BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

Maestría en Ingeniería Electrónica, Opción Instrumentación Electrónica

Tesis para obtener el grado de

MAESTRO EN INGENIERÍA ELECTRÓNICA

Diseño e implementación de un nodo VANET considerando un sistema de control disparado por eventos

---

Presenta:

Néstor Antonio Tolentino Medrano*

Director de tesis:

Dra. Josefina Castañeda Camacho F.C.E.

Co-asesor de tesis:

Dr. José Fermi Guerrero Castellanos F.C.E.

*Becario CONACYT

Puebla, Pue. Enero 2021

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por le beca de manutención que se me fue otorgada durante el estudio de mi posgrado para la realización de este trabajo.

Agradezco a la Benemérita Universidad Autónoma de Puebla y a la Facultad de Ciencias de la Electrónica por el apoyo brindado para mi formación académica por medio del acceso a los laboratorios, bibliotecas, congresos, y por la beca de colegiatura.

Agradezco a mi directora de tesis la Dra. Josefina Castañeda Camacho por ser una guía indispensable para la culminación de este trabajo, por todas las asesorías impartidas, por contar siempre con la disposición de auxiliarme y por el apoyo brindado a lo largo de este posgrado.

Agradezco a mi co-director de tesis el Dr. José Fermi Guerrero Castellanos por toda su ayuda otorgada para el desarrollo de la tesis, los artículos, por el espacio otorgado para trabajar en el laboratorio y por ser una guía fundamental en este proceso.

Agradezco a mis sinodales el Dr. Gerardo Mino Aguilar, el Dr. Aldrin Barreto Flores y a la M.C. Ana María Rodríguez Domínguez por todas las observaciones realizadas para la mejora de este trabajo.

Agradezco enormemente de todo corazón el apoyo y amor incondicional otorgado por parte de mis padres, la Sra. Alma Rosa Medrano Vega y el Ing. Juan Antonio Tolentino García ya que me brindaron las herramientas y la oportunidad de estudiar en una ciudad lejana de mi hogar para continuar con mi formación profesional permitiéndome seguir mi sueños y siempre creyendo en mi, sin ustedes nada de esto hubiera sido posible y los amo enormemente.

Agradezco con especial cariño a mis hermanas Claudia Tolentino, Alma Tolentino y a mis sobrinos Jesús Guitiérrez y Leonardo Gutiérrez por todo el apoyo y amor brindado durante este proceso.

Agradezco con gran afecto a mis amigos Sandra Huerta y Mauricio Longinos por el gran apoyo proporcionado para salir adelante y por su tan preciada amistad, así como agradezco a Enrique Pérez, , Rodrigo Cuevas, Israel Sáenz, Saúl Villegas por su amistad y compañerismo.

Agradezco profundamente a todas las personas que me recordaron que era capaz de hacerlo y me motivaron a seguir adelante y no rendirme, especialmente agradezco con gran cariño a Nicté Carracedo, una persona muy especial para mi.

Resumen

En este trabajo se realiza un estudio sobre nodos VANET que conforman redes multi-agentes y que se rigen por medio de un control colaborativo, haciendo uso de una estrategia basada en eventos. Partiendo de ello, se presenta el modelo de un nodo VANET basado en un vehículo con movilidad omnidireccional (3,0), creado a partir de las herramientas proporcionadas por Matlab/Simulink en conjunto con el meta sistema operativo ROS para ejecutar rutinas que permitan la ejecución de formaciones de varios nodos que forman parte de una red de comunicaciones VANET. Por medio de simulaciones se comprueba el funcionamiento de los nodos y se implementan con ayuda del software Gazebo, y un conjunto de robots omnidireccionales controlados cada uno por una tarjeta Raspberry Pi 3B.

El documento describe en el capítulo 1 el establecimiento de los aspectos introductorios que definen a un nodo VANET, los objetivos a cumplir para el tema de tesis, el estado del arte, la justificación del tema y la descripción general de como se efectuó el trabajo realizado. Posteriormente, en el capítulo 2, se presentan las características las redes VANET, algoritmos de enrutamientos, el estándar IEEE 802.11, sistemas mecatrónicos, control colaborativo, control disparado por eventos y se presenta el modelado matemático del vehículo con movilidad omnidireccional que se utiliza como nodo VANET junto con el modelado de la estrategia de control colaborativo. En el capítulo 3 se muestran las simulaciones realizadas para verificar diseño del vehículo, la estrategia de control colaborativo, y el diseño de un algoritmo de enrutamiento basado en un algoritmo AODV en Matlab/Simulink. En el capítulo 4 se describe el proceso de implementación virtual e implementación física de los nodos VANET a través de las herramientas de Matlab/Simulink, ROS, Gazebo y una tarjeta Raspberry Pi. Finalmente, en el capítulo 5, se plasman los resultados de las implementaciones de los nodos VANET en interacción con otros nodos formando una red tanto de la parte virtual como de la parte física.

Capítulo 1

Introducción

Las redes de comunicaciones inalámbricas son fundamentales en la actualidad, debido a que constantemente se tiene la necesidad de enviar y recibir información variada entre dos o más usuarios. En sus orígenes, estas redes se diseñaron para proveer servicios a partir de una infraestructura fija conformada por estaciones base, las cuales son las que atienden a los usuarios que se encuentran dentro del rango de cobertura de estas. Tiempo después surge el concepto de las redes inalámbricas adaptables, mejor conocidas como MANET (redes Ad hoc móviles, por sus siglas en inglés), las cuales ya no requieren de una infraestructura definida para gestionar las comunicaciones. En estas redes, los usuarios, denominados nodos o agentes, forman una red dinámica en donde cada nodo sirve como un proveedor de información, así como de enrutador, de tal manera que cuando se requiere enviar información, se busca el camino óptimo para llegar al destino a través de los nodos vecinos [1].

Una subcategoría especial de las MANET, son las denominadas VANET (redes Ad hoc vehiculares por sus siglas en inglés), en las cuales los nodos móviles consisten en vehículos que conforman dicha red inalámbrica. Los avances en el área de la electrónica en conjunto con ramas como lo son la mecánica, las comunicaciones, entre otras, han dado como resultado el surgimiento de vehículos autónomos como lo son los VANT (vehículos aéreos no tripulados) y los VTA (vehículos terrestres autónomos). En la actualidad existen muchos trabajos que involucran a las VANET [2],[3], sin embargo, la gran mayoría de dichas investigaciones no efectúan una caracterización completa del sistema de comunicaciones en conjunto con el sistema de control requerido para efectuar apropiadamente el cumplimiento de los objetivos de dichas redes. El esquema de control necesario para el estudio de las VANET es el control colaborativo, donde los agentes son considerados como elementos individuales, pero a la vez como una sola entidad.

1.1. Objetivo general

Diseñar e implementar un nodo de comunicaciones como parte de una red vehicular Ad Hoc considerando un sistema de control disparado por eventos.

1.2. Objetivos específicos

1. Analizar los principios de funcionamiento de las redes de comunicaciones Ad Hoc vehiculares más utilizadas para el control de agentes no tripulados.

2. Analizar el principio de funcionamiento del control disparado por eventos.

3. Simular el funcionamiento de un nodo de comunicaciones dentro de una red VANET utilizando una estrategia de control disparado por eventos.

4. Implementar un sistema de transmisión de datos en el nodo VANET en interacción con otros nodos de la misma naturaleza.

5. Probar el sistema de comunicaciones completo en el control de una rutina de formación de vehículos no tripulados y su capacidad de operación con respecto a lo reportado en la literatura.

1.3. Estado del arte

En diferentes trabajos de investigación se han abordado temas referentes al análisis y diseño de redes VANET. Por ejemplo, en los trabajos abordados en [4], [5], [6] y [7], se analizan aspectos importantes de una red de comunicaciones compuestas por nodos VANET, tal como protocolos de enrutamiento reactivos y proactivos, así como algunas de sus métricas de evaluación importantes como lo son: fracción de entrega de paquetes, retardo promedio de fin a fin y carga de enrutamiento normalizada. Sin embargo, muchos de estos trabajos se centran en el aspecto de las comunicaciones y no incluyen el control como parte del análisis para el establecimiento de la red.

Por otra parte, en [8], [9], [10], [11] y [12], se analizan sistemas multi-agente, donde se diseña una estrategia de control colaborativo basada en un paradigma de disparo por eventos, considerando una formación de agentes con un líder-seguidor. Sin embargo, los análisis son específicamente enfocados en el área de control; además, los resultados obtenidos se presentan en etapa de simulación sin llegar a la implementación; eso sin mencionar que estos análisis no son enfocados a vehículos no tripulados, sino a sistemas multi-agente de una manera general.

En la mayoría de los trabajos relacionados con VANET, se da por sentado que el sistema de comunicaciones o el sistema de control ya están definidos y de esa forma, se procede a analizar únicamente el sistema de interés.

En [13] el trabajo principal se basa en la unión de los sistemas de comunicaciones y el control disparado por eventos, para simular el comportamiento de la red; mientras que en [14] se realiza un trabajo más completo donde se lleva a cabo la implementación del sistema utilizando vehículos aéreos no tripulados en una estrategia de control disparado por eventos usando un líder seguidor. Sin embargo, a pesar de ser un trabajo sumamente completo, la red es parcialmente descentralizada debido a que parte del procesamiento se realiza de manera externa, en lugar de que los agentes sean los encargados de realizar en su totalidad el proceso de control y de gestionar el direccionamiento de sus comunicaciones.

1.3.1. Conclusión

Al haber analizado algunos ejemplos sobre el trabajo realizado en el campo de las VANET, es posible apreciar un campo reducido de trabajos en el área de comunicaciones en conjunto con el área de control, por lo que esta propuesta permite continuar con el trabajo realizado en [14], en donde el objetivo es llevar el procesamiento y la inteligencia del enrutamiento a cada uno de los nodos que conforman la red VANET, sin hacer necesario el uso de paqueterías que permitan gestionar la distribución de la información y el enrutamiento de los agentes, así como el cálculo de sus señales de control, valiéndose de los complementos añadidos a Matlab/Simulink, para trabajar con el sistema ROS para ambientes robóticos.

1.4. Justificación

Las redes móviles Ad hoc orientadas a vehículos no tripulados son de gran importancia en la actualidad, debido a que estas pueden ser utilizadas en una gran diversidad de aplicaciones, tales como medios de asistencia y/o rescate para situaciones de emergencia, desastres naturales (temblores, erupciones volcánicas, huracanes, entre otras, donde las infraestructuras fijas puedan llegar a resultar deshabilitadas), control vehicular para evitar congestionamientos y accidentes en carreteras, sensores ambientales que permitan monitorear actividades humanas y compartir datos de interés, para uso militar como defensa aérea, marítima, terrestre, entre otros [13]. Esto resulta posible ya que las VANETs conforman redes inteligentes con capacidad de reconfiguración, que permiten llevar a cabo tareas actuando de manera conjunta en una forma descentralizada.

Para mejorar ampliamente la posibilidad de aplicación de estas tecnologías es fundamental caracterizar por completo cada uno de los nodos VANET en una red de comunicaciones, tomando especial interés en el sistema de comunicación y el sistema de control

colaborativo donde cada agente tiene la posibilidad de saber su localización, así como la de sus vecinos, además de que se busca implementar una estrategia de control disparado por eventos, la cual permite que el mecanismo de control de los nodos sea de naturaleza asíncrona en lugar de en tiempo continuo, de tal forma que la comunicación entre los agentes únicamente es habilitada cuando es requerido por alguno de ellos, permitiendo de esta manera reducir tanto el consumo energético como el consumo de los recursos computacionales [15].

1.5. Estructura y delimitación del trabajo

Para el trabajo de tesis propuesto, se presenta el siguiente diagrama de flujo en la figura 1.1 en función con las actividades requeridas para cumplir los diferentes objetivos específicos planteados.

