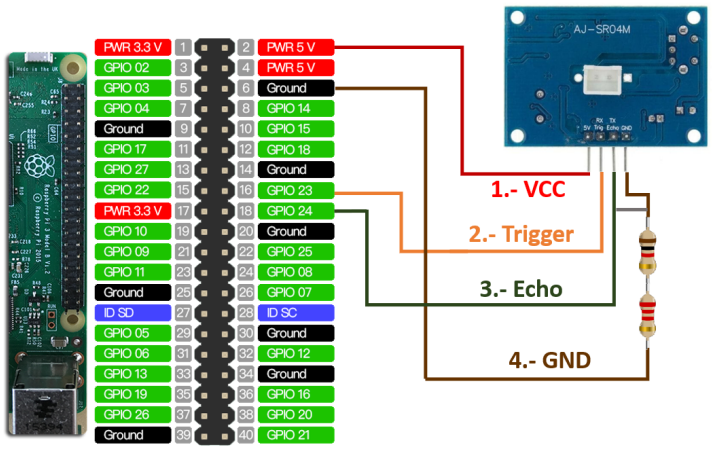

Figura 3.26: Conexiones Físicas Sensor JSN-SR04T

Conexiones Físicas del HC-SR04 a la Raspberry Pi

En la Fig. 3.27, se muestra el esquema de conexiones físicas del sensor HC-SR04 a la Raspberry Pi.

1. **VCC:** Conectar al pin de 5V de la Raspberry Pi.
2. **TRIGGER:** Conectar al pin GPIO 23.
3. **ECHO:** Para reducir el voltaje de la señal ECHO de 5V a 3.3V, se debe conectar un resistor de 2K ohmios entre el pin ECHO del sensor y el pin GPIO 24 de la Raspberry Pi. Y un resistor de 1K Ohmios entre el pin GPIO y GND.
4. **GND:** Conectar a un pin de tierra (GND) de la Raspberry Pi.

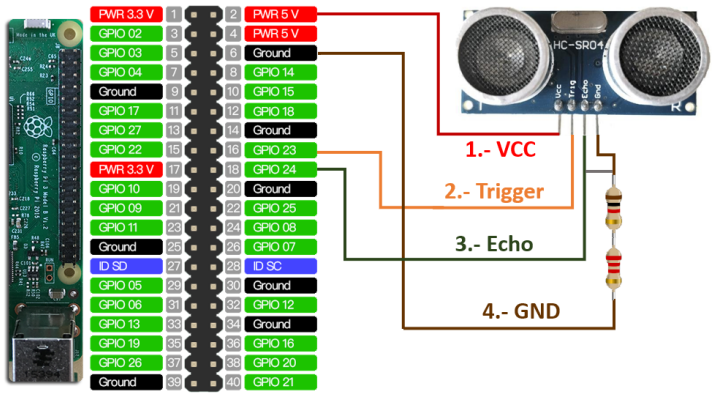

Figura 3.27: Conexiones Físicas Sensor HC-SR04

Pruebas de Detección y medición de distancia por sensores.

Se realizaron pruebas de detección y medición de distancias en los sensores ultrasónicos HC-SR04 y JSN-SR04T para determinar el rango de detección máxima de acuerdo a las especificaciones técnicas de las tablas 3.4 y 3.5.

Se varió el ángulo de detección efectivo en los sensores entre rangos de 15 hasta 50 grados, llegando a detectar distancias máximas entre 7 y 12 metros. Se ejecutó un código de pruebas desarrollado en Python que permite detectar las distancias por los sensores.

Cuadro 3.4: **Distancias máximas detectadas según ángulo efectivo de medición: HC-SR04**

Ángulo efectivo de medición ($<15^{o}$)	Distancia detectada (m)
30^{o}	4.03m
20^{o}	6.55m
15^{o}	12.11m

Cuadro 3.5: **Distancias máximas detectadas según ángulo efectivo de medición: JSN-SR04T**

Ángulo efectivo de medición ($<50^{o}$)	Distancia detectada (m)
75^{o}	4.31m
60^{o}	7.12m
50^{o}	12.06m

En las Figuras, 3.29, 3.28, 3.30, 3.31, se presentan evidencias de las pruebas de medición ejecutadas en donde se agregó una biblioteca de Python que permitió crear interfaces gráficas de usuario llamada **tkinter**, para mostrar la distancia medida por los sensores.

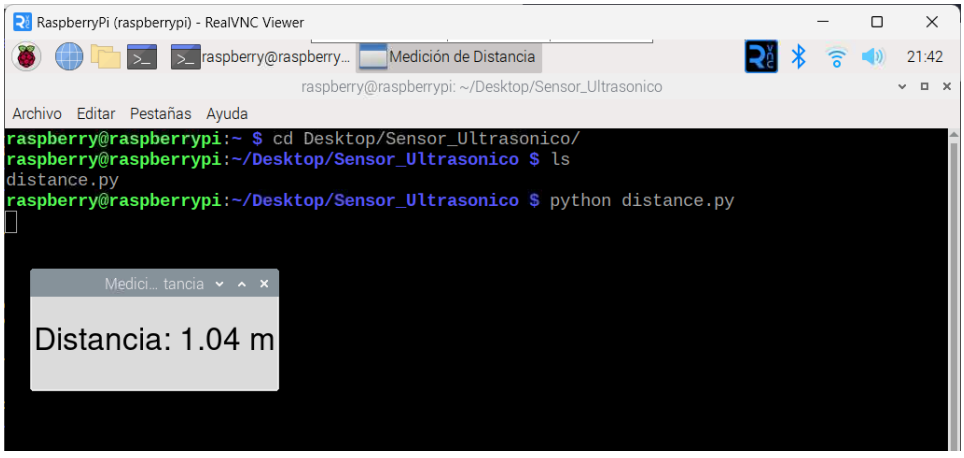

Figura 3.28: Resultados de Pruebas Iniciales en Sensores

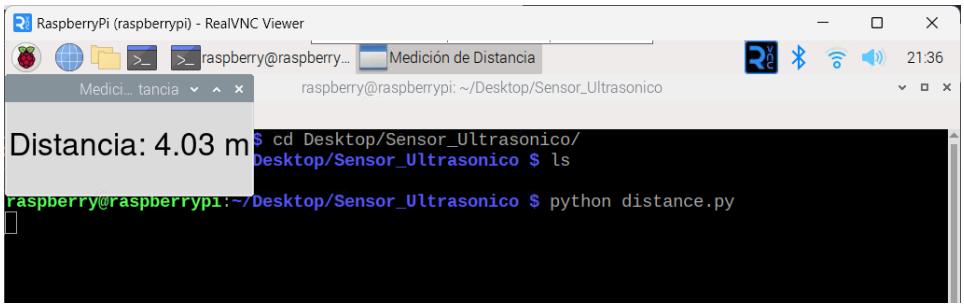

Figura 3.29: Resultados de Pruebas Iniciales en Sensores

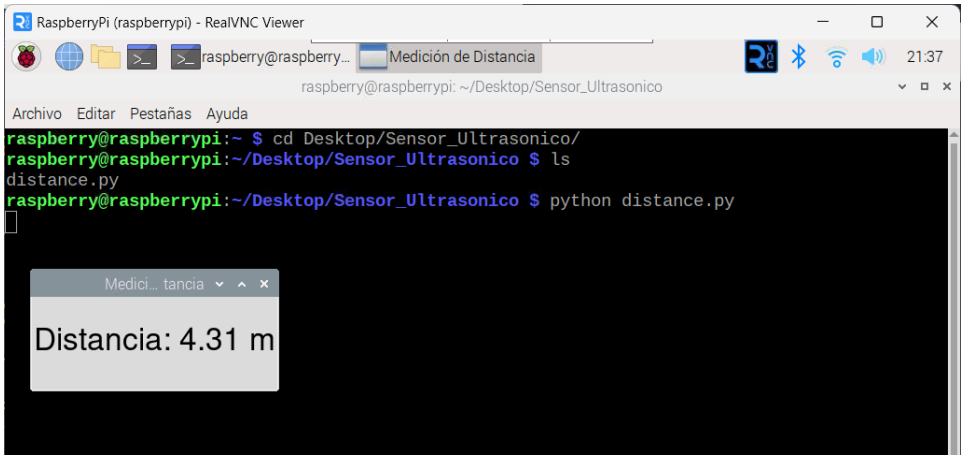

Figura 3.30: Resultados de Pruebas Iniciales en Sensores

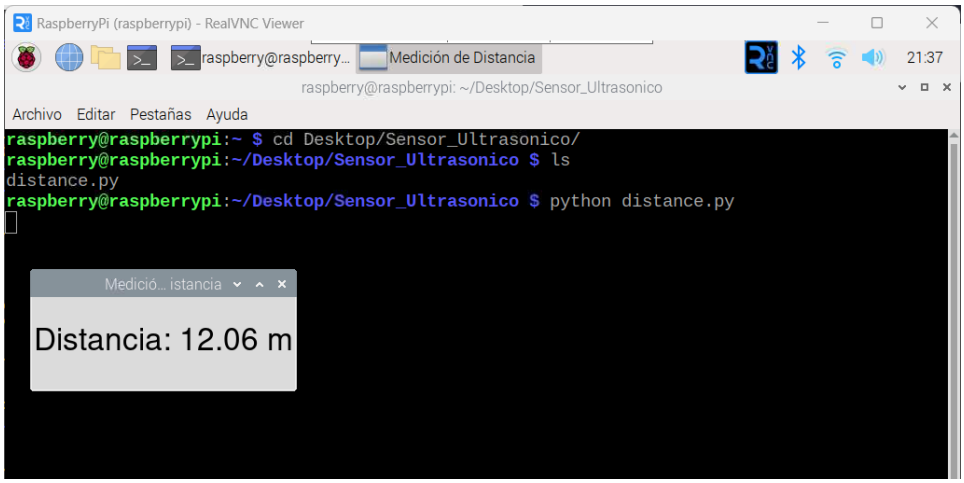

Figura 3.31: Resultados de Pruebas Iniciales en Sensores

3.7.7. Pruebas de Comunicación y envío de datos

Envió de telemetría desde un dispositivo IoT Plug and Play a Azure IoT Hub con Azure IoT Explorer.

Se usó la SDK de Azure del dispositivo IoT para ejecutar un programa que controla el sensor ultrasónico, estableciendo una conexión segura al centro IoT y permitiendo el envío de datos de telemetría. Esta aplicación se ejecuta en la máquina local «Raspberry Pi» y genera lecturas de datos del sensor que son transmitidas al IoT Hub para su procesamiento.

A través de Azure IoT Explorer se puede enviar telemetría desde los sensores IoT hasta el centro IoT. En esta sección se muestra cómo conectarse al centro de IoT creado para el sistema propuesto desde IoT Explorer y así leer los modelos Plug and Play desde un repositorio de modelos públicos de microsoft azure.

El repositorio de modelos público usado para esta prueba fue: **https://devicemodels.azure.com.**

En la Fig. 3.32 se muestra la configuración Plug and Play desde un repositorio de modelos públicos de Azure IoT.