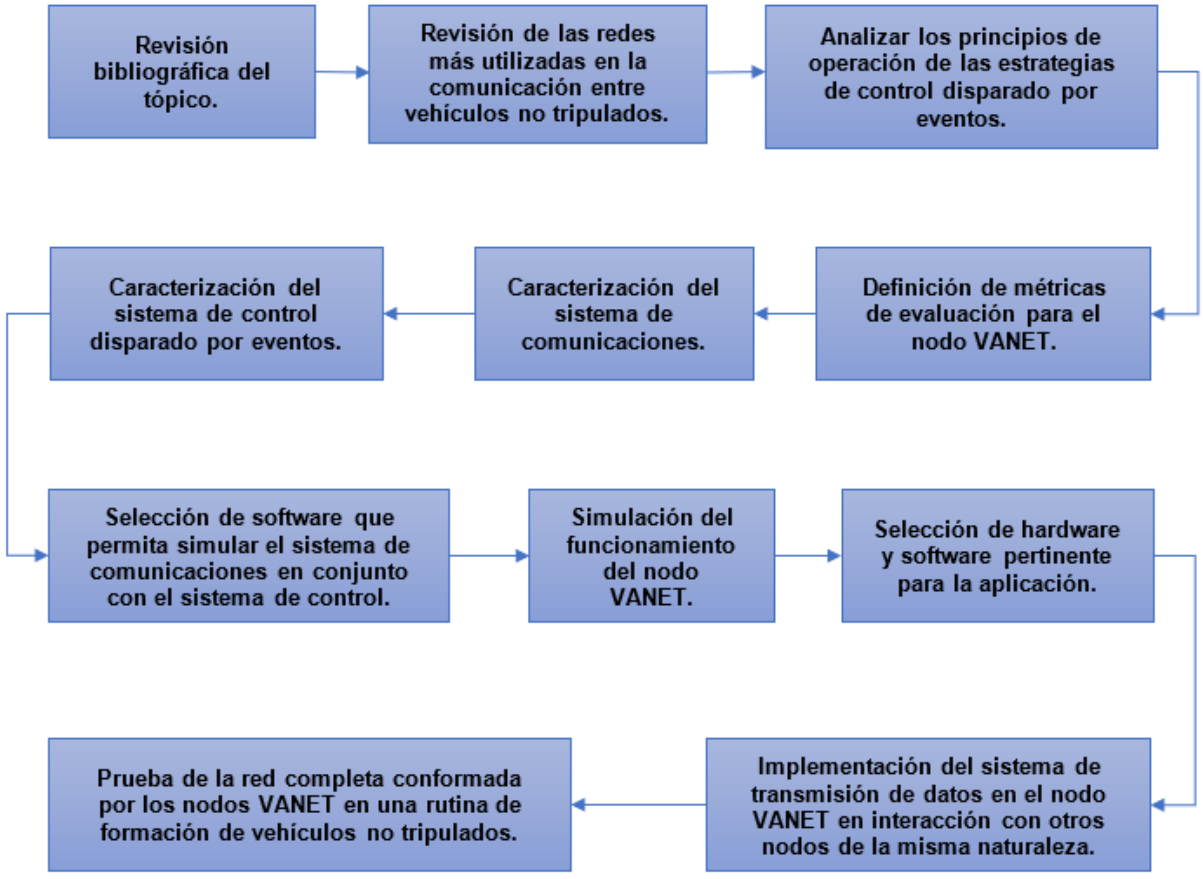

Figura 1.1: Diagrama de flujo de actividades realizadas.

Se toma como punto de partida el trabajo [14]. En dicho trabajo se llevó a cabo el diseño una red FANET caracterizando tanto el sistema de control como el sistema de comunicaciones. Sin embargo, la parte de implementación del trabajo se efectuó mediante una paquetería que gestionaba las comunicaciones y el control de los nodos y se busca

que cada agente sea capaz de realizar estas operaciones por cuenta propia.

Inicialmente, para realizar las pruebas correspondientes se considerará el grafo ilustrado en la figura 1.2

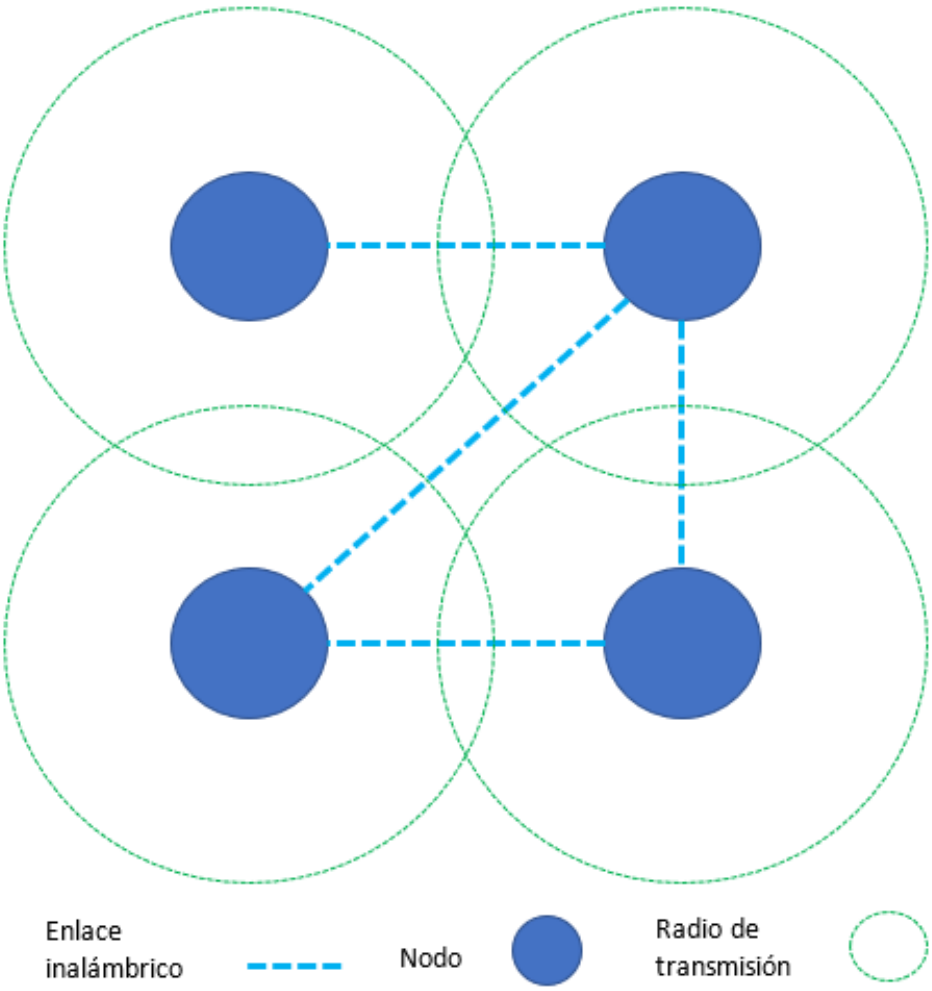

Figura 1.2: Grafo de 4 nodos.

La formación está compuesta por 4 nodos que representan vehículos no tripulados en donde se propone utilizar vehículos terrestres omnidireccionales, de tal manera que para este trabajo, cada nodo se caracteriza y se configura permitiendo que cada uno pueda gestionar su proceso de comunicación y control por eventos según las condiciones que se lleguen a presentar para después llevar al cabo el proceso de comunicación por medio de la implementación del estándar IEEE 802.11b. De igual manera se parte del protocolo de enrutamiento AODV (ad hoc on demand distance vector), ya que ha demostrado presentar buenos resultados relacionados al desempeño de una red con estas características [14].

Para la parte experimental se cuenta con una arena de pruebas rectangular, con un

sistema compuesto de cuatro cámaras desplegadas en cada esquina superior denominado Optitrack, con el propósito de poder determinar la posición de los vehículos utilizando el software ROS como se presenta en la figura 1.3. También se cuenta con el software Matlab/Simulink y el simulador Gazebo para una implementación virtual.

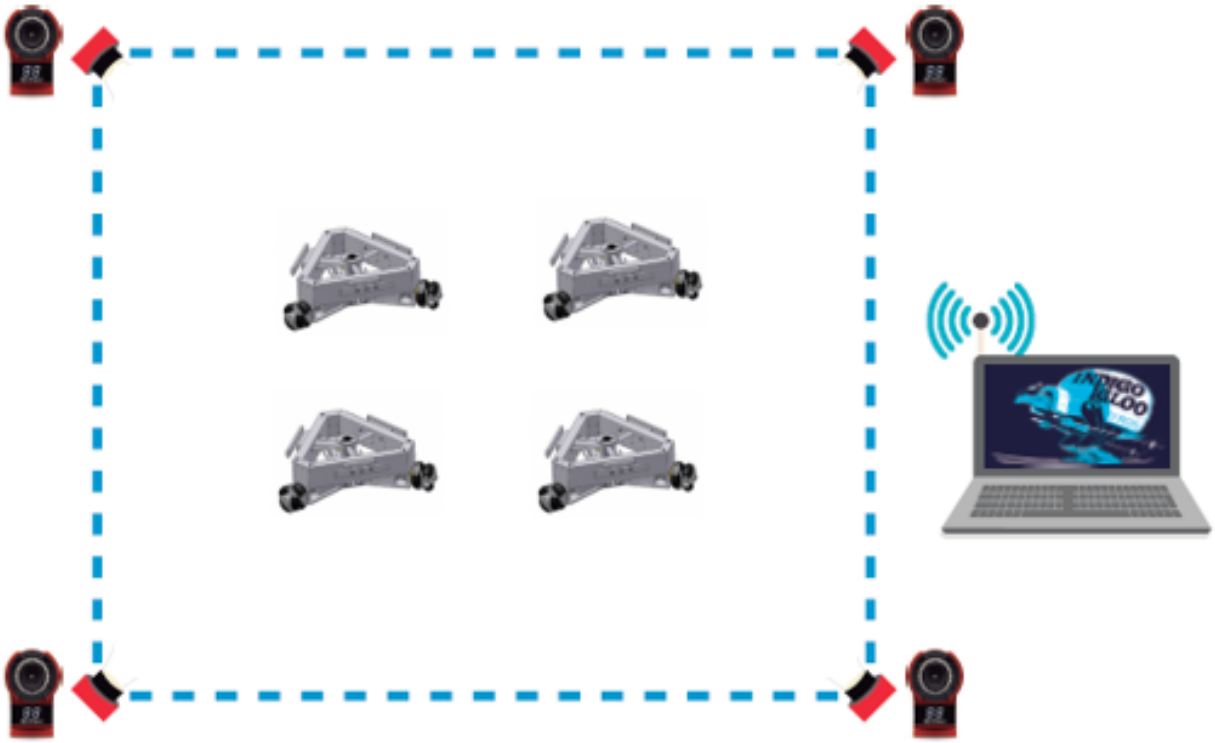

Figura 1.3: Arena de prueba y sistema de cámaras Optitrack

A grandes rasgos y de manera inicial, la composición general de cada nodo estará establecida por el diagrama mostrado en la figura 1.4.

En este diagrama, el detector de eventos espera una señal de evento donde el reloj muestrea cada cierto tiempo si se cumple o no dicha condición y una vez que sea el caso, manda el aviso al generador de eventos para proceder a calcular la señal de control con base en las posiciones de los otros nodos VANET y del mismo nodo, además de que también toma en consideración los valores previos de las posiciones (almacenados en la memoria), para efectuar dicho cálculo. Posteriormente, la señal de control es dirigida a los motores, los cuales funcionan como actuadores, con el fin de mover el nodo a la posición deseada. Mientras tanto, el sistema Optitrack o el entorno de Gazebo, leen continuamente la posición del nodo, para retroalimentar el mismo control y retroalimentar a los nodos vecinos. El procesador obtiene dicha posición y a su vez se dedica a enviar la información a los demás nodos para, de acuerdo con las tablas de enrutamiento, enviar la información de su posición a sus vecinos correspondientes mediante el algoritmo de enrutamiento.