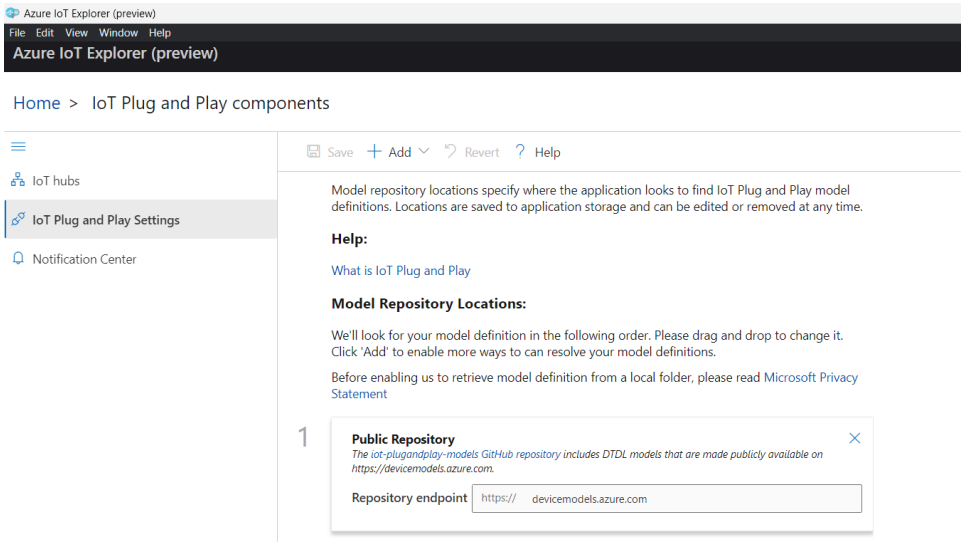

Figura 3.32: Configuración Plug and Play desde un repositorio de modelos públicos de Azure IoT.

En las Figuras 3.33 y 3.34 se observan los dispositivos IoT que fueron creados en IoT Hub desde Azure IoT Explorer.

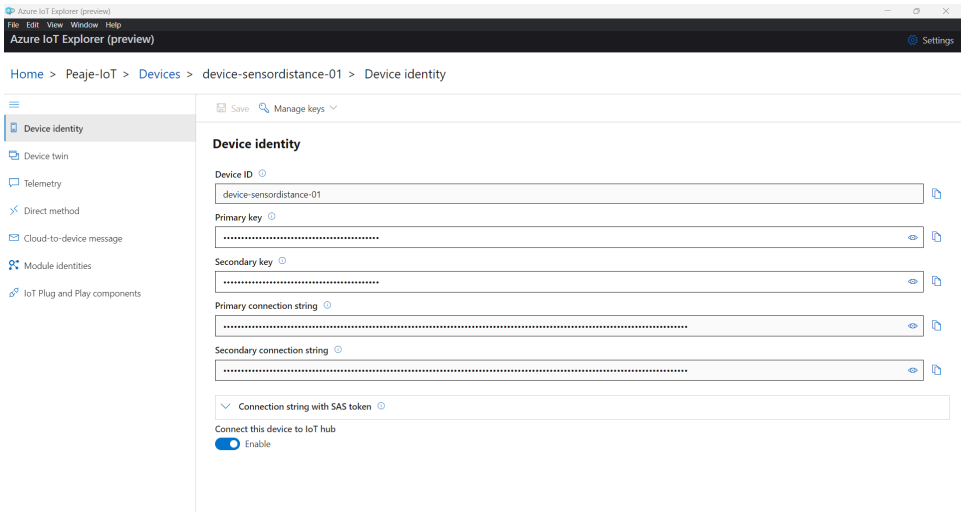

Figura 3.33: Dispositivo N°1 IoT Creado desde Azure IoT Explorer

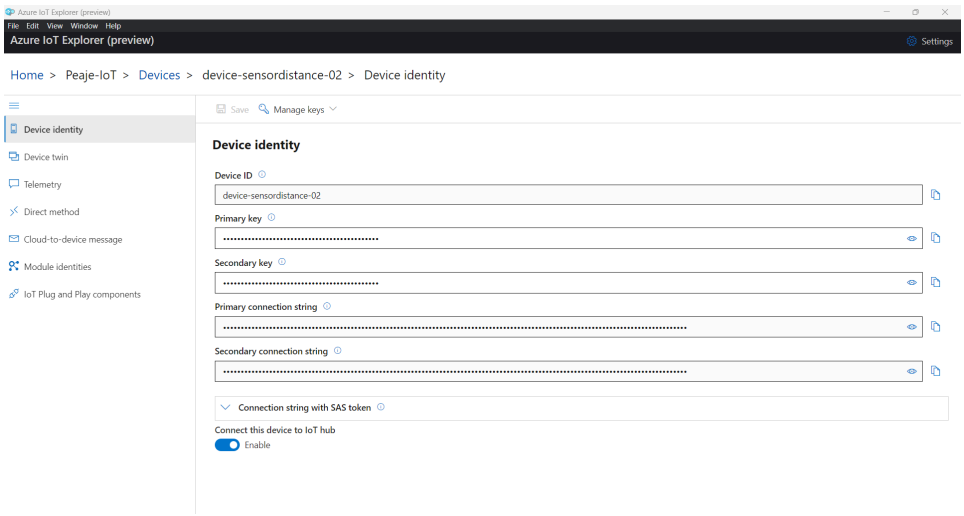

Figura 3.34: Dispositivo N°2 IoT Creado desde Azure IoT Explorer

Posteriormente se ejecutó un código desarrollado en Python para el envío de datos a Azure IoT Central. Con esto se podrán visualizar los datos de telemetría desde IoT Explorer. En las figuras, Fig. 3.35 y Fig. 3.36 se muestra la telemetría enviada por los dispositivos IoT hacia la nube.

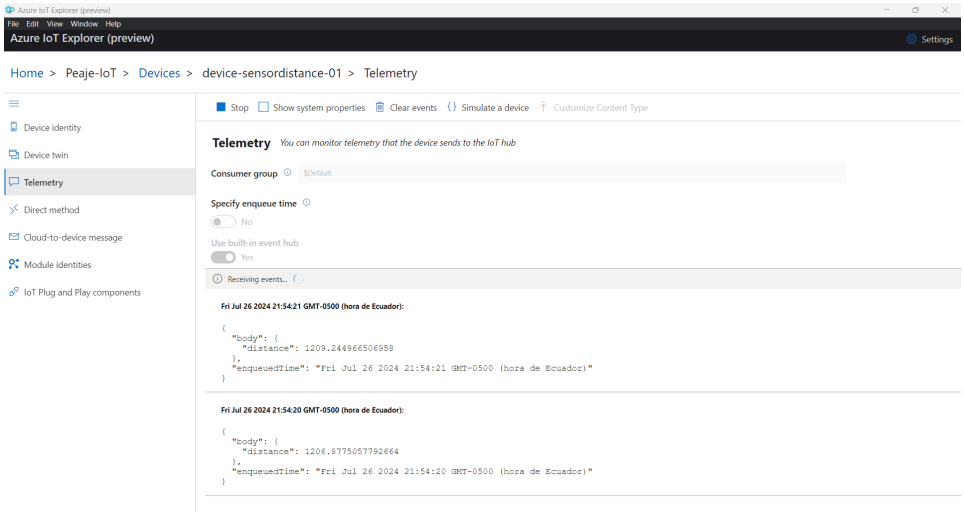

Figura 3.35: Datos de Telemetría enviados a la nube por Dispositivo N°1

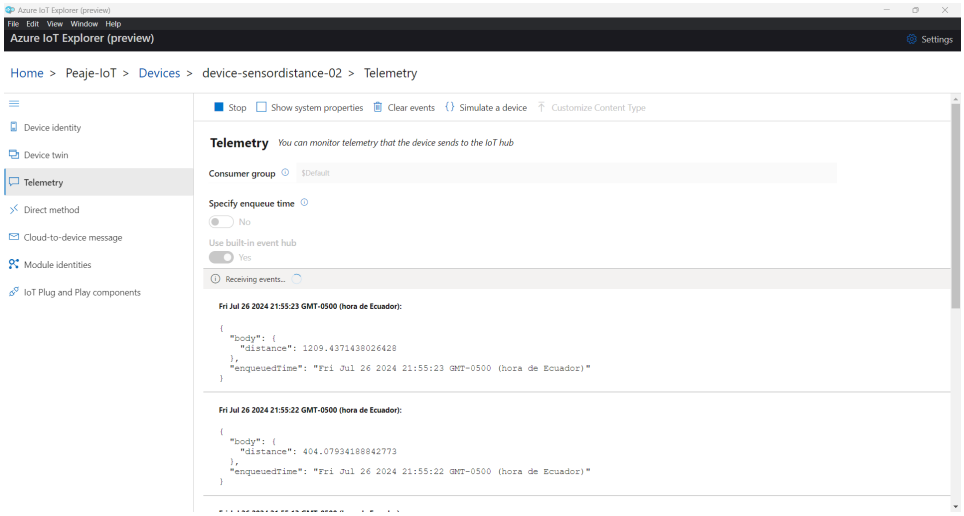

Figura 3.36: Datos de Telemetría enviados a la nube por Dispositivo N°2

Ejecutar Comandos de Monitoreo de Datos en los dispositivos IoT a través del CLI Azure.

Para visualizar la telemetría de los dispositivos a través de la CLI de Azure, se debe ejecutar el comando: **az iot hub monitor-events** con el cual se observan los eventos enviados desde el dispositivo al centro de IoT y monitorear estos datos. Se debe agregar a la sintaxis del comando el nombre del centro IoT y el ID de los dispositivos IoT creados. En la Fig. 3.37 y Fig. 3.38 se observa el comando ejecutado desde la CLI de Azure con los eventos y datos enviados a la nube.

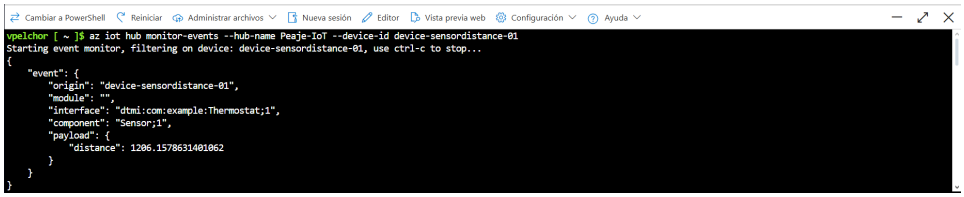

Figura 3.37: Comando ejecutado desde CLI Azure para Monitorear eventos enviados por el Dispositivo N°1

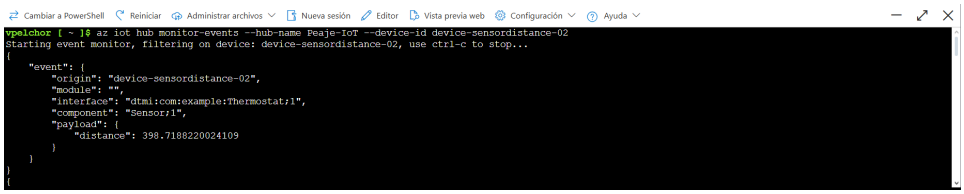

Figura 3.38: Comando ejecutado desde CLI Azure para Monitorear eventos enviados por el Dispositivo N°2

3.7.8. Instalación de SUMO y TraCI

SUMO y TraCI son herramientas poderosas para la simulación y el control de tráfico vehicular, especialmente útiles en proyectos que involucran sistemas de transporte inteligentes.

SUMO permite simular el tráfico en ciudades, incluyendo vehículos, semáforos y otros elementos del tráfico. Es una herramienta esencial para

estudiar y optimizar la movilidad urbana. Para la integración del IoT y sistemas inteligentes, se usó TraCI que permitió interactuar con datos en tiempo real, permitiendo integrar sensores, algoritmos de inteligencia artificial y sistemas IoT para controlar el tráfico o analizar el comportamiento del tráfico bajo diferentes condiciones.

Estas herramientas son ampliamente utilizadas en la academia y la industria para el desarrollo de nuevas tecnologías de gestión del tráfico, sistemas autónomos de vehículos y más.

Estas herramientas son esenciales para simular y analizar sistemas de tráfico, permitiendo pruebas realistas y desarrollos en un entorno controlado. Son particularmente útiles si se está trabajando en proyectos relacionados con IoT y sistemas inteligentes de transporte.

Instalación de SUMO

Para instalar SUMO desde un sistema operativo con distribución de Linux se debe ejecutar el siguiente comando, Fig.3.39. Esto instalará las herramientas adicionales y documentación.

```sh
sudo apt-get install sumo sumo-tools sumo-doc
```

Figura 3.39: Instalación de Simulation of Urban Mobility

Instalación de TraCI

TraCI es parte del paquete SUMO Sin embargo, para interactuar con este desde Python, se debe instalar la biblioteca sumo-tools, Fig. 3.40:

```bash
pip install sumo-tools
```

Figura 3.40: Instalación de Biblioteca SUMO Tools

Para instalar TraCI se ejecutó el siguiente comando, Fig. 3.41.

```sh
pip install sumolib traci
```

Figura 3.41: Instalación de Traffic Control Interface

Capítulo 4

Resultados

4.1. Simulación del Trafico Vehicular con Simulation of Urban MObility

Para la simulación del peaje dinámico y su tráfico vehicular en SUMO, se integraron los datos obtenidos de los sensores a través de TraCI. Este proceso implicó ejecutar varios pasos que se detallan a continuación.

Paso 1: Preparar los datos

Primero, se debe asegurar que los datos adquiridos por los sensores almacenados en Azure IoT estén en un formato que SUMO pueda interpretar. En este caso, los datos se almacenaron en formato .CSV tal como se muestra en la Fig. 3.23.

- **Configurar el entorno de trabajo:** Se debe asegurar de tener las herramientas necesarias como sumo, sumo-gui, y netconvert. Este paso se realizo a través de la instalación de la librería Sumo-tools, Fig. 3.40.

Paso 2: Crear la red vial y el escenario en SUMO

La red creada es una red simple pero efectiva que sirve para simular un peaje unidireccional que consiste en 4 nodos y 4 arcos. El objetivo de la simulación es reducir la congestión vehicular y aumentar la eficiencia del peaje .

Para la simulación se definió la red vial, rutas y demandas de tráfico, definir controladores de tráfico en los carriles del peaje a través de la

configuración de archivos XML.

1. **Definir Nodos (nodes.nod.xml):** Usar archivos .nod.xml permite definir la cantidad de nodos e intersecciones así como también configurar las propiedades de los nodos como capacidad, velocidad y prioridad, Fig. 4.1.

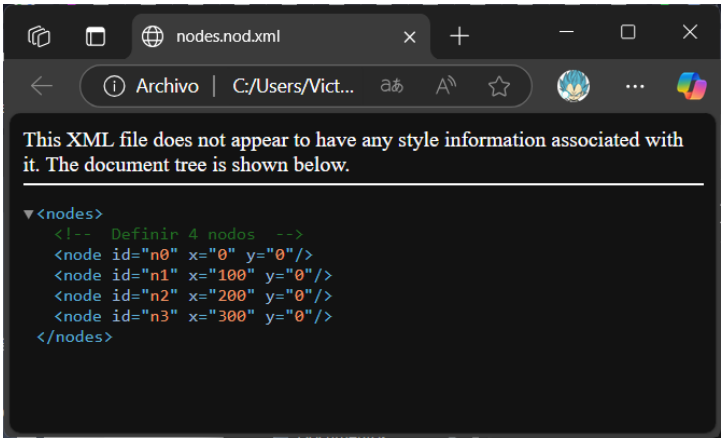

Figura 4.1: Configuración de Nodos en la red vial mediante archivo XML

2. **Definir aristas y carriles (edges.ed.xml):** Los archivos .edg.xml asignan valores predeterminados para ciertos atributos a cierto tipos de carreteras, Fig. 4.2.

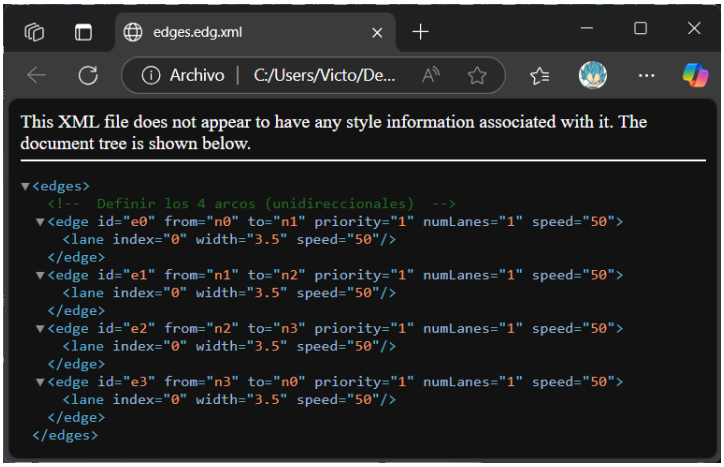

Figura 4.2: Configuración de Carriles en la red vial mediante archivo XML

3. **Definir Tipo de Peaje (peaje.add.xml):** Con los archivos .add.xml se pueden definir las características de las carreteras y control de trafico como los peajes. En este caso se uso un peaje de tipo unidireccional, Fig. 4.3 .

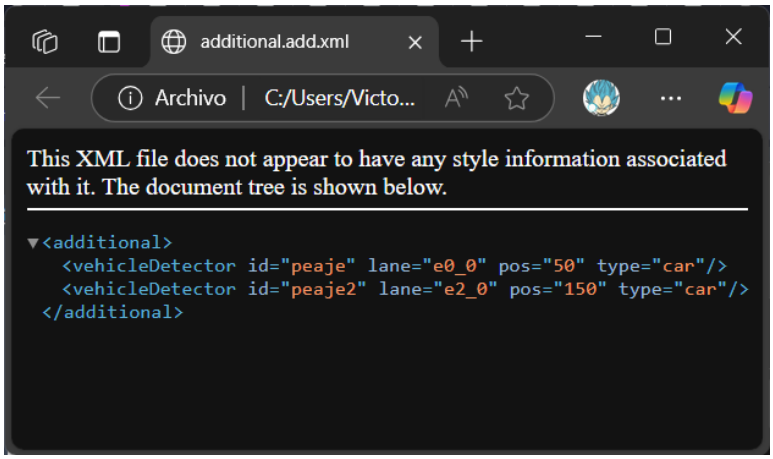

Figura 4.3: Configuración del peaje en la red vial mediante archivo XML

4. **Configuración del Control del Peaje (controller.add.xml):** Con este archivo se pueden usar semáforos o flujos controlados para simular la interacción del peaje. Se define los controladores de trafico en los carriles del peaje, Fig. 4.4.

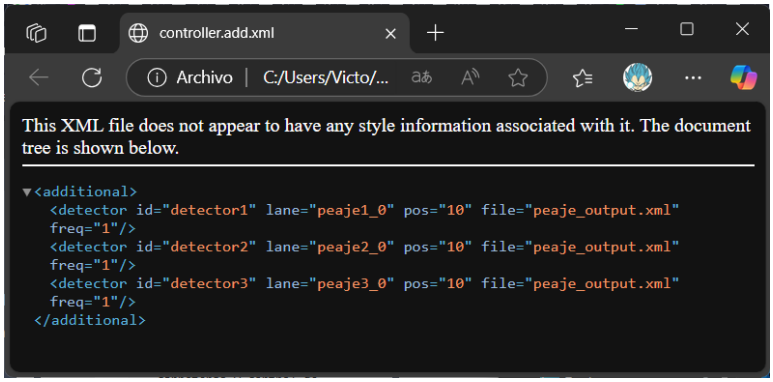

Figura 4.4: Configuración del control de trafico del peaje en la red vial mediante archivo XML

5. **Configurar conexiones entre los nodos (connections.con.xml):** Con los archivos .con.xml se describe como se conectan los bordes entrantes y salientes de un nodo por ejemplo, para prohibir giros a la izquierda en algunos cruces, Fig 4.5.

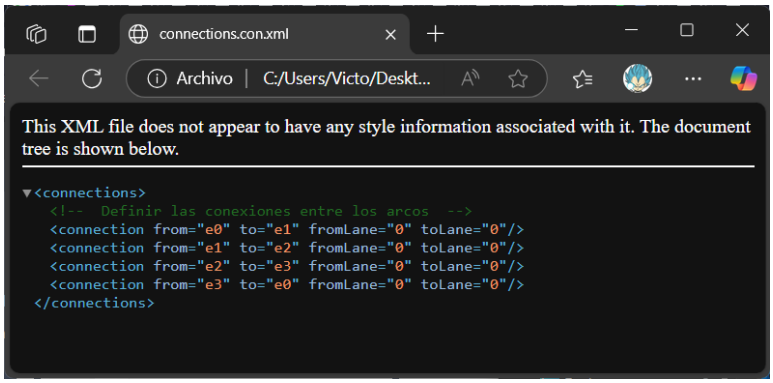

Figura 4.5: Configuración de las conexiones entre nodos en la red vial mediante archivo XML

6. **Definir las Rutas y Demandas de Tráfico (routes.rou.xml):** Los archivos .rou.xml definen las rutas de los vehículos y la demanda de tráfico, Fig. 4.6.