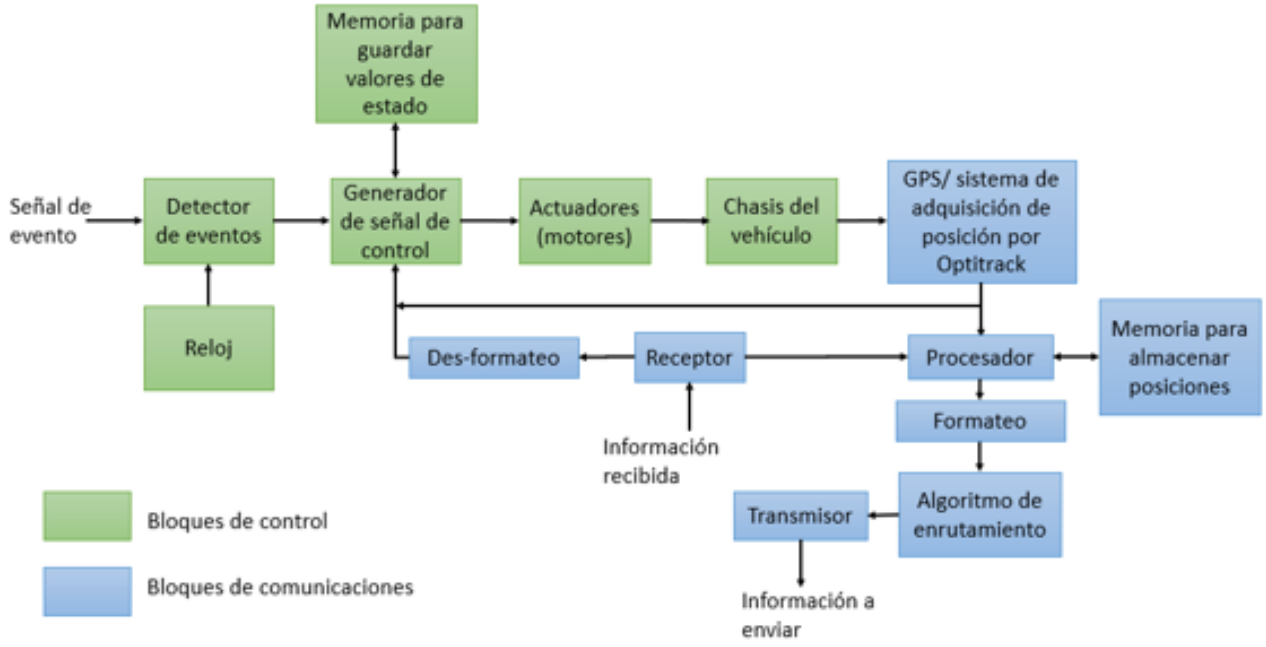

Figura 1.4: Diagrama de bloques general de cada nodo VANET.

Capítulo 2

Marco teórico

A lo largo del presente capítulo, se tratan los fundamentos teóricos requeridos para llevar a cabo el cumplimiento de los objetivos de la tesis planteados en el capítulo 1. La estructura consiste en presentar los aspectos generales de las redes Ad hoc, para después adentrarse en los CPVS, centrándose principalmente en estos como redes VANET, para posteriormente hablar sobre los tipos de enrutamiento y sus características y así, finalmente detallar los sistemas mecatrónicos y robóticos, profundizando en el diseño y modelado matemático, control colaborativo y control disparado por eventos del nodo VANET diseñado en este trabajo de tesis.

2.1. Redes Ad hoc móviles

Las redes MANET (mobile Ad hoc networks), consisten en una agrupación de nodos o agentes que no precisan de una infraestructura establecida para ejecutar el proceso completo de comunicación. Estos agentes tienen un comportamiento dinámico, lo que permite a los usuarios de estas redes tener una mayor movilidad. Hoy en día este tipo de redes forman parte fundamental de la actual generación de telefonía 4G/4.5G, ya que estas utilizan redes híbridas, es decir, una combinación de redes con estaciones base y redes Ad hoc, para brindar servicio a sus usuarios [1].

En la figura 2.1, se ejemplifica una red Ad hoc que contiene 6 nodos. Las líneas que unen los nodos se denominan enlaces, estas están relacionadas a su vez con las distancias, que pueden expresarse en unidades de distancia o del costo de recursos que se requiere para llegar al nodo destino. Los nodos a los que cada agente puede acceder al atravesar únicamente un enlace se denominan nodos vecinos [1].

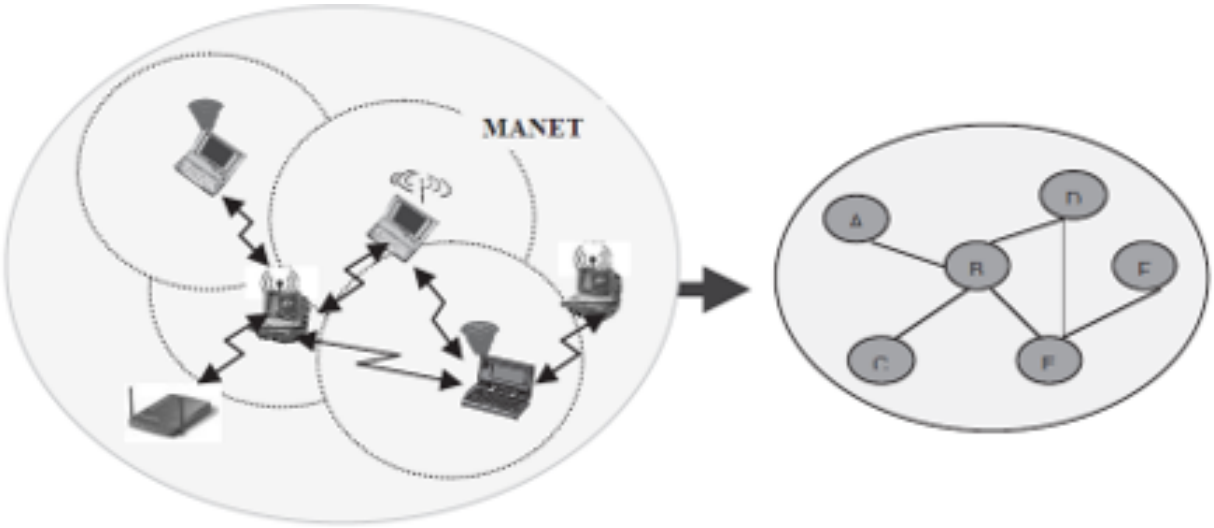

Figura 2.1: Ejemplo de una red Ad hoc.

2.1.1. Características de las MANET

- Inalámbricas. Los nodos que conforman la red transmiten toda la información deseada a enviar sin la utilización de alambrado.
- Autónoma y sin infraestructura. Al no requerir infraestructura, los nodos de la red tienen un comportamiento independiente, debido a que no existe una administración centralizada que los comande, de tal forma que los agentes envían su propia información de manera independiente y son capaces de actuar como enrutadores.
- Enrutamiento multi-salto. Cada nodo actúa como enrutador, de tal forma que la información llega al destino atravesando los nodos necesarios que se encuentren disponibles por medio de saltos de agente en agente, según la estrategia de enrutamiento que se esté utilizando.
- Movilidad. En estas redes, cada nodo tiene la libertad de desplazarse sin que se interrumpa la conexión (siempre y cuando los nodos aún se encuentren dentro de los límites del área de cobertura). Al ser una red dinámica, los patrones de comunicación cambian conforme la topología de los nodos cambia.
- Auto-organización. Al no poseer una estructura centralizada, los nodos determinan por sí mismos todos los datos necesarios para que la comunicación se efectúe de manera apropiada.
- Manejo de batería. Debido a que los nodos no están fijos en un solo lugar, para mantener las comunicaciones es necesario contar con batería suficiente, así como estrategias de optimización del consumo energético del sistema para asegurar un largo tiempo de utilización de la red.

2.2. CPVS y redes VANET

Los CPS (sistemas ciber-físicos, por sus siglas en inglés) integran elementos que interactúan con variables físicas, ya sea por medio de sensores (entrada) o actuadores (salida), para ser procesados por recursos computacionales teniendo a su vez la capacidad de comunicarse con otros CPS de la misma naturaleza. Cuando estos CPS ejercen específicamente acciones de generación de movimiento, caen en la categoría de CPVS (sistemas vehiculares ciber-físicos) [16], [17]. Cuando estos CPVS conforman una red para realizar acciones coordinadas, entran en la categoría de VANET. Las VANET son una rama de las MANET, en las cuales los nodos que conforman la red consisten en vehículos no tripulados. De acuerdo con [18], algunas características fundamentales de las VANET son:

- Topología altamente dinámica: debido a las frecuentes altas velocidades de los vehículos, los nodos VANET tienden a abandonar e incorporarse rápidamente a alguna red cercana, lo que genera cambios rápidos en la topología actual de la red.
- Desconexión frecuente: debido al dinamismo de la red, es común que se generen desconexiones repetidamente lo que pudiera generar interrupciones en los enlaces formados.
- Comunicación geográfica: un factor importante para realizar la comunicación entre nodos VANET es la ubicación geográfica de cada nodo, para tener una transmisión y recepción de la información más eficiente.
- Movilidad restringida y predicción: a pesar de tener topologías dinámicas, los nodos VANET generalmente siguen un cierto patrón de movimiento, delimitado por calles, obstrucción de objetos o de otros nodos VANET.
- Modelo de propagación: debido a los diferentes lugares en los que opera un nodo VANET (entornos rurales, autopistas, ciudades) existen diferentes factores físicos que afectan de una manera u otra el proceso de propagación de la información por medio de ondas, lo que exige un análisis sobre estos factores para tener un mejor proceso de comunicación.

2.2.1. Consideraciones y dificultades de diseño

Al momento de diseñar una MANET, debido a que estos tipos de redes tienen un grado mayor de complejidad que las redes basadas en infraestructura, surgen desafíos necesarios de afrontar al momento de su implementación. En la tabla 2.1, se observan las diferentes consideraciones para cada nivel del modelo OSI. Sin embargo, durante este trabajo, se afronta únicamente con la capa física y la capa de enlace de datos del modelo OSI.

Tabla 2.1: Retos de cada capa del modelo OSI.

Capa de red	**Retos de cada capa**
Nivel 7: Aplicación	Aplicaciones nuevas/destructivas, configuración de red, servicios de localización, seguridad
Nivel 6: Presentación	
Nivel 5 Sesión	
Nivel 4: Transporte	Adaptación TCP
Nivel 3: Red	Enrutamiento IP, direccionamiento, optimización.
Nivel 2: Enlace de datos	Control de acceso al medio, control de error.
Nivel 1: Física	Uso de espectro.

2.2.2. Estándar IEEE 802.11

Para solucionar los problemas que se presenten en la capa física y la capa de enlace de datos, se implementa el estándar IEEE 802.11, el cual consiste en el estándar para redes inalámbricas, establecido por el instituto de ingenieros eléctricos y electrónicos.

El 802.11b surge como una categoría de la 802.11. y se creó con el fin de conseguir una tasa de transmisión mayor en la banda de los 2.4 GHz, HigherSpeed Physical Layer Extension en la banda de 2.4 GHz, usualmente se le conoce como Wireless Fidelity (Wi-FI). Es capaz de operar a velocidades de hasta 11 Mbps, soportando también 5.5, 2 y 1 Mbps.

Este estándar utiliza exclusivamente la modulación DSSS con el sistema de codificación CCK (Complementary Code Keying), que sólo funciona con este tipo de modulación y es de los estándares que se utilizan con más frecuencia en la actualidad.