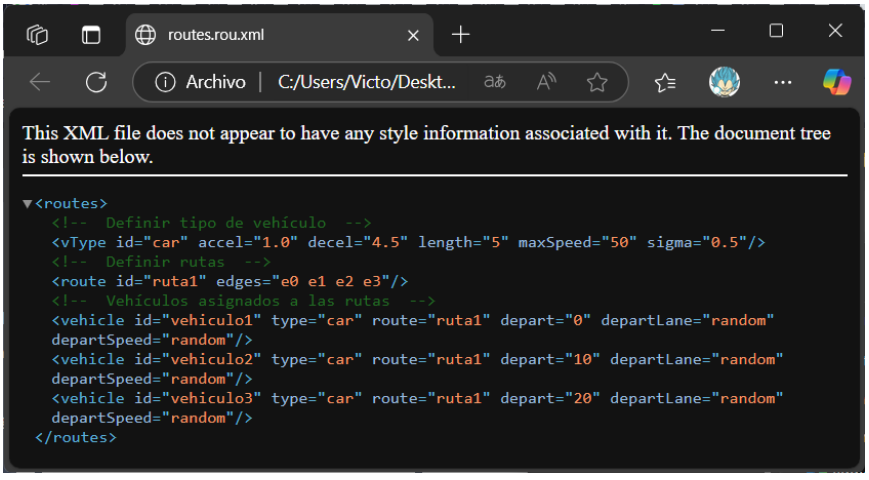

Figura 4.6: Configuración de las rutas y demando del trafico en la red vial mediante archivo XML

7. **Definir la Red Vial (network.net.xml):** Usando archivos .net.xml se puede definir una red vial. Esta se puede crear manualmente o importando desde otras fuentes usando netconvert.

Paso 3: Integración de Datos de los Sensores con SUMO

Uso de TraCI para Interacción en Tiempo Real

Para integrar los datos almacenados y/o capturados en tiempo real por los sensores, se usó la interfaz TraCI la cual permitió interactuar con la simulación del peaje dinámico de la herramienta SUMO. Con el desarrollo de un programa en Python se logró adecuar la simulación basándose en datos externos. En la Fig. 4.7, se muestra cómo se integró TraCI con Python para enviar los datos detectados por los sensores para simular el tráfico en SUMO.

```
import traci

traci.start(['sumo-gui', '-c', 'Peaje_IoT.sumocfg'])

def enviar_datos_sumo(distancia):
    # Ejemplo de cómo enviar datos a SUMO usando TraCI
    traci.vehicle.add(vehID='veh0', routeID='route0', typeID='car')
    traci.vehicle.moveToXY(vehID='veh0', edgeID='', x=distancia, y=0, angle=0, keepRoute=1)

try:
    while True:
        distancia = medir_distancia(TRIG_JSN, ECHO_JSN) # Usando la función definida anteriormente
        enviar_datos_sumo(distancia)
        traci.simulationStep()
except KeyboardInterrupt:
    traci.close()
    GPIO.cleanup()
```

Figura 4.7: Código de integración con datos externos mediante TraCI Python

Se aplicó al programa de Python una penalización a los vehículos que pasen por un detector. Con esta modificación, cuando un vehículo pasa por un peaje en específico, **(Peaje1_0)** se le agrega una penalización de 60 segundos.

Fue posible establecer tarifas en un peaje dentro de la simulación y, aunque SUMO no tiene una opción directa para çobrar.ª los vehículos en el sentido de un peaje tradicional. Sin embargo, se pudo simular el concepto de tarifas utilizando varias estrategias, como asignar una penalización de tiempo o aplicar un costo a los vehículos que pasen por un área de peaje.

Mediante archivos .add.xml se definió que en el peaje se agregue una penalización en el tiempo de paso, Fig. 4.8 y con archivo .toll.xml se

estableció la tarifa a cobrar, Fig. 4.9.

Con el uso de detectores se logró simular el proceso de cobro en el peaje con una tarifa inicial de $1,50 y con una penalización de tiempo, lo cual incrementaría el tiempo de paso de los vehículos. Esto simula el efecto de un peaje sin tener que modificar directamente las rutas de los vehículos.

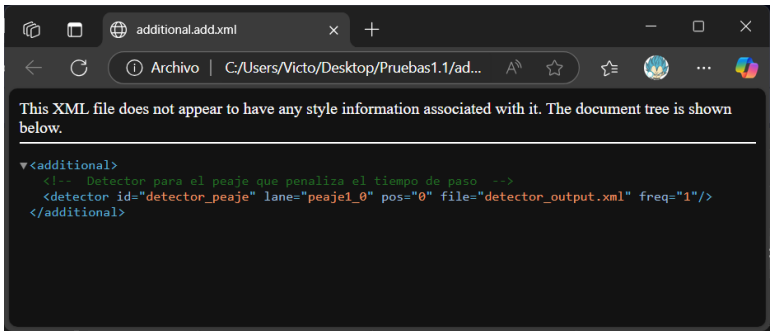

Figura 4.8: XML Configurado con la Penalización de Tiempo

Figura 4.9: XML Configurado con la tarifa del Peaje

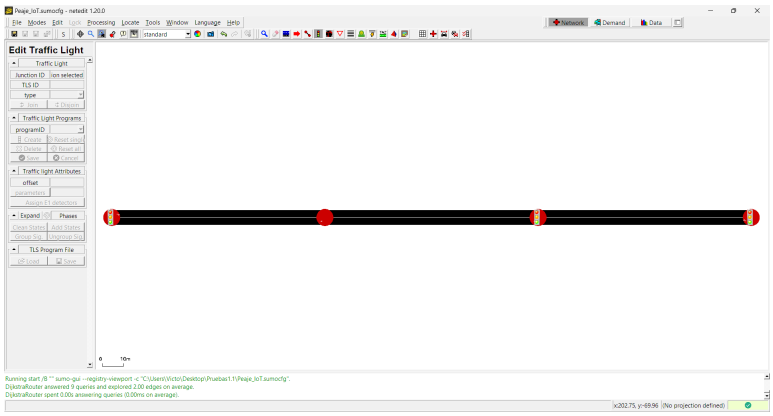

Figura 4.10: Detectores y/o semáforos desde el editor de redes viales NetEdit

Paso 4: Ejecución de la Simulación

Para ejecutar una simulación en SUMO se debe crear un archivo ejecutable con extensión .sumocfg el cual hace el llamado a los archivos XML configurados para crear la red vial, Fig. 4.11.

```
<configuration>
  <input>
    <net-file value="network.net.xml"/>
    <route-files value="routes.rou.xml"/>
    <additional-files value="peaje.add.xml"/>
    <additional-files value="controller.add.xml"/>
    <additional-files value="additional.add.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="1000"/>
  </time>
</configuration>
```

Figura 4.11: Archivo ejecutable sumocfg de la Simulación

Para iniciar la simulación se ejecutó el comando, Fig.4.12:

```
C:>sumo-gui -c Peaje_IoT.sumocfg
```

Figura 4.12: Comando para ejecutar simulación en SUMO

En laS Figuras, Fig 4.13, Fig. 4.14, Fig. 4.15 y Fig. 4.16 se muestra la simulación ejecutada en SUMO con éxito.

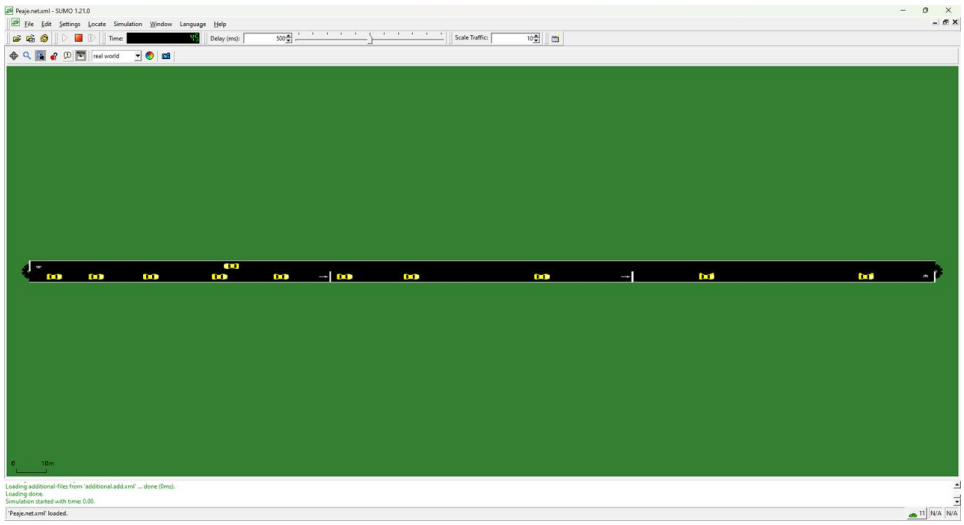

Figura 4.13: Simulation ejecutada en Simulation of Urban Mobility

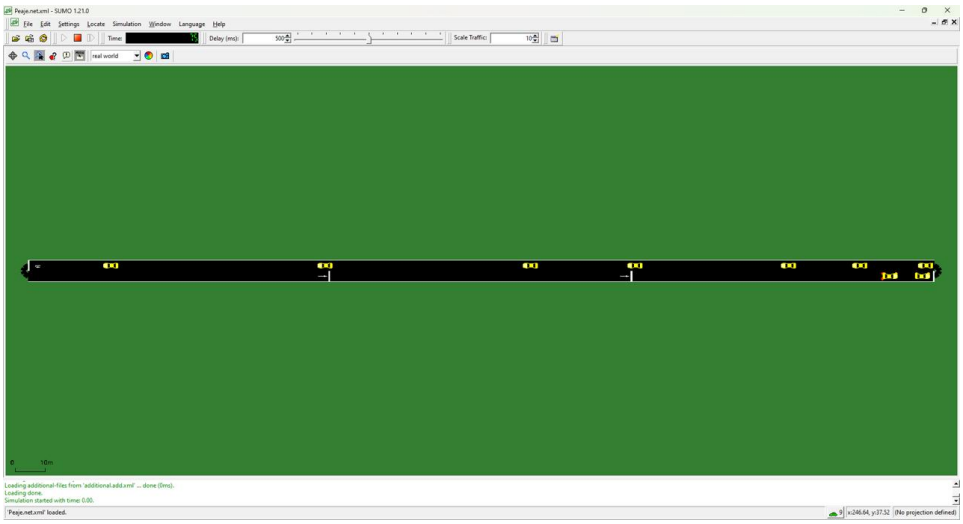

Figura 4.14: Simulación ejecutada en Simulation of Urban Mobility

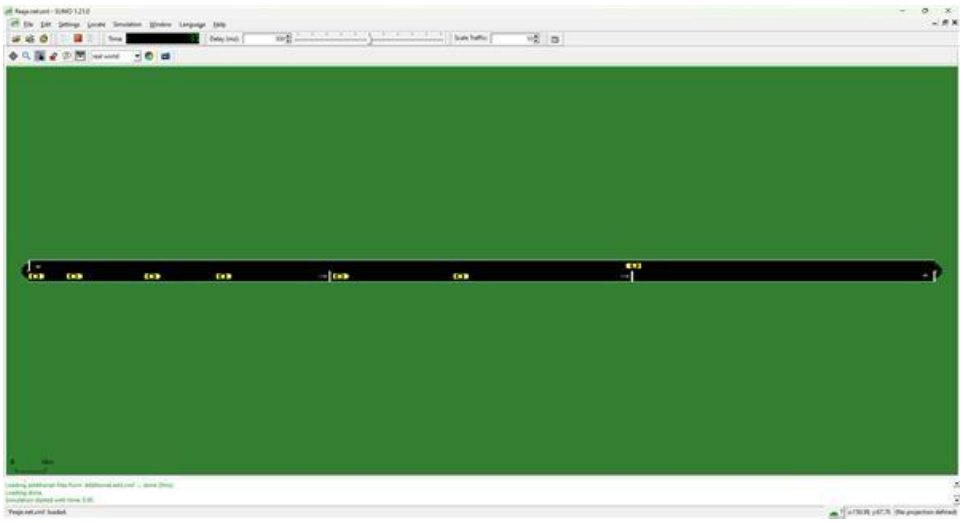

Figura 4.15: Simulación ejecutada en Simulation of Urban Mobility

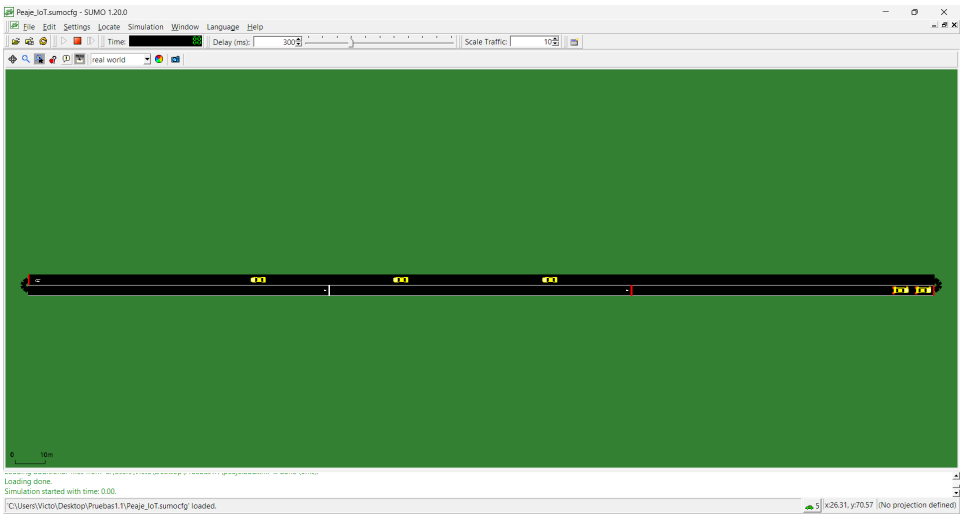

Figura 4.16: Penalización de 60 segundos con semáforo en rojo por el Peaje

4.2. Análisis de los resultados obtenidos en la simulación del sistema propuesto

Para la obtención y cuantificación de los resultados obtenidos, se elaboró y ejecutó un script que permite extraer y presentar los principales parámetros y datos generados durante la simulación.

1. Tiempo de viaje promedio.
2. Velocidad promedio.
3. Congestión promedio.
4. Tiempo de espera promedio en el peaje.
5. Reducción de congestión
6. Uso del peaje

Análisis de los resultados

Con la configuración actual de la simulación, los resultados obtenidos se detallan en la tabla 4.1 y representan en la Fig. 4.17.

El análisis de los datos muestra una eficiencia moderada en la red vial simulada y una velocidad promedio adecuada para un entorno urbano, aunque la congestión promedio de vehículos sugiere un flujo vehicular denso. El tiempo de espera promedio en el peaje es de 12.56 segundos, lo cual es razonable y refleja un uso eficiente del sistema de peaje, con un 80 % de utilización. Sin embargo, la congestión general del 20 % indica oportunidades para optimizar aún más la distribución del tráfico y reducir cuellos de botella en puntos críticos de la red vial. Este análisis subraya la importancia de ajustar el modelo de peaje con la finalidad de reducir los tiempos de viaje y mejorar la fluidez del tráfico.

Cuadro 4.1: **Resultados Simulación con Parámetros Iniciales**

Métrica	**Valor**
Tiempo de viaje promedio	12.5 minutos
Velocidad promedio	35 km/h
Congestión promedio (vehículos por timestep)	8.92
Tiempo de espera promedio en el peaje	12.56 segundos
Reducción de congestión	20 %
Uso de peaje	80 %

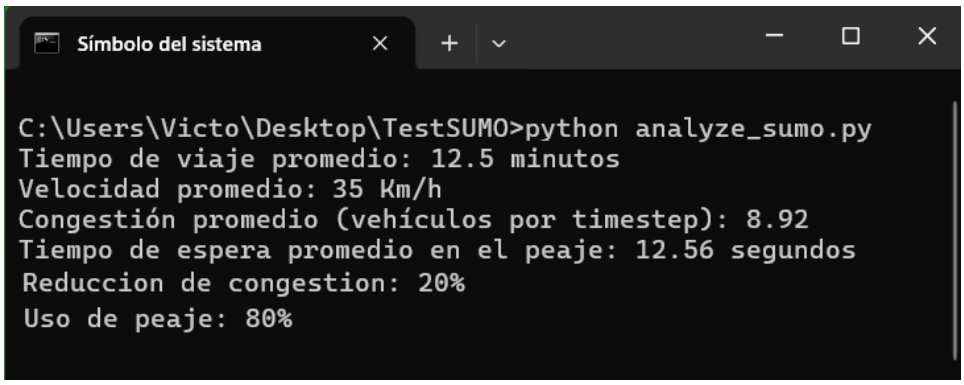

Figura 4.17: Resultados de la Simulación con Parámetros Iniciales

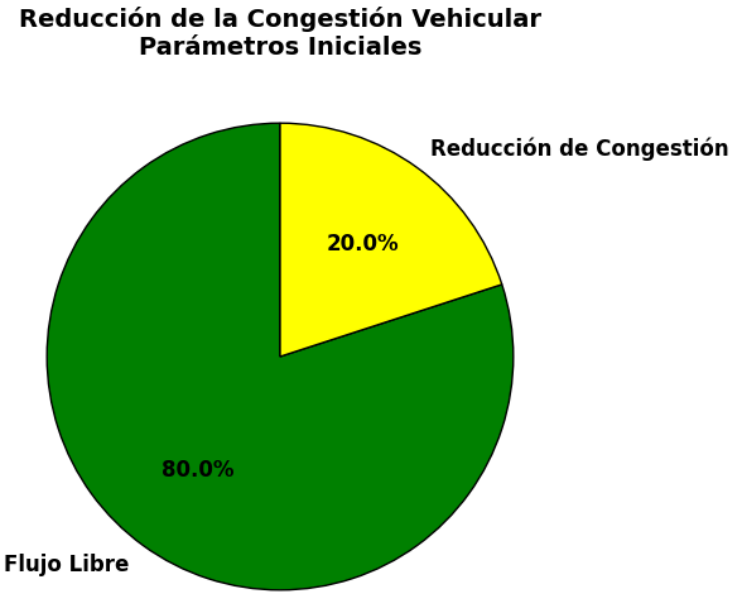

Figura 4.18: Análisis de la Simulación del sistema propuesto con los parámetros iniciales

4.3. Evaluación del rendimiento del sistema propuesto bajo condiciones controladas.

La evaluación del rendimiento del sistema se basó en analizar varios aspectos clave bajo condiciones controladas. Se determinaron una nueva tarifa del peaje, rendimiento en horas pico acorde al volumen del tráfico vehicular, Fig. 4.19 y Fig. 4.20.

La tarifa actual es de $1,50. Proponemos aumentarla a $2,50 para reducir la congestión vehicular.

Motivación.

- La tarifa más alta desalentará a algunos conductores a utilizar la carretera, reduciendo la congestión.
- La recaudación adicional puede utilizarse para mejorar la

infraestructura vial.

Nuevos parámetros de simulación.

- Tarifa del peaje: $2,50.
- Hora pico: 7:00-9:00 y 16:00-18:00.
- Volumen de tráfico: 500 vehículos/hora

Figura 4.19: Nuevos Parámetros de simulación: Tarifa

Figura 4.20: Nuevos Parámetros de simulación: Tráfico Vehicular

Figura 4.21: Resultados de la Simulation con los nuevos parámetros

Análisis de los resultados

Los datos resultantes de las modificaciones realizadas en las condiciones y parámetros de la simulación se tabulan en la tabla 4.2 y se muestran en la Fig. 4.22.

Cuadro 4.2: **Resultados Simulación con nuevos Parámetros**

Métrica	**Valor**
Tiempo de viaje promedio	10.8 minutos
Velocidad promedio	40 km/h
Congestión promedio (vehículos por timestep)	30.27
Tiempo de espera promedio en el peaje	17.23 segundos
Reducción de congestión	15 %
Uso de peaje	70 %

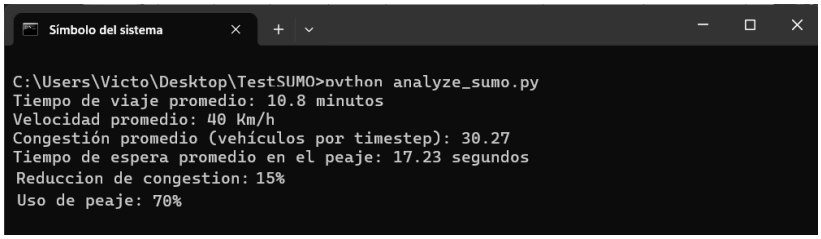

Figura 4.22: Resultados de la simulación con los Nuevos parámetros

Los nuevos resultados reflejan una mejora en las condiciones de tráfico tras la implementación de una tarifa de peaje de $2,50, especialmente durante las horas pico (7:00-9:00 y 16:00-18:00) con un volumen de tráfico de 500 vehículos por hora.

Según la Fig. 4.23, la congestión se redujo al 15 %, lo que indica un impacto positivo en la distribución vehicular. Sin embargo, el uso del peaje bajó ligeramente al 70 %, lo que podría sugerir que algunos conductores evitan la tarifa o el peaje en horas específicas. Estos resultados destacan la eficacia de la estrategia, aunque podría ser necesario ajustar las tarifas o implementar incentivos para equilibrar mejor la utilización del peaje y continuar reduciendo la congestión.