2.2.3. Capa física

La capa física es la parte base del modelo OSI en la que se establecen los parámetros eléctricos y mecánicos necesarios para que la red pueda operar de manera apropiada, incluyendo aspectos tales como frecuencia, el ancho de banda, el tipo de modulación, las características de las antenas y la velocidad de transmisión de datos. En la tabla 2.2 se muestran algunas de las principales características de cada sub-estándar que compone al estándar 802.11.

Tabla 2.2: Estándar IEEE 802.11

Estándar	Frecuencia	Ancho de banda	Modulación	Arreglo de antenas	Transmisión de datos
802.11	2.4 GHz	20 MHz	DSSS, FHSS	N/A	2 Mbits/s
802.11b	2.4 GHz	20 MHz	DSSS	N/A	11 Mbits/s
802.11a	5 GHz	20 MHz	OFDM	N/A	54 Mbits/s
802.11g	2.4 GHz	20 MHz	DSSS, OFDM	N/A	542 Mbits/s
802.11n	2.4, 5 GHz	20, 40 MHz	OFDM	MIMO	600 Mbits/s
802.11ac	5 GHz	40, 80, 160 MHz	OFDM	MIMO	6.93 Gbits/s

2.2.4. Capa de enlace de datos

También conocida como capa MAC (media Access control). Esta capa se encarga de proveer una confiable transmisión de datos utilizando recursos de la capa física, donde se agregan bloques de control de error, control de flujo y sincronización.

2.3. Enrutamiento

El alto dinamismo de las VANET resulta en una dificultad al momento de realizar el enrutamiento para enviar la información requerida al nodo destino, por lo que se requiere de la implementación de protocolos para un correcto y eficiente establecimiento de la ruta entre dos agentes. Según [1], algunas de las características fundamentales de estos protocolos de enrutamiento son:

- Fácil implementación.
- Establecimiento de rutas óptimas libres de ciclos.
- Asignación de múltiples caminos.
- Adaptación de cambios en la topología.
- Eficiencia en consumo energético, ancho de banda y cómputo.
- Escalabilidad.
- Seguridad y confiabilidad.
- Soporte de calidad de servicio.

De acuerdo con lo establecido en [1], se pueden apreciar dos tipos de protocolos de enrutamientos.

2.3.1. Protocolos proactivos de enrutamiento

- DSDV (Destination-Sequenced Distance-Vector): Es un protocolo de distancia por vector, diseñado para ser implementado en las MANET. Cada nodo mantiene una tabla de enrutamiento con la ruta más conveniente a seguir, conforme a los saltos requeridos. Para evitar rutas cíclicas, se agrega un número de secuencia, el cual incrementa cada vez que se presenta un cambio en sus nodos vecinos. Posteriormente, este número es usado para seleccionar rutas alternativas eligiendo siempre el número con mayor secuencia.

- CGSR (clusterhead Gateway switch routing): Al DSDV se le añade programación de prioridad de token, reservación de ruta, etc., para mejorar el rendimiento del protocolo e incrementar su escalabilidad.

- WRP (Wireless routing protocol): Se utilizan cuatro tablas en cada nodo (distancia, costo del enlace, rutas e información de retransmisión del mensaje).

- OLSR (Optimized link state routing): Utiliza Multipoint Relays (MPR) para enviar información de ciertos conjuntos de enlaces.

- FSR (Fisheye state routing): El protocolo propaga información del estado del enlace de manera frecuente a los nodos cercanos.

- LAN-MAR (Land-Mark): Actúa como un FSR jerárquico donde divide la red en subgrupos de nodos.

2.3.2. Protocolos reactivos en demanda

- DSR (Dynamic source routing): Es un protocolo libre de ciclos en el cual cada nodo cuenta con una memoria cache las cuales permiten descubrir rutas para entregar la información al nodo destino.

- AODV (Ad hoc on-Demand distance vector): Es una versión mejorada del protocolo DSDV, la cual minimiza el número de transmisiones de ruta creando rutas bajo demanda.

- TORA (Temporally ordered routing algorithm): Es un protocolo bajo demanda de ruta altamente adaptativo, útil para ambientes con una alta movilidad y una densa población de nodos.

- ABR (Associativity-Based Routing): Es un protocolo libre de ciclos con una métrica de enrutamiento, llamada grado de asociación de estabilidad, con selección de rutas más estables con menos actualizaciones.

2.4. Métricas de evaluación

Existen diversas métricas, las cuales permiten evaluar la manera en la que se desempeña una red Ad hoc, con base en sus protocolos de enrutamiento. Algunos de los parámetros, de acuerdo con [19], se enlistan de la siguiente manera:

- Gasto de enrutamiento: Indica cuantos paquetes deben ser enviados para el descubrimiento de rutas y para su mantenimiento.
- Retardo promedio: Indica cuánto tarda un paquete en enviar la información de principio a fin.
- Rendimiento: Describe el número total de bits enviados a capas superiores por segundo.
- Retardo de acceso al medio: Representa el tiempo que le toma a un nodo accesar a la capa de acceso al medio desde que se comienza la transmisión de paquetes.
- Tasa de entrega de paquetes: Indica la cantidad de paquetes que fueron enviados con respecto a la cantidad de paquetes recibidos.

2.5. Sistemas mecatrónicos

En sus inicios, los sistemas mecánicos fueron adoptando el área eléctrica con el fin de incluir sistemas que pudieran controlar ciertos parámetros por medio de sensores, actuadores, etc. Posteriormente se introdujeron sistemas con tecnología para el procesamiento de información y control automático, dando apertura a la inclusión de computadoras y microprocesadores para estas tareas. Estos sistemas más completos son denominados sistemas mecatrónicos [14].

2.5.1. Esquema de flujo de información en los sistemas mecatrónicos

Los sistemas mecatrónicos se conforman por la parte eléctrica que permite medir las variables importantes y generar las señales físicas necesarias, la parte mecánica que permite ejecutar los parámetros deseados y la parte del procesamiento de información que controla el flujo de esta a través del sistema completo. La manera en la que estos elementos coexisten y la forma en la que fluye la información se muestran en la figura 2.2, en donde el usuario utiliza la interfaz hombre-máquina para modificar los valores de referencia que permitan al sistema llegar a un valor determinado, entonces el procesamiento central se encarga de generar las señales eléctricas necesarias para llegar a dichos parámetros de

acuerdo con los valores establecidos y los valores medidos. De manera similar, es importante considerar los sistemas energéticos que se utilizan para alimentar el sistema y que este sea energéticamente eficiente [20].

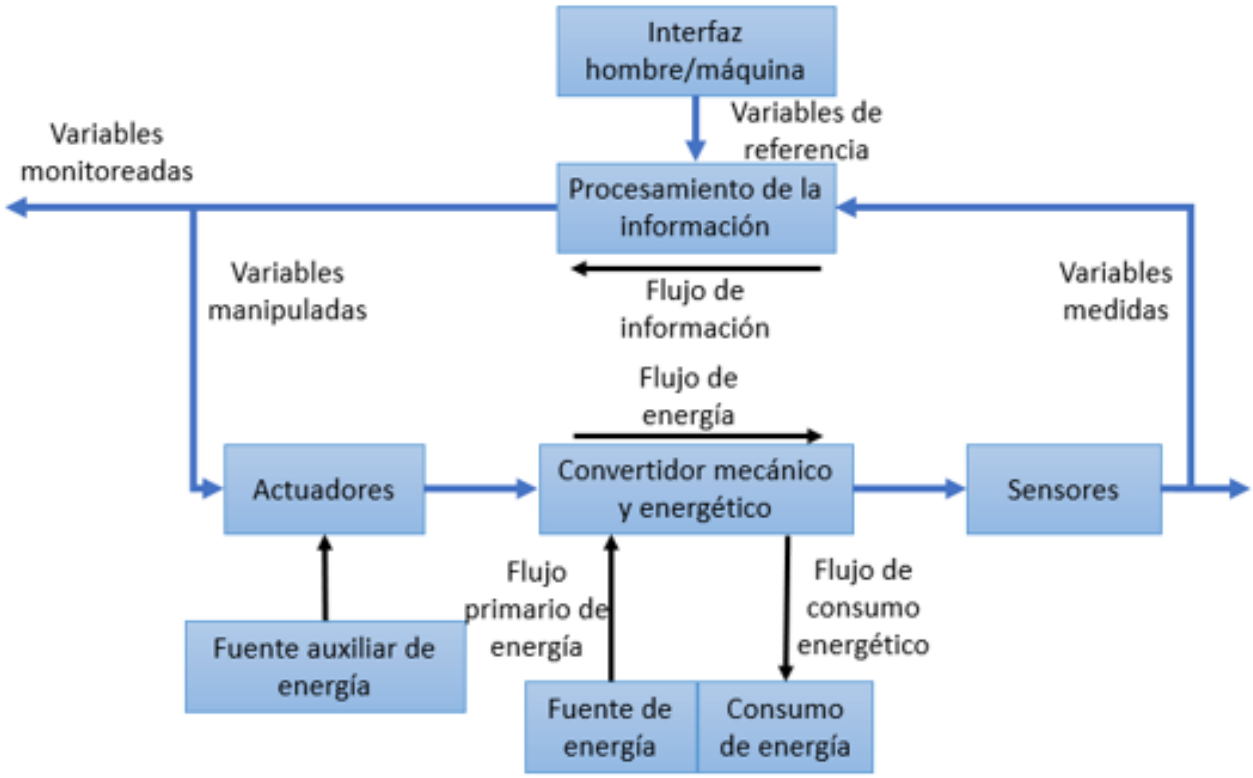

Figura 2.2: Diagrama de flujo de información en los sistemas mecatrónicos.

2.5.2. Diagrama "V" en el diseño de sistemas mecatrónicos

De acuerdo con [21], los sistemas mecatrónicos se encuentran en constante evolución debido a las mejoras que se realizan en cuestiones de hardware y software que se ven involucradas en estos sistemas, por ello, es posible utilizar herramientas de diseño implementadas en áreas similares para llevar a cabo el desarrollo de estos sistemas. La metodología de diseño para sistemas mecatrónicos fue desarrollada por la asociación alemana de ingenieros en 2002. Sus principales elementos consisten en:

- El ciclo general del problema a resolver a nivel micro.
- El modelo V a nivel macro.
- Módulos de procesos predefinidos para pasos de trabajo recurrentes.

En el primer punto, se analizan los requerimientos específicos a cumplir, además de que se analiza la situación dada, mientras que en el nivel macro se utiliza el diagrama V

para describir el proceso de diseño del sistema mecatrónico. En la figura 2.3, se aprecia dicho diagrama. En este, se parte de los requerimientos para realizar el modelado y diseño del sistema en los diferentes dominios de este (mecánico, electrónico e informático) para después proceder a la integración de estos sistemas que permitan llevar a cabo la función requerida o el producto, siempre midiendo los parámetros para validar que los resultados son los correctos.