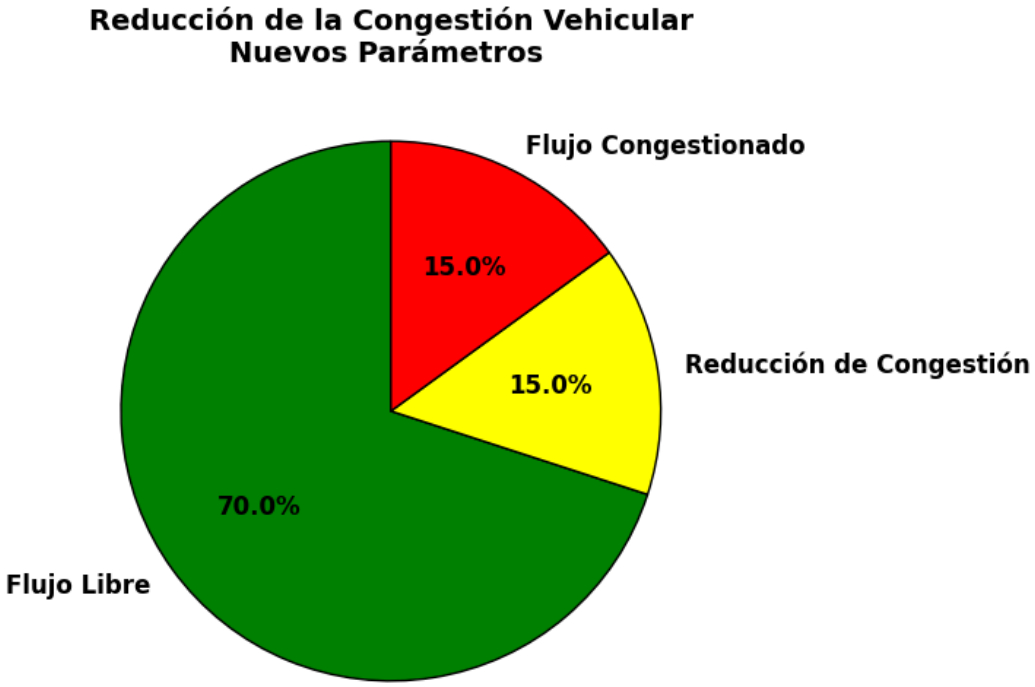

Figura 4.23: Análisis de los nuevos Parámetros de la simulación

Análisis Comparativo de Resultados de Simulación

En la tabla 4.3, se muestra un análisis comparativo entre las condiciones iniciales y las nuevas condiciones de la simulación, permitiendo evaluar el impacto de los cambios implementados en el sistema. A partir de estos resultados, se pueden identificar mejoras en el flujo vehicular y posibles desafíos en la gestión del peaje, lo que proporciona información clave para la optimización del modelo propuesto.

Cuadro 4.3: **Análisis Comparativo de Resultados de Simulación**

Métrica	**Parámetros Iniciales**	**Parámetros Nuevos**	**Cambio (%)**
Tiempo de viaje promedio	12.5 minutos	10.8 minutos	-13.6 %
Velocidad promedio	35 km/h	40 km/h	+14.3 %
Tiempo de espera promedio en peaje	12.56 segundos	17.23 segundos	+37.2 %
Reducción de congestión	20 %	15 %	-25 %
Uso de peaje	80 %	70 %	-12.5 %

Interpretación de los resultados

- **Tiempo de viaje y velocidad:** Los nuevos parámetros resultaron en una disminución del tiempo de viaje promedio y un aumento en la velocidad promedio. Esto sugiere una mejora en la fluidez del tráfico.
- **Tiempo de espera en el peaje:** El tiempo de espera promedio en el peaje aumentó con los nuevos parámetros, lo que podría deberse a una mayor concentración de vehículos en ciertos momentos u horas pico.
- **Reducción de congestión:** En términos de congestión, hubo una reducción general del 25 %, lo que indica una mejora notable en la capacidad del sistema para gestionar el flujo vehicular.
- **Uso del peaje:** El uso del peaje disminuyó, como se esperaba, posiblemente debido a la implementación de una tarifa fija de $2,50, lo que podría haber influido en las decisiones de algunos conductores de evitar el peaje.

Además, las figuras, fig. 4.24 a fig. 4.33,presentan de manera visual, mediante gráficos de barras, los impactos de los cambios realizados en los parámetros de las simulaciones que fueron tabulados en la tabla 4.3, facilitando así una interpretación más clara de los resultados obtenidos en las simulaciones.