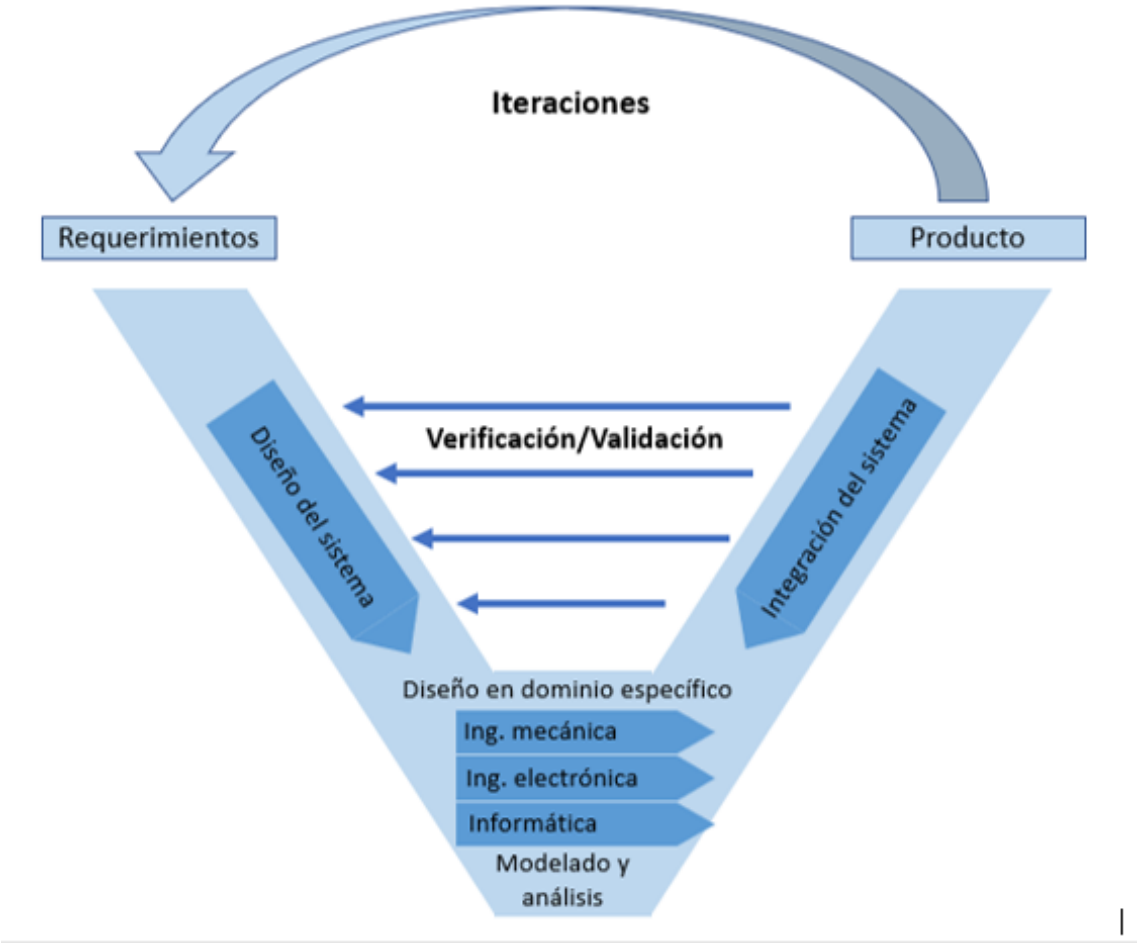

Figura 2.3: Diagrama V para el diseño de sistemas mecatrónicos

2.6. Sistemas robóticos como caso especial de sistemas mecatrónicos

2.6.1. Definición y clasificación

Los sistemas robóticos surgen de la necesidad de realizar tareas que pueden ser peligrosas o agobiantes para los seres humanos, por lo que estos sistemas tienen la finalidad de realizar tareas de manera automática destinadas para seres humanos de una manera igual o incluso mejor que estos [22]. La clasificación de los sistemas robóticos consiste en:

- Robots manipuladores: Utilizados principalmente en la industria. Consisten en brazos articulados programables diseñados para el movimiento de objetos tales como materiales de producción, herramientas, dispositivos especiales, etc.
- Robots móviles: Surgen a partir de la necesidad de extender el campo de aplicación de los sistemas robóticos, permitiendo que estos no estén únicamente anclados a una superficie y de esta manera se limite aún más la intervención humana en los procesos.
- Robots autónomos y telerobótica.
    - Teleoperados: Las tareas de percepción del entorno, planificación y manipulación compleja son realizadas por humanos a distancia.
    - Autónomos o inteligentes: Son máquinas capaces de percibir, modelar el entorno, planificar y actuar para alcanzar objetivos con poca o nula intervención humana.

2.6.2. Aplicaciones civiles y militares

Los robots son ampliamente utilizados en una gran cantidad de áreas, algunas de ellas son:

- Civiles: Algunas de las aplicaciones civiles más importantes radican en el sector industrial como lo es la carga de objetos, colocación de material en el proceso, aplicaciones médicas, manufactura, pintura, asistencia a gente con discapacidades, asistencia para rescates en ambientes peligrosos, exploración espacial, exploración oceánica, asistencia en el hogar [23].
- Militares: Robots de carga y transporte de objetos, micro robots para espionaje, vehículos aéreos no tripulados, vehículos terrestres no tripulados, vehículos submarinos no tripulados [24].

2.6.3. Vehículos omnidireccionales

De acuerdo con [25], los robots omnidireccionales consisten en aquellos que pueden efectuar movimientos en cualquier dirección desde cualquier posición, sin tener la necesidad de rotar primero para ello. Pueden efectuar los tres componentes fundamentales de movimiento como las translaciones frontales/en reversa, laterales y de rotación. Esta ventaja que poseen este tipo de vehículos radica principalmente en la configuración de sus ruedas.

Omnidireccional de 3 ruedas: Consiste en tres ruedas omnidireccionales, colocadas como se muestra en la figura 2.4 [25]. Estos vehículos tienen la ventaja de contar con un

sistema de control simple, el cual consiste en que para una velocidad deseada se realiza una única combinación de velocidades de cada rueda.

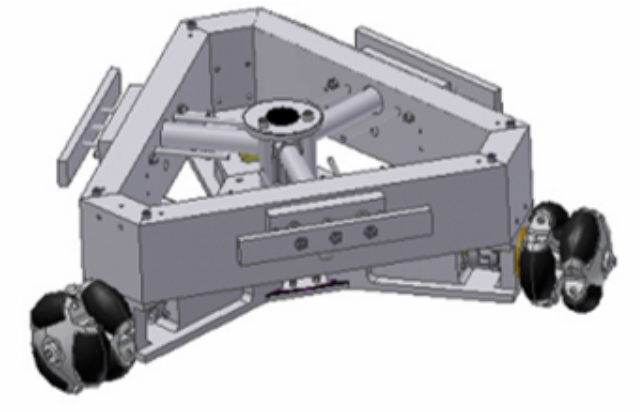

Figura 2.4: Estructura del vehículo omnidireccional de 3 ruedas.

Omnidireccional de 4 ruedas: Se conforma en una aproximación similar. En la figura 2.5 [25] se aprecia el modelo de este vehículo. Su característica y ventaja es que para realizar un movimiento existen varias combinaciones posibles de velocidades para poder efectuarlo y si existe un deslizamiento en una rueda es posible detectarlo y tratar el problema.

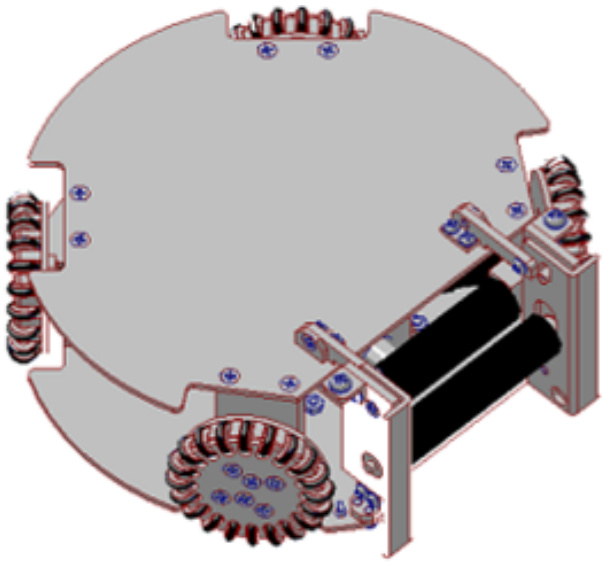

Figura 2.5: Estructura del vehículo omnidireccional de 4 ruedas.

Algunas aplicaciones de estos tipos de vehículos son:

- Industria: Pueden servir como montacargas y transporte.
- Hogar: Se pueden utilizar para robots que ayuden a personas con discapacidades.
- Entretenimiento: Una de las aplicaciones más comunes de este tipo de robots consiste en ser utilizados para lucha de robots o fútbol entre ellos.

2.7. Modelo del vehículo omnidireccional de tres ruedas

En este trabajo se utiliza el vehículo omnidireccional de tres ruedas. Para determinar el modelo del sistema se parte de un sistema de referencia móvil $\dot{\eta}_m$, donde sus velocidades dependen de las velocidades $\dot{x}_m$, $\dot{y}_m$ y de su velocidad angular $\dot{\phi}_m$; buscando transportarlas al sistema de referencia inercial $\dot{\eta}_w$, el cual depende de las velocidades $\dot{x}_w$, $\dot{y}_w$ y $\dot{\phi}_w$. Para pasar de un sistema a otro se utiliza la matriz de rotación R_m^w, que permite llevar del sistema móvil al sistema inercial, de tal forma que

$$\dot{\eta}_w = R_m^w \dot{\eta}_m \tag{2.1}$$

Donde

$$R_m^w = \begin{pmatrix} cos\phi_w & -sin\phi_w & 0 \\ sin\phi_w & cos\phi_w & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

siendo ϕ_w el ángulo de giro del vehículo omnidireccional, como se puede observar en la figura 2.6.

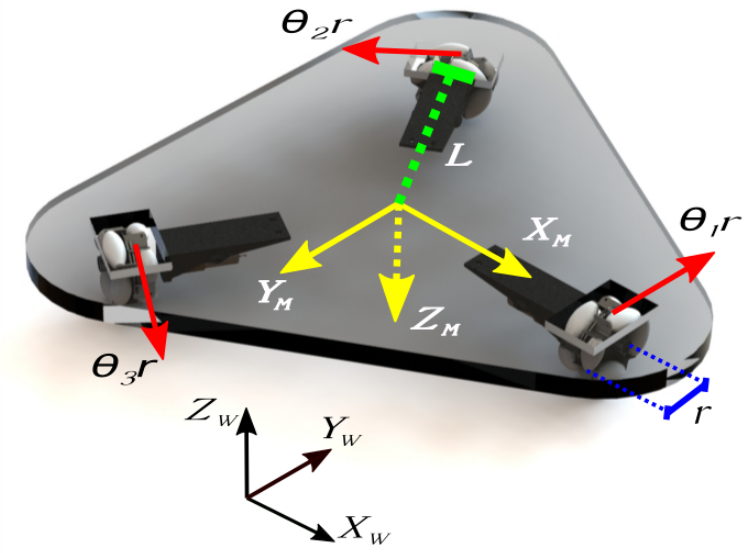

Figura 2.6: Diagrama físico del vehículo omnidireccional.

Sin embargo, para generar las velocidades mencionadas es necesario conocer la velocidad angular de cada una de las llantas, la cual se puede apreciar en la figura 2.7, en donde se tiene un radio de la rueda R_r y la velocidad angular de la rueda $\dot{\theta}$.

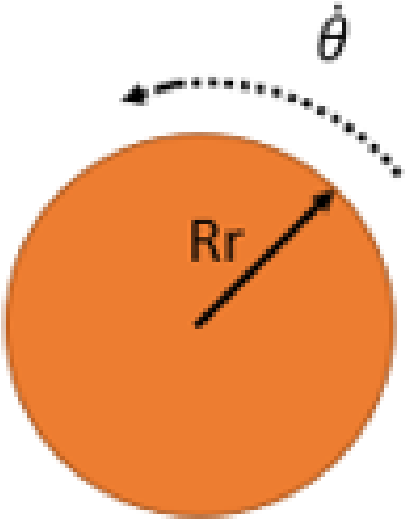

Figura 2.7: Rueda del vehículo omnidireccional.