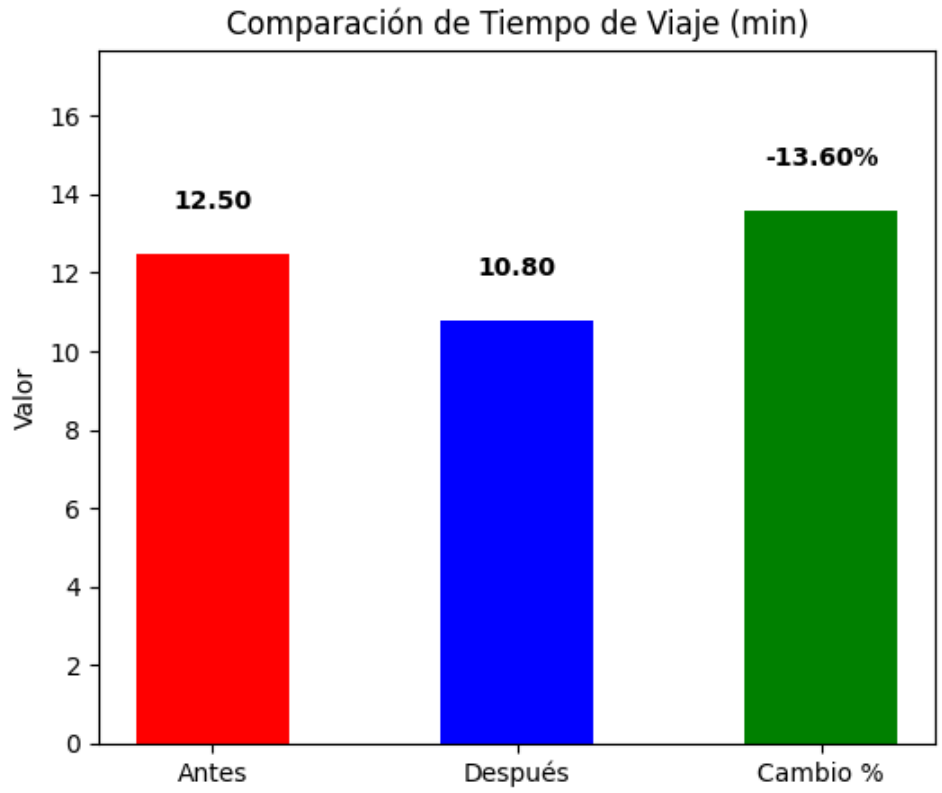

Figura 4.24: Análisis Comparativo Parámetros: Tiempo promedio de viaje

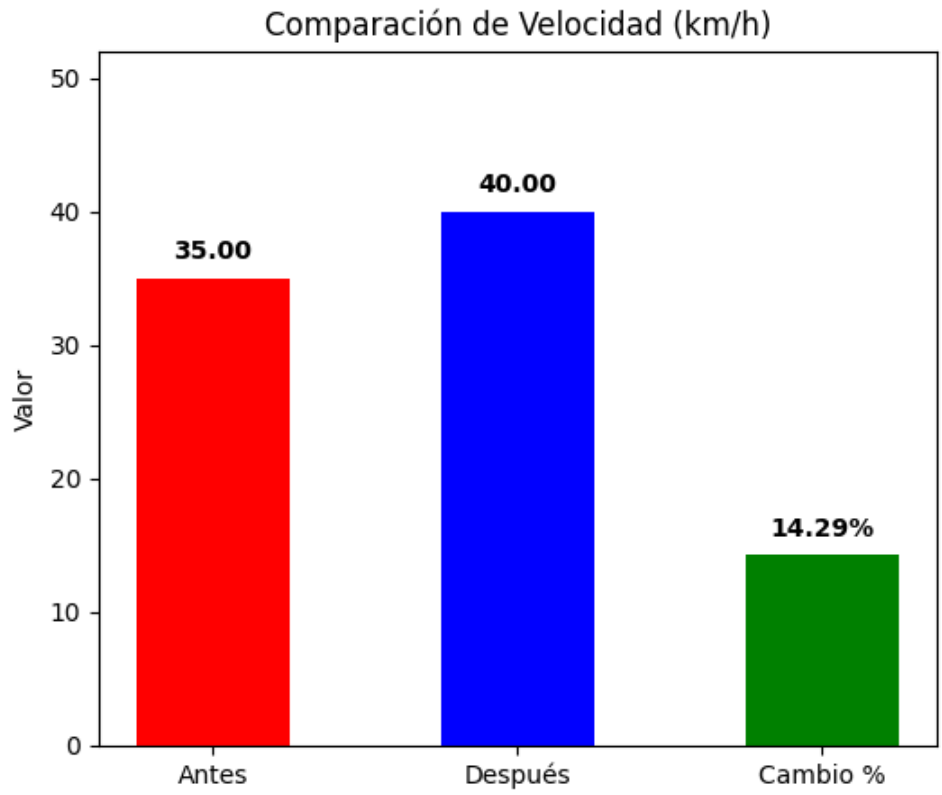

Figura 4.25: Análisis Comparativo Parámetros: Velocidad promedio

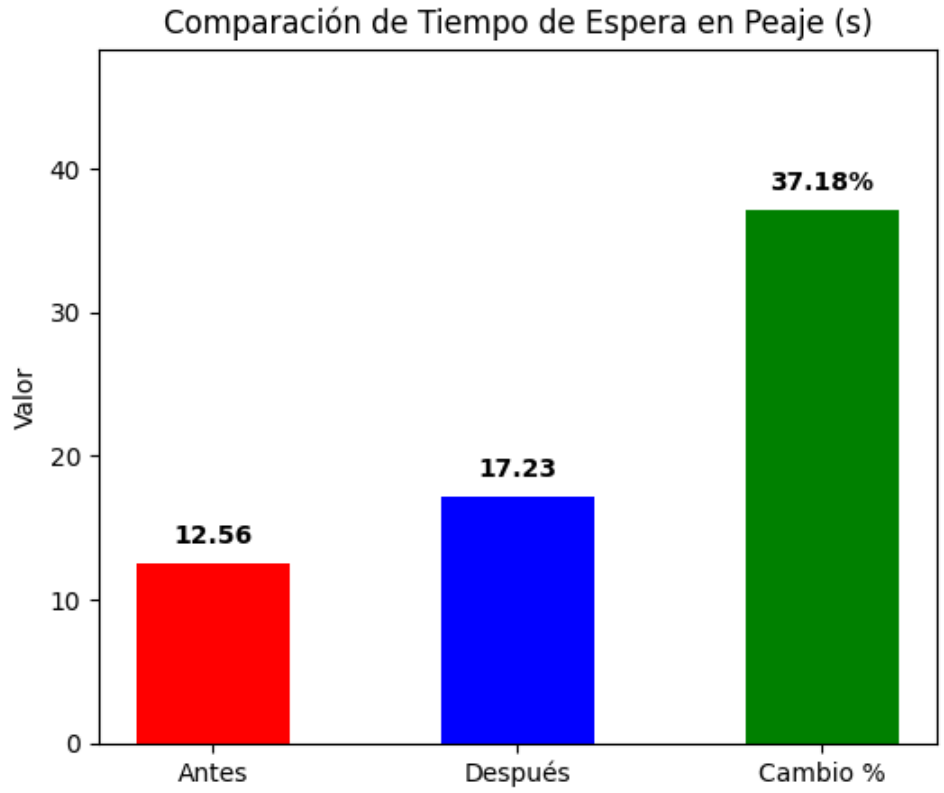

Figura 4.26: Análisis Comparativo Parámetros: Tiempo de espera en peaje

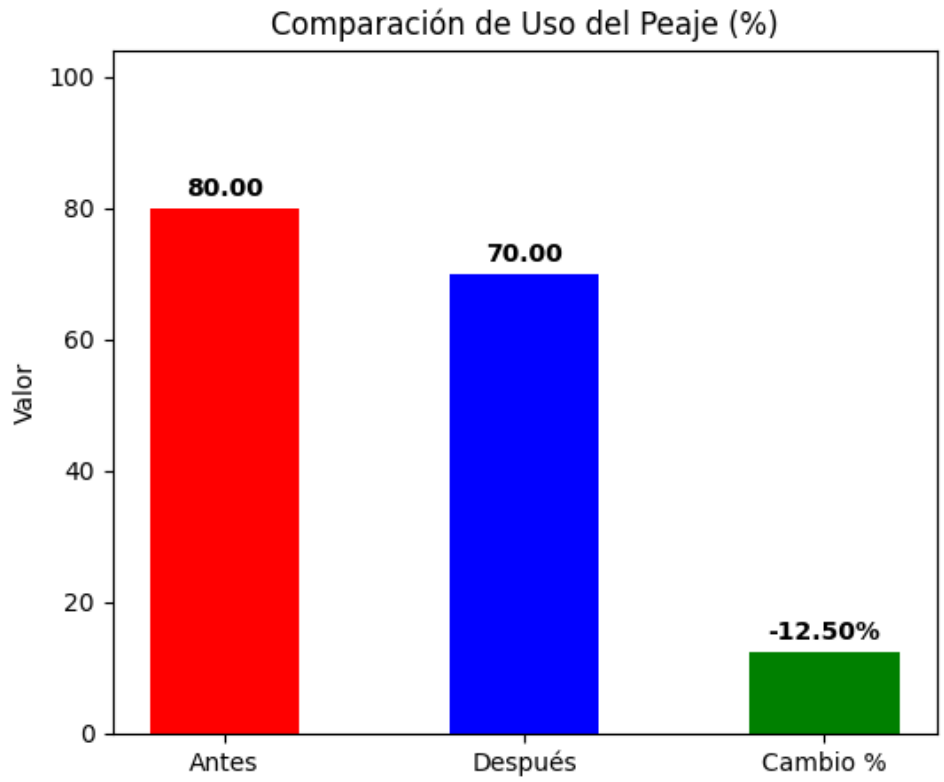

Figura 4.27: Análisis Comparativo Parámetros: Uso de peaje

Asimismo, las figuras, Fig. 4.28, Fig. 4.29, Fig. 4.30, Fig. 4.31 y Fig. 4.32, ilustran las diferencias de los resultados obtenidos en las simulaciones realizadas con los parámetros iniciales frente a aquellas efectuadas con los parámetros ajustados.

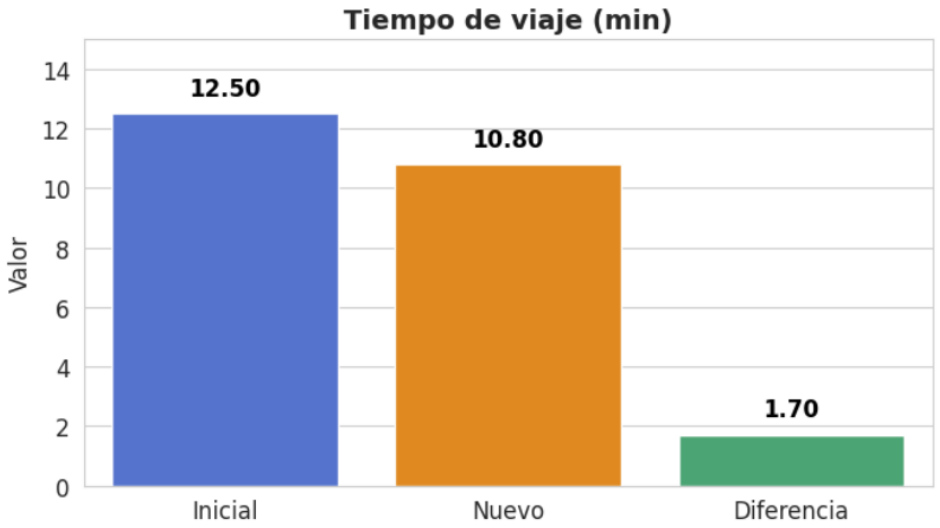

Figura 4.28: Análisis diferencial entre los resultados de las simulaciones: Tiempo promedio de viaje

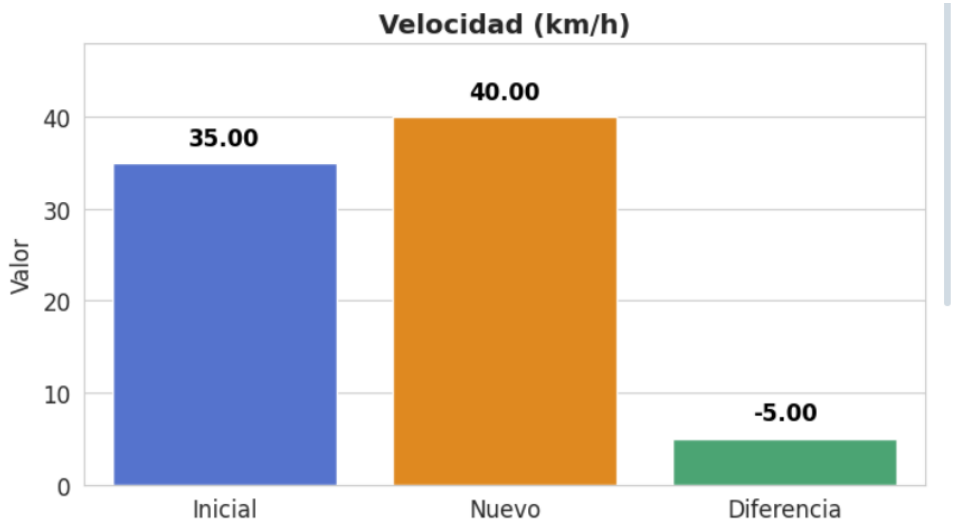

Figura 4.29: Análisis diferencial entre los resultados de las simulaciones: Velocidad promedio

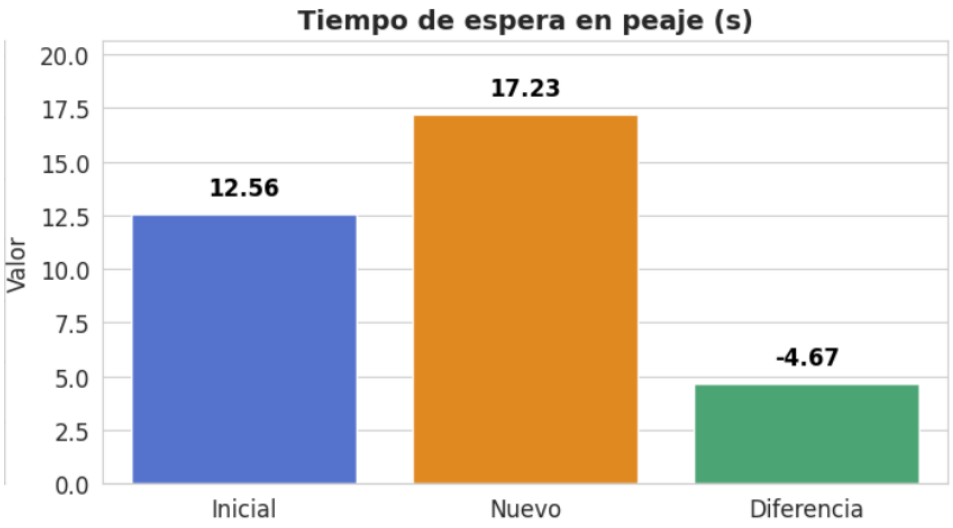

Figura 4.30: Análisis diferencial entre los resultados de las simulaciones: Tiempo de espera en peaje.

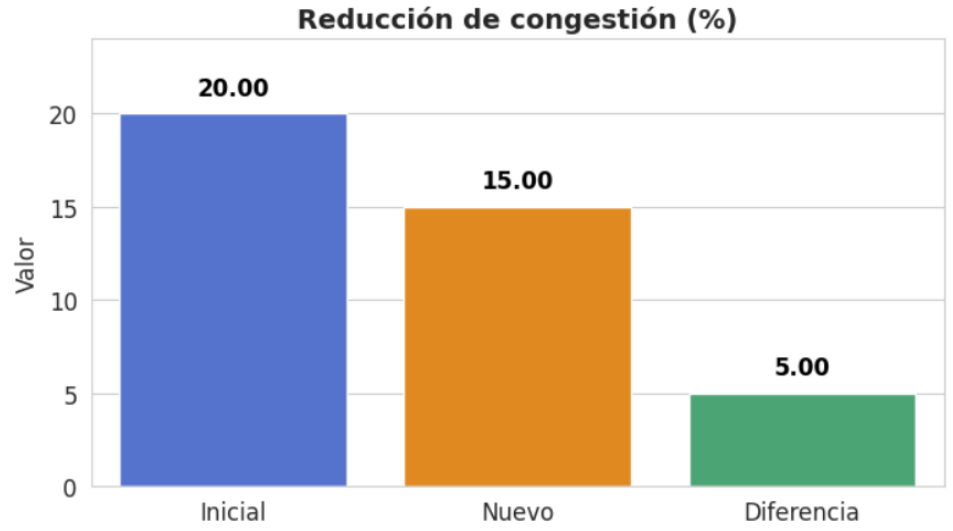

Figura 4.31: Análisis diferencial entre los resultados de las simulaciones: Reducción de congestión vehicular.

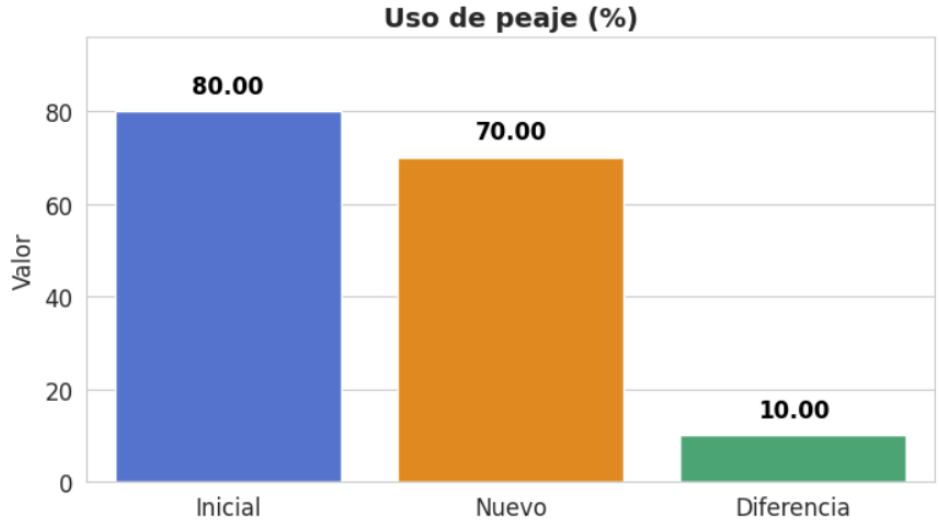

Figura 4.32: Análisis diferencial entre los resultados de las simulaciones: Uso de Peaje.

En general, las condiciones implementadas en las simulaciones lograron optimizar el flujo de tráfico y reducir tiempos de viaje, aunque el ligero descenso en el uso del peaje sugiere que hay margen para estrategias adicionales, como ajustar la tarifa en función de la demanda o proporcionar incentivos para mantener una alta tasa de utilización del peaje, Fig. 4.33.

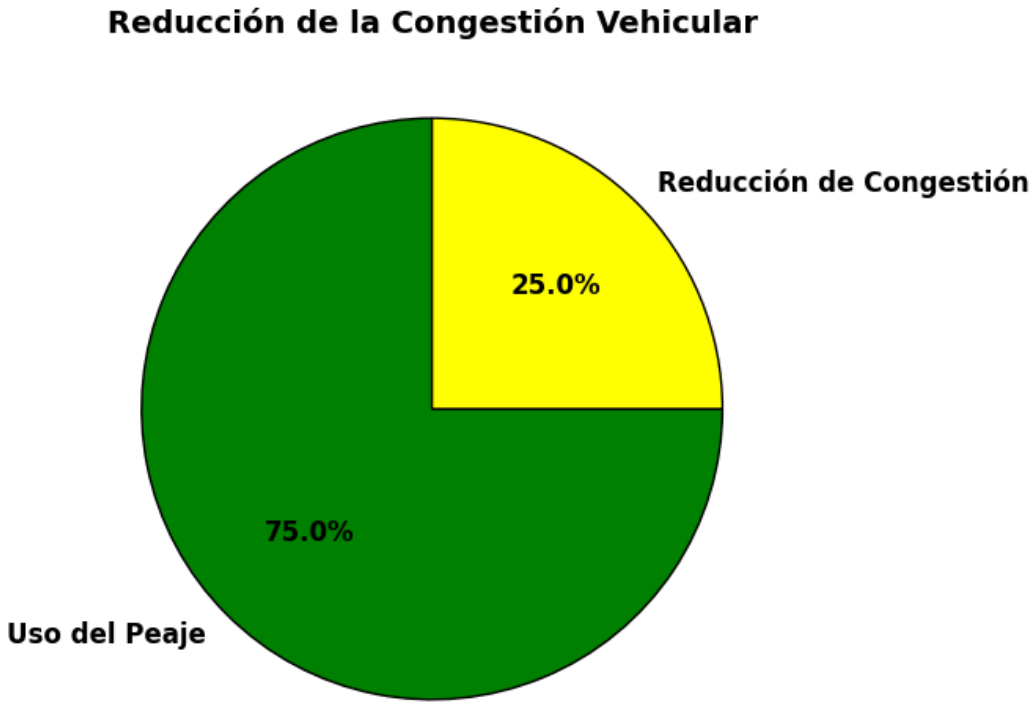

Figura 4.33: Análisis Comparativo Parámetros: Reducción de congestión vehicular

Capítulo 5

Conclusiones

El desarrollo de un sistema de peaje dinámico basado en IoT permitió demostrar su viabilidad como una alternativa innovadora para optimizar la gestión del tráfico vehicular, cumpliendo satisfactoriamente con el objetivo general de este proyecto.

Se realizó un análisis detallado sobre el funcionamiento de la tecnología IoT, así como los principios fundamentales relacionados con los sistemas de peajes dinámicos y la gestión del tráfico vehicular. Este estudio permitió establecer las bases teóricas necesarias para el desarrollo del sistema propuesto. Con base en este marco conceptual, se diseñó e implementó un prototipo funcional que integró sensores ultrasónicos, una Raspberry Pi y servicios en la nube a través de Azure IoT Central. Esta arquitectura demostró ser efectiva para capturar, procesar y transmitir datos, validando su viabilidad como solución tecnológica para optimizar el flujo vehicular.

Esta integración de hardware y software demostró ser una solución robusta que, combinada con el uso de herramientas como SUMO y TraCI, permitió modelar y simular escenarios de tráfico con precisión.

La simulación del sistema en un entorno controlado permitió replicar condiciones reales de tráfico y evaluar el comportamiento del sistema ante diferentes escenarios de congestión, evidenciando su capacidad para adaptarse dinámicamente a los flujos vehiculares. Los resultados de estas simulaciones demostraron una reducción del 13.6 % en el tiempo promedio de viaje (de 12.5 a 10.8 minutos) y un incremento del 14.3 % en la velocidad promedio (de 35 km/h a 40 km/h). Aunque el tiempo de espera en el

peaje aumentó un 37.2 % y el uso del peaje disminuyó del 80 % al 70 %, la utilización de la infraestructura vial fue más eficiente, reduciendo los niveles de congestión.

En conclusión, el sistema propuesto cumple con los objetivos planteados, como son los de optimizar el tráfico vehicular, reducir la congestión, los tiempos de viaje y garantizar el uso eficiente del peaje. Sin embargo, se identificaron oportunidades para futuras mejoras, como el diseño de tarifas dinámicas más precisas y la incorporación de nuevos sensores y herramientas analíticas para una mayor precisión.

Bibliografía

e. a. Angela Ribeiro. Sistema de posicionamiento global (gps): DescripciÓn, anÁlisis de errores, aplicaciones y futuro. *Dpto. de Tecnología Electrónica, E.T.S. Ingenieros de Telecomunicación. Universidad de Málaga*, 2000.

J. A. Ascencio. Big data e internet de las cosas para los sistemas inteligentes del transporte. *Insituto Mexicano de Transporte*, page 9, 2020.

J. Camarena. Aplicaciones del iotpara el control de congestión vehicular. *Universidad Tecnológica de Panamá*, page 91, 2018.

D. de la Fuente. Comunicaciones infraestructura a vehículo basadas en ieee 802.15.4 para señalización vial inalámbrica. *Uwicore, Ubiquituous Wireless Communications Research Laboratory*, 2011.

C. X. C. del Hierro. *Análisis del sistema de telepeaje de la Autopista General Rumiñahui*. Instituto de altos Estudios Nacionales, 2009.

J. R. Fernández. Aplicación de ia en las soluciones de peaje. caso de éxito en eeuu. *Exeforum*, page 26, 2019.

A. S. Florido. Comunicaciones vehículo/infraestructura. conducción asistida. *Universidad de Malaga*, 2018.

e. a. Francesc Robusté. Nuevas tecnologías en la gestión de autopistas: El peaje y los sistemas inteligentes de transporte. economía industrial. *Centro de Innovación del Transporte. CENIT, Universidad Politécnica de Cataluña*, 2003.

J. H. Gao. Roadrunner: infrastructure-less vehicular congestion control. *Massachusetts Institute of Technology, Computer Science & Artificial Intelligence Laboratory*, 2014.

Globalvia. Telepeaje o peaje dinámico., 2012. URL http://www.acega.es/index.php?option=com_content&view=article&id=7&Itemid=8&lang=es#:~:text=Este%20sistema%20permite%20abonar%20el,transmisor%20instalado%20en%20el%20vehÃŋculo.

P. Gokhale. Introduction to iot. *International Advanced Research Journal in Science, Engineering and Technology*, 5:1, 2018.

A. J. González García. *IoT: Dispositivos, tecnologías de transporte y aplicaciones*. Universitat Oberta de Catalunya, 2017.

Infraex. Todo lo que necesitas saber sobre los sistemas de peaje, 2023. URL https://infraex.net/todo-lo-que-necesitas-saber-sobre-los-sistemas-de-peaje.

S. Irigoyen. Congestion de transito y peajes: Una historia de dos ciudades. *Universidad Torcuato di tella*, 2022.

J. C. Leon Martinez. *Prototipo de plataforma informatica IoT para el sistema de transporte público masivo.* UNIVERSIDAD AUTONOMA DE BUCARAMANGA, 2018.

L. Letham. *GPS Fácil, Uso del sistema de posicionamiento global.* Paidotribo, 2001.

C. Z. E. Li. The embedded modules solution of household internet of things system and the future development. *Procedia Computer Science*, 166:351, 2020.

Microsoft. Azure iot sdk., 2024a. URL https://learn.microsoft.com/es-es/azure/iot-hub/iot-hub-devguide-sdks.

Microsoft. Azure cosmos db., 2024b. URL https://azure.microsoft.com/es-es/products/cosmos-db.

Microsoft. Azure iot hub., 2024c. URL https://azure.microsoft.com/es-es/products/iot-hub.

Microsoft. Azure iot central., 2024d. URL https://learn.microsoft.com/es-es/azure/iot-central/core/overview-iot-central.

Microsoft. Azure stream analtycs., 2024e. URL https://azure.microsoft.com/es-es/products/stream-analytics.

J. d. l. H. Molina. *Evaluacion de los sistemas de cobro electronico de peajes en la gestion de carreteras*. Universidad Politécnica de Madrid, 2019.

V. I. Orrala Rey. *Propuesta de arquitectura para capturar datos de buses del transporte público de Guayaquil basada en IOT*. Universidad Politécnica Salesiana, 2023.

S. V. Pazmiño. Implementación de un prototipo iot con semáforos inteligentes a través de una infraestructura de computación en la niebla. *Universidad de Las Americas*, 2019.

e. a. Roberto Rodriguez Solano. Las técnicas gps como herramienta en la gestión ambiental. *Observatorio Medioambiental*, 1999.

SUMO. Sumo, 2024a. URL https://sumo.dlr.de/docs/index.html.

SUMO. Traci, 2024b. URL https://sumo.dlr.de/docs/TraCI.html#introduction_to_traci.

V. M. Toalombo. Sistemas inteligentes de transporte: Diferentes aplicaciones para el caso de transporte terrestres. In *Knowledge E*. VII International Congress of Science, Technology, Entrepreneurship and Innovation (SECTEI), 2021.

M. L. Urquiza Cuellar. *Importancia de la implementación de sistemas inteligentes, como apoyo a la gestión del transporte terrestre automotor de carga en Colombia*. UNIVERSIDAD PILOTO DE COLOMBIA SECCIONAL DEL ALTO MAGDALENA, 2022.

L. F. P. Vaca. Sistemas inteligentes de transporte y nuevas tecnologías en el control y administración del transporte. *Facultad de Ingeniería, Universidad Pedagógica y Tecnológica de Colombia sede Tunja*, 2015.

J. Vasquez. Implementación de una red vehicular en modo infraestructura para aplicaciones en ciudades inteligentes. *Benemérita Universidad Autónama de Puebla*, 2020.

M. Ventura Navarro. *Trabajo de investigación sobre la automatización de despacho haciendo uso óptimo del transporte mediante la aplicación de tecnologías de internet de las cosas*. Universidad Peruana de Ciencias Aplicadas, 2018.