Al relacionar las velocidades de la rueda con las del vehículo se obtiene

$$\begin{pmatrix} \dot{\theta}_1 * R_r \\ \dot{\theta}_2 * R_r \\ \dot{\theta}_3 * R_r \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{3} & \frac{1}{2} & L \\ 0 & -1 & L \\ -\frac{1}{2}\sqrt{3} & \frac{1}{2} & L \end{pmatrix} \begin{pmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\phi}_m \end{pmatrix} \tag{2.3}$$

Donde la primera matriz corresponde a la de velocidades de las ruedas $\dot{\theta}$ y la variable l es la distancia de las ruedas hasta el centro de masa del vehículo, de tal manera que:

$$\dot{\eta}_m = M^{-1}\dot{\theta} \tag{2.4}$$

Por lo que el sistema completo esta denotado por:

$$\dot{\eta}_w = R_m^w M^{-1}\dot{\theta} \tag{2.5}$$

Sin embargo, el modelo para hacer girar la llanta se determina por medio del sistema eléctrico del motor como se muestra en la figura 2.8, en donde V es el voltaje aplicado, R es la resistencia de armadura, L la inductancia de armadura, i es la corriente de armadura, V_c es la fuerza electromotriz, $J - m$ es la masa del motor, θ_m es el ángulo de giro antes de los engranes, N es la relación entre los engranes y θ es el giro de las llantas.

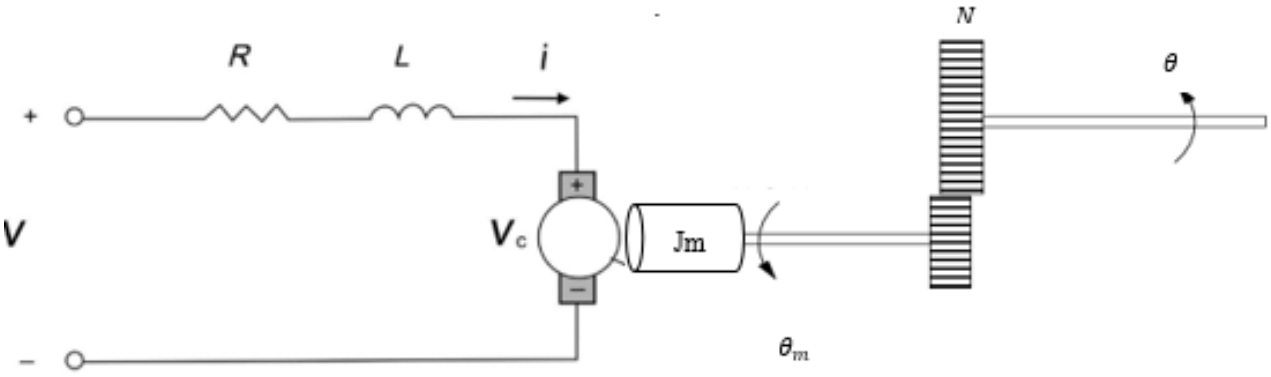

Figura 2.8: Diagrama eléctrico y mecánico del motor de la rueda del vehículo omnidireccional.

Donde las ecuaciones resultantes del modelo son:

$$V = Ri + L\frac{di}{dt} + V_c \tag{2.6}$$

$$J_m\ddot{\theta}_m = k_a i - f(\dot{\theta}_m) - \frac{\tau}{N} \tag{2.7}$$

Donde k_a es una constante, τ es el par generado después de los engranes y $f(\dot{\theta}_m)$ es la fricción en función de la velocidad angular antes de los engranes. Sustituyendo la ec. 2.6 en la ec. 2.7 y haciendo $V_c = k_b$ y $\dot{\theta}_m = n\theta_m$ resulta el modelo:

$$J_m\ddot{\theta} = \frac{k_a V}{RN} - \frac{k_a k_b \dot{\theta}}{R} - \frac{f(\dot{\theta}_m)}{N} - \frac{\tau}{N^2} \tag{2.8}$$

Para simplificar el análisis fundamental del sistema se igualan las fricciones $-\frac{f(\dot{\theta}_m)}{N} - \frac{\tau}{N^2} = 0$, de tal forma que resulta la expresión reducida al hacer estos términos cero y al dividir la expresión por J_m, obteniendo entonces:

$$\ddot{\theta} = \frac{k_a V}{RNJ_m} - \frac{k_a k_b \dot{\theta}}{RJ_m} \tag{2.9}$$

Ahora, se engloban las constantes de forma que $b_m = \frac{k_a}{RNJ_m}$ y $a_m = \frac{k_a k_b}{RJ_m}$ además si $\dot{\theta}$ es la velocidad angular entonces se puede considerar que $\omega = \dot{\theta}$ por lo que $\dot{\omega} = \ddot{\theta}$ y entonces la ecuación resultante es:

$$\dot{\omega} = -a_m\omega + b_m v \tag{2.10}$$

Donde al obtener la transformada de Laplace inversa se obtiene:

$$\Omega(s)s = -a_m\Omega(s) + b_m V(s) \tag{2.11}$$

De tal manera que la función de transferencia resultante es:

$$H(s) = \frac{\Omega(s)}{V(s)} = \frac{b_m}{s+a_m} \tag{2.12}$$

2.8. Control colaborativo basado en eventos

Una red de comunicaciones se compone por un conjunto de agentes que colaboran para alcanzar un fin en específico, creando así un sistema multi-agente. Cada elemento que forma parte de este sistema debe de contar con sus propias variables de estado, así como de sus propias dinámicas y poseen la capacidad de intercambiar dichos valores con los demás nodos de la red para alcanzar el objetivo común de esta. Entonces, el esquema de control colaborativo consiste en que dichas variables alcancen el mismo valor deseado (por ejemplo, posicionarse en un determinado conjunto de coordenadas) usando técnicas y protocolos de sincronización denominados consenso. Al momento de diseñar un control para una red VANET es importante tener cuidado, ya que todos los protocolos implementados deben ser distribuidos en cada agente como una estrategia de control distribuido, permitiendo saber la información de sus nodos cercanos según la topología que se esté formando [26]. Entonces para desarrollar el control colaborativo basado en eventos es necesario definir la topología de comunicaciones, la función de control colaborativo y la función de evento que disparará dicho control.

2.8.1. Teoría de grafos

Se le conoce como un grafo a la representación gráfica de la forma en la que se conectan los nodos en un sistema multi-agente. Está compuesto por vértices (nodos) y arcos (unión entre dos nodos), como se puede apreciar en la figura 2.9. En un ambiente de un sistema multi-agente de nodos VANET, físicamente los vértices representarán al vehículo no tripulado, mientras que los arcos hacen referencia a las comunicaciones que existen entre ellos a través de la forma de comunicación establecida.

De acuerdo a [14] de un grafo es posible obtener tres matrices importantes:

- $A = [a_{ij}]$, es la matriz de conectividad la cual será cuadrada de dimensión N, con N igual al número de agentes que conforman la red. Si existe un enlace del nodo i al nodo j (donde i y j toman valores desde 1 a N), entonces el valor del elemento de la matriz será 1. En caso contrario, dicho elemento tendrá el valor de 0. Como ejemplo de este caso, la matriz correspondiente al grafo bidireccional de la figura

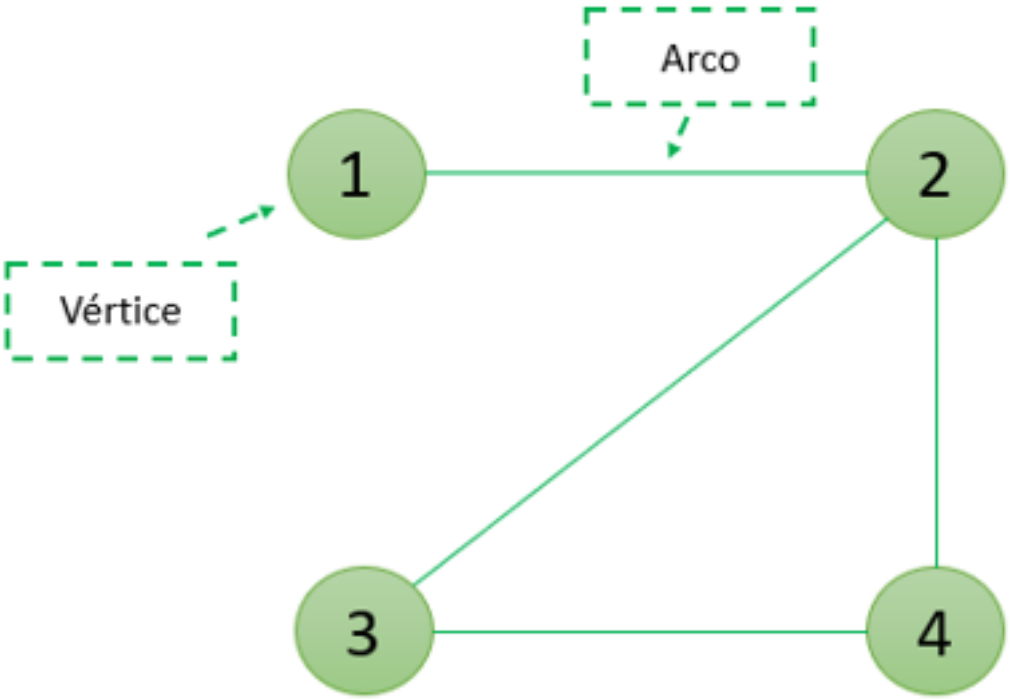

Figura 2.9: Ejemplo de grafo con 4 agentes.

2.9 es el siguiente

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{2.13}$$

- D es una matriz diagonal cuyos pesos de los valores de la diagonal, corresponden al número de enlaces que tiene cada nodo, por lo que la matriz D de la figura 2.9 corresponde a:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \tag{2.14}$$

- La matriz L corresponde a la diferencia $L = A - D$, siendo esta:

$$L = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -3 & 1 & 1 \\ 0 & 1 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{pmatrix} \tag{2.15}$$

.

En resumen, se define un grafo $G = (V, E)$ que representa el conjunto de nodos $V =$

$\{v_1, \cdots, v_N\}$ y el conjunto de arcos denotados por E. La matriz de conectividad $A \in \mathbb{R}^{N\times N}$ es aquella en la cual sus elementos están dadas por $a_{ij} = 1$ si $(i,j) \in E$ y $a_{ij} = 0$ en caso contrario. La matriz D es diagonal y sus elementos d_i corresponden al conjunto de vecinos del nodo i denotados por $N_i = \{j \in V : (i,j) \in E\}$. La diferencia entra la matriz diagonal y la matriz de conectividad resulta en la matriz laplaciana $L(G) = D - A$ siendo esta semi-definida positiva para grafos no dirigidos. El vector $1 = (1, \cdots, 1)^T \in \mathbb{R}^N$ es un eigenvalor asociado con $\lambda_1(G) = 0$.

2.8.2. Consenso

Para conseguir la sincronización de los diferentes nodos que conforman una red de comunicaciones, es necesario recurrir a una técnica denominada consenso, en donde los agentes llegan a un acuerdo que permite que sus variables alcancen un valor común o un valor deseado.

Matemáticamente, el consenso se define como:

$$x_i(n) = \frac{1}{N}\sum_{j=1}^{N} x_j(0), \forall i \tag{2.16}$$

Donde tal fórmula representa el promedio de los valores de los estados de todos los nodos que conforman la red.

2.8.3. Modelado del sistema y control distribuido

Para implementar el sistema multi-agente se considera un conjunto de N robots móviles conectados mediante el grafo G con vértices $V = \{1, 2, \cdots, N\}$. Se define entonces por medio de la representación de espacio de estados el vector $x_{i,s} = \begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \end{pmatrix}^T \in \mathbb{R}^3$ el cual se conforma por los elementos de las posiciones $\dot{x_m}$ y $\dot{y_m}$ así como de la orientación $\dot{\phi_m}$ respectivamente, las cuales fueron presentadas en la ec. (2.3). También, se define $m_{i,s} = \begin{pmatrix} m_{i,1} & m_{i,2} & m_{i,3} \end{pmatrix}^T$ como el conjunto de los estados $x_{i,s}$ la última vez que ocurrió un evento, donde $i \in V$ y $s \in \{1, 2, 3\}$. De igual manera, se introduce $u_{i,s} = \begin{pmatrix} u_{i,1} & u_{i,2} & u_{i,3} \end{pmatrix}^T \in \mathbb{R}^3$, el cual es el vector que conforma un control por linealización exacta, permitiendo así que el modelo cinemático del vehículo omnidireccional resulte en:

$$\begin{aligned} \dot{x}_{i,1} &= u_{i,1}cos(x_{i,3}) - u_{i,2}sin(x_{i,3}) \\ \dot{x}_{i,2} &= u_{i,1}sin(x_{i,3}) + u_{i,2}cos(x_{i,3}) \\ \dot{x}_{i,3} &= u_{i,3} \end{aligned} \tag{2.17}$$

en donde el control por linealización exacta $u_{i,s}$ que se desea para este sistema se define como:

$$\begin{aligned} u_{i,1} &= cos(x_{i,3}) * r_{i,1} + sin(x_{i,3}) * r_{i,2} \\ u_{i,2} &= -sin(x_{i,3}) * r_{i,1} + cos(x_{i,3}) * r_{i,2} \\ u_{i,3} &= r_{i,3} \end{aligned} \tag{2.18}$$

De tal manera que el modelo del i-ésimo robot resultará en un sistema delimitado por simples integradores de la forma:

$$\begin{aligned} \dot{x}_{i,1} &= r_{i,1} \\ \dot{x}_{i,2} &= r_{i,2} \\ \dot{x}_{i,3} &= r_{i,3} \end{aligned} \tag{2.19}$$

donde $r_{i,s} = \begin{pmatrix} r_{i,1} & r_{i,2} & r_{i,3} \end{pmatrix}^T$ representa la estrategia de control colaborativa, basada en el consenso para llegar a un punto común la cual se definirá como:

$$r_{i,s} = r_{i,s}^{\alpha} + r_{i,s}^{\gamma} \tag{2.20}$$

donde $r_{i,s}^{\gamma}$, corresponde al control colaborativo por consenso y $r_{i,s}^{\alpha}$ corresponde a un control para evadir colisiones entre agentes.

El control basado en consenso se define como:

$$r_{i,s}^{\gamma} = \kappa_1 \left[\sum_{j \in N} (m_{j,s} - m_{i,s}) + g_i(\xi_{0,s} - m_{i,s}) \right] \tag{2.21}$$

donde $s \in \{1, 2, 3\}$, $\kappa > 0$, y g_i, siendo la ganancia de fijación que determina si el agente i estará siguiendo a un líder virtual (con $g_i = 1$ en caso de ser así y con $g_i = 0$ en caso contrario) y $\xi_{0,s}$ denota la referencia o trayectoria a seguir por el líder virtual. Para generar formaciones y que los agentes no lleguen al mismo punto, se agrega un elemento para generar la agrupación deseada mediante un conjunto F de locaciones asociadas, dado por:

$$F = \{\zeta_1, \zeta_2, ...\zeta_N\},\ \zeta_i \in \mathbb{R}^3,\ i = 1, .., N \tag{2.22}$$

donde

$$\|\zeta_i - \zeta_j\| = \varrho_{ij} \tag{2.23}$$

convirtiendo así la ec. (2.21) en:

$$r_{i,s}^{\gamma} = \kappa \left[\sum_{j \in N} (m_{j,s} - m_{i,s}) - (\zeta_{j,s} - \zeta_{i,s}) + g_i(\xi_{0,s} - m_{i,s}) \right] \tag{2.24}$$

donde $d_i = (\zeta_{j,s} - \zeta_{i,s})$ representa la distancia entre el i-ésimo agente y el j-ésimo vecino.

2.8.4. Control disparado por eventos

En gran parte de la literatura referente al control de sistemas de diferentes naturalezas, suelen implementarse estrategias de control basadas en el tiempo, es decir, se monitorea constantemente una variable y con base en esta se genera una señal de control para obtener una cierta salida deseada. Sin embargo, esto produce que los dispositivos que se encarguen entregar una señal de control estén en constante funcionamiento, lo que ocasiona que se consuma energía y constante cálculo computacional. Una alternativa para esto consiste en el control basado por eventos. Esta estrategia permite que se genere una señal de control para la planta a controlar únicamente cuando se produce un evento determinado, ya sea cuando se detecte un error que sobrepase cierto umbral, cuando se incremente abruptamente una variable medida, cuando se genere una interrupción, etc., de tal manera que al hacer esto, es posible ocupar el dispositivo de control para otras tareas o simplemente inducirlo a un modo de ahorro de energía, lo que es sumamente importante cuando se tratan, por ejemplo, de vehículos autónomos que requieren ahorrar la mayor cantidad de energía posible, así como también permite reducir la cantidad de mensajes enviados a través de la red que estos forman, logrando una mejor utilización de los recursos como lo es el ancho de banda. Entonces, la naturaleza del control pasa de ser completamente síncrona, a ser semi-asíncrona, ya que a pesar de que la señal de control se genera únicamente con la ocurrencia de un evento, es necesario contar con una frecuencia de muestreo por parte del subsistema encargado de detectar el evento.

Entonces para un sistema con un control basado en eventos, se integra una función de evento e_i.

La función de evento para esta implementación se define por:

$$e_{i,s} = \begin{cases} 1 & \text{if } |m_{i,s} - x_{i,s}| - \delta \geq 0 \\ 0 & \text{en caso contrario} \end{cases} \tag{2.25}$$

donde $i \in V$, $s \in \{1, 2, 3\}$, $\delta > 0$ y se encuentra compuesta por el valor de las posiciones actuales del nodo $i, x_{i,s}$ y una memoria $m_{i,s}$ del valor de $x_{i,s}$ la última vez que $e_{i,s}$ resultó en un valor positivo, permitiendo así que i se comunique con j solo cuando $(e_{i,s} > 0)$ y en caso contrario, que no se habilite la transmisión de información, siendo δ el umbral que determina cuando se genera un evento; mientras mayor sea δ menor será la cantidad de eventos. Con base en la función de evento, el sistema multi-agente cuenta con un control distribuido en donde el agente i usa sus propias variables de estado y la información obtenida por los agentes vecinos j, que son almacenadas en las memorias m_i y m_j respectivamente, resultando en la función presentada en la ec. (2.20). Entonces la función definida en (2.25), que depende de $x_{i,s}$ y $m_{i,s}$, permite que el i-ésimo robot distribuya su estado y actualice la señal de control mediante:

$$e_i(x_i, m_i) = e_{i,1} \wedge e_{i,2} \wedge e_{i,3} \tag{2.26}$$

donde el símbolo $\wedge$ denota una operación lógica OR de tal manera que el valor de la memoria $m_{i,s}$ se actualizará de la forma:

$$m_{i,s} = \begin{cases} x_{i,s} & \text{for } e_{i,s} = 1 \\ m_{i,s} & \text{for } e_{i,s} = 0 \end{cases} \tag{2.27}$$

2.8.5. Evasión de colisiones entre agentes

En [27] se introduce un método para implementar un algoritmo de evasión de colisiones. Este algoritmo considera un comportamiento en "flocking" de los agentes que conforman el grupo. Cada nodo es considerado como un agente de segundo orden, usando los valores de sus posiciones para el término que previene las colisiones, mientras que las velocidades son utilizadas para el término que permite el consenso. Sin embargo, en este trabajo los agentes son definidos como un sistema de primer orden y, por tanto, el consenso es calculado usando también las posiciones y orientaciones de los vehículos. El término para la evasión de colisiones entre agentes, según [27], es u_i^α. Esta función está basada en una función de campos potenciales atractiva/repulsiva que permite calcular un término adicional cuando se puede dar una colisión. Los vehículos cercanos se denominan α-agentes los cuales, en consecuencia del enfoque basado en eventos, tienen un conjunto de posiciones previas $\hat{m}_j = \begin{pmatrix} m_{j,1} & m_{j,2} \end{pmatrix}^T \in \mathbb{R}^2$ mientras que el conjunto de posiciones actuales del nodo son $\hat{m}_i = \begin{pmatrix} m_{i,1} & m_{i,2} \end{pmatrix}^T \in \mathbb{R}^2$. Los nodos considerados como α-agentes son:

$$N_i = j \in V : \|\hat{m}_j - \hat{m}_i\| < r \tag{2.28}$$

donde r es el rango de interacción entre agentes. La ley de control introducida para la evasión de colisiones con α-agentes es:

$$r_{i,s}^\alpha = c_\alpha \textstyle\sum_{j \in N_i} \phi_\alpha(\|\hat{m}_j - \hat{m}_i\|_\sigma)\hat{n}_{ij} \tag{2.29}$$

donde la norma sigma $\|z\|$ de la matriz z es calculada por:

$$\|z\|_\sigma = \tfrac{1}{\epsilon}[\sqrt{1 + \epsilon \|z\|^2} - 1] \tag{2.30}$$

además $c_\alpha > 0$ y $\hat{n_{ij}}$ es un vector definido como:

$$\hat{n}_{ij} = \tfrac{\hat{m}_j - \hat{m}_i}{\sqrt{1 + \epsilon \|\hat{m}_j - \hat{m}_i\|}} \tag{2.31}$$

con $\epsilon > 0$. El término ϕ_α es una función de acción para $z = \|\hat{m}_j - \hat{m}_i\|$ que se desvanece

siempre que los α-agentes se encuentran lo suficientemente lejos el uno del otro. La función de acción se define como:

$$\phi_\alpha = \rho_h(z/r_\alpha)\phi(z - d_\alpha) \tag{2.32}$$

donde $\rho_h(z)$ es una función "bump" que consiste en una función escalar que varia suavemente desde 0 a 1 y se define por:

$$\rho_h(z) = \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2}[1 + cos(\pi\frac{(z-h)}{1-h})] & z \in [h, 1] \\ 0 & de\, otra\, manera \end{cases} \tag{2.33}$$

y $\phi(z)$ es una función sigmoidal desigual definida por:

$$\phi(z) = \tfrac{1}{2}[(a + b)\sigma_1(z + c) + (a - b)] \tag{2.34}$$

donde:

$$\sigma_1 = z/\sqrt{1 + z^2} \tag{2.35}$$

con $0 < a \leq b, c = a - b/\sqrt{4ab}$ y donde $r_\alpha = \|r\|_\sigma$ y $d_\alpha = \|d\|_\sigma$ son la σ-norma del radio r y de la distancia d entre los agentes respectivamente. El valor d se obtiene con:

$$d = \|\hat{m}_j - \hat{m}_i\| \tag{2.36}$$

Finalmente, la ec. (2.20) con base en la ec. (2.21) y en la ec. (2.29) resulta:

$$r_{i,s} = \kappa\left[\sum_{j\in N}(m_{j,s} - m_{i,s}) - (\zeta_{j,s} - \zeta_{i,s}) + g_i(\xi_{0,s} - m_{i,s})\right] + c_\alpha \sum_{j\in N_i} \phi_\alpha(\|\hat{m}_j - \hat{m}_i\|_\sigma)\hat{n}_{ij} \tag{2.37}$$

Capítulo 3

Simulaciones

A lo largo del presente capítulo se plasman los resultados obtenidos en las diferentes simulaciones que fueron realizadas con el fin de cumplir el objetivo de simular la red VANET considerando tanto el sistema de control como el sistema de comunicaciones de cada vehículo no tripulado.

3.1. Simulación del vehículo omnidireccional

Con base en las ecuaciones obtenidas del modelo del vehículo omnidireccional de tres ruedas, fue posible llevar a cabo una simulación del sistema en Simulink, en donde se diseñó el diagrama de bloques mostrado en la figura 3.1. En esta simulación se construyó el sistema de tal forma que en el primer bloque se agregó un conjunto coordenado que sirvió como referencia para mover el vehículo omnidireccional. Se considera que el vehículo avanza hacia al frente (movimiento en eje X) por lo que las velocidades del sistema de referencia móvil requeridas son $\dot{x}_m = 1$; $\dot{y}_m = 0$; $\dot{\phi}_m = 0$.

En el segundo bloque se multiplican las posiciones deseadas por la matriz M como se muestra en la ec. (2.3) para, de esta manera, obtener el vector de velocidades angulares de las ruedas multiplicadas por sus respectivos radios $\dot{\theta}_1 R_r$, $\dot{\theta}_2 R_r$ y $\dot{\theta}_3 R_r$.

En el siguiente bloque se tiene un controlador proporcional-integral con el fin de aproximar las velocidades angulares entregadas de los motores a los valores deseados. Al realizar la parametrización del vehículo aplicando una señal escalón y midiendo la velocidad angular de las ruedas se obtienen $a_m = 11{,}11$ y $b_m = 617{,}21$, mientras que los valores del controlador PI son $P = 0{,}1827$ e $I = 0{,}1239$.

La salida corresponde al conjunto de velocidades angulares pero después de que fuera aplicado el control correspondiente $\dot{\theta}_1 R_r$, $\dot{\theta}_2 R_r$ y $\dot{\theta}_3 R_r$.

En el último bloque entran las velocidades angulares de las ruedas para ser multiplicadas por la matriz M inversa y después por la matriz de rotación que se encuentra en función al ángulo ϕ_w, razón por la cual la salida de este bloque conformado por las

velocidades con respecto al sistema de referencia inercial $\dot{x}_m$, $\dot{y}_m$ y $\dot{\phi}_m$ entra al integrador (únicamente la velocidad angular es la que entra a este bloque).

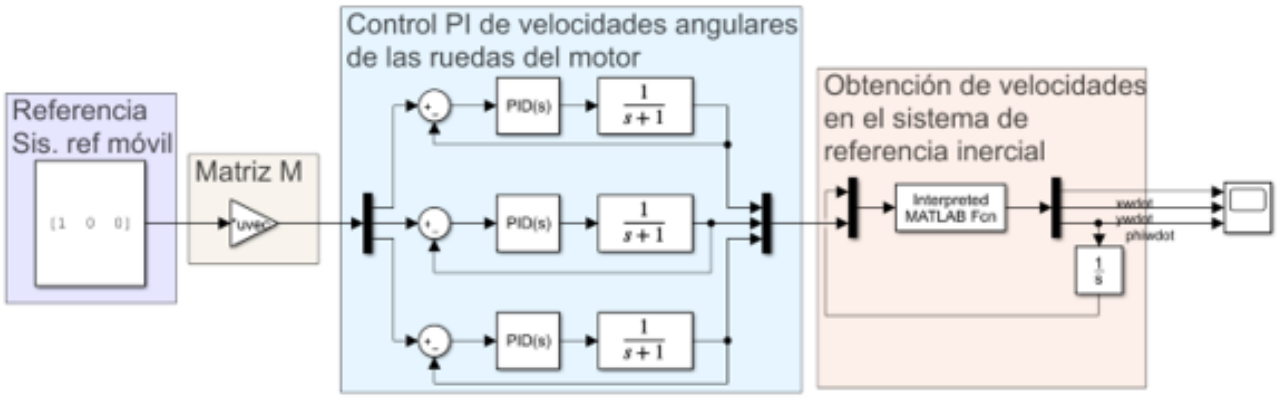

Figura 3.1: Diagrama del modelo del vehículo omnidireccional en Simulink.

Entonces la respuesta obtenida de las velocidades $\dot{x}_m$, $\dot{y}_m$ y $\dot{\phi}_m$ son observadas en las figuras 3.2, 3.3 y 3.4. En la primera se puede apreciar que efectivamente la salida $\dot{x}_m$ tiende a la referencia proporcionada que fue de 1 m/s, consiguiendo establecerse en un tiempo menor a 5 segundos. En cuestión a las figuras 3.3 y 3.4 se aprecian oscilaciones que son despreciables, ya que están en el orden de 10^{-16} por lo que no afectan de manera significativa y en realidad corresponden con los valores proporcionados por las referencias.

De igual forma se crea un modelo en dos dimensiones del movimiento del vehículo y se le asigna movimiento. En la figura 3.5 se observa el desplazamiento hacia en frente con una velocidad de 1 m/s y se corre la simulación por 4 segundos.

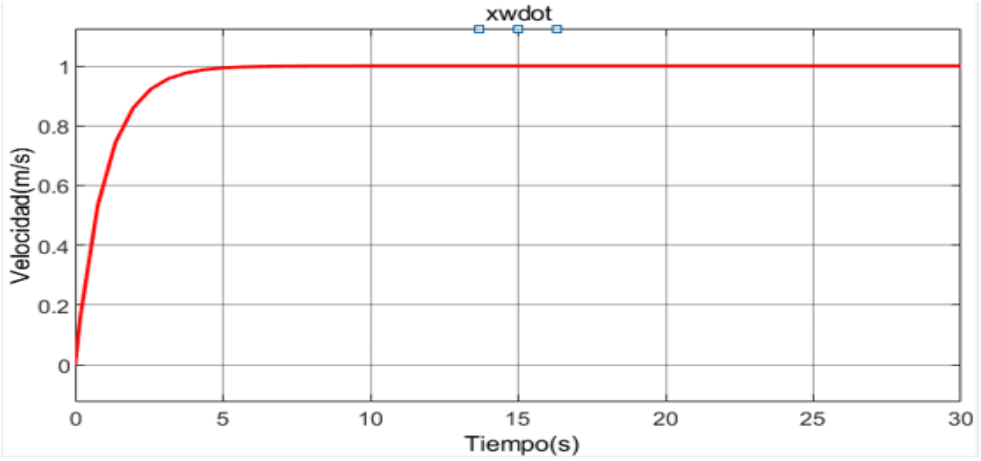

Figura 3.2: Respuesta de la velocidad $\dot{x}_m$ del vehículo omnidireccional.

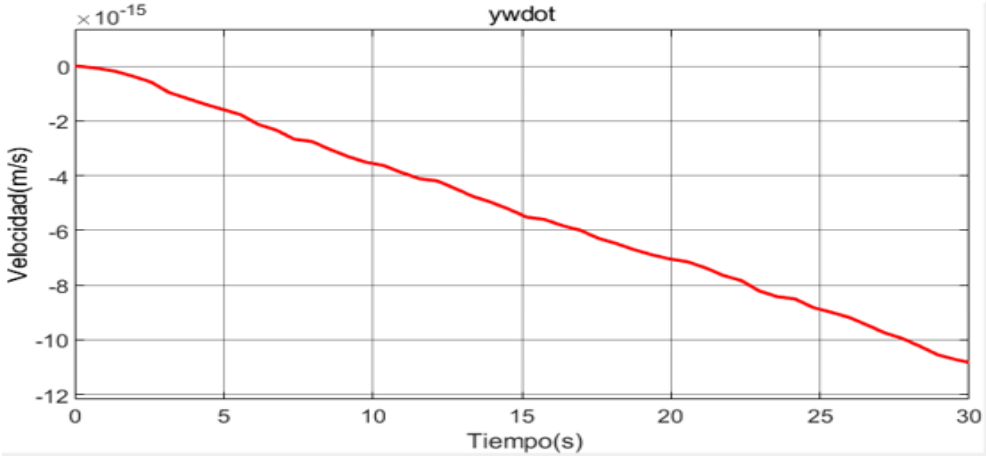

Figura 3.3: Respuesta de la velocidad $\dot{y}_m$ del vehículo omnidireccional.

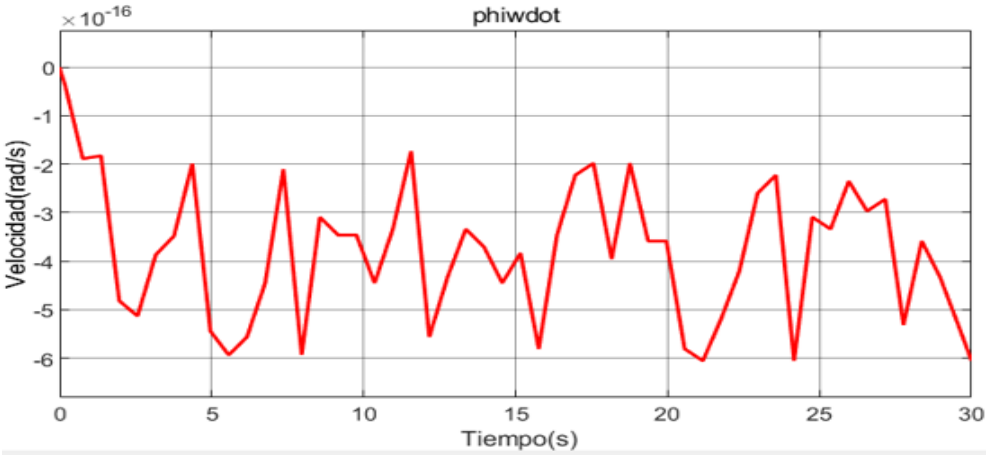

Figura 3.4: Respuesta de la velocidad $\dot{\theta}_m$ del vehículo omnidireccional.

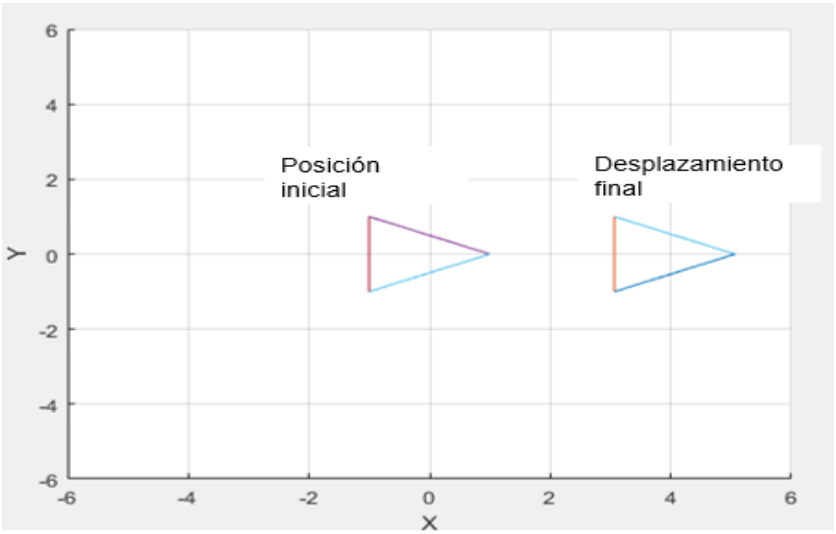

Figura 3.5: Desplazamiento frontal del vehículo.

3.2. Control colaborativo disparado por eventos

En esta sección se realizó una simulación del sistema de control colaborativo con una red compuesta por 4 nodos con el fin de implementar una estrategia de control colaborativo que realiza consenso para ejecutar formaciones, utilizando una técnica de disparo por eventos. En la figura 3.6 se observa el diagrama de bloques en Simulink representando el grafo mostrado en la figura 2.9 donde $V = \{1, 2, 3, 4\}$. Este modelo se compone de 4 secciones principales, las cuales corresponden a cada uno de los agentes que conforman el grafo, en los cuales entran las variables de estado $x_{i,s}$ y las memorias $m_{i,s}$ donde $i \in V$ y $s \in \{1, 2, 3\}$. Dentro de cada nodo se encuentra el control distribuido del sistema multi-agente, que contiene la ley descrita en la ec. (2.37) que permite realizar formaciones por medio del consenso evitando que se produzcan colisiones entre ellos al ejecutar la rutina. Posteriormente se introduce el modelo del vehículo mostrado en la ec. (2.17) junto con el control propuesto en la ec. (2.18) para que el sistema resulte en un simple integrador.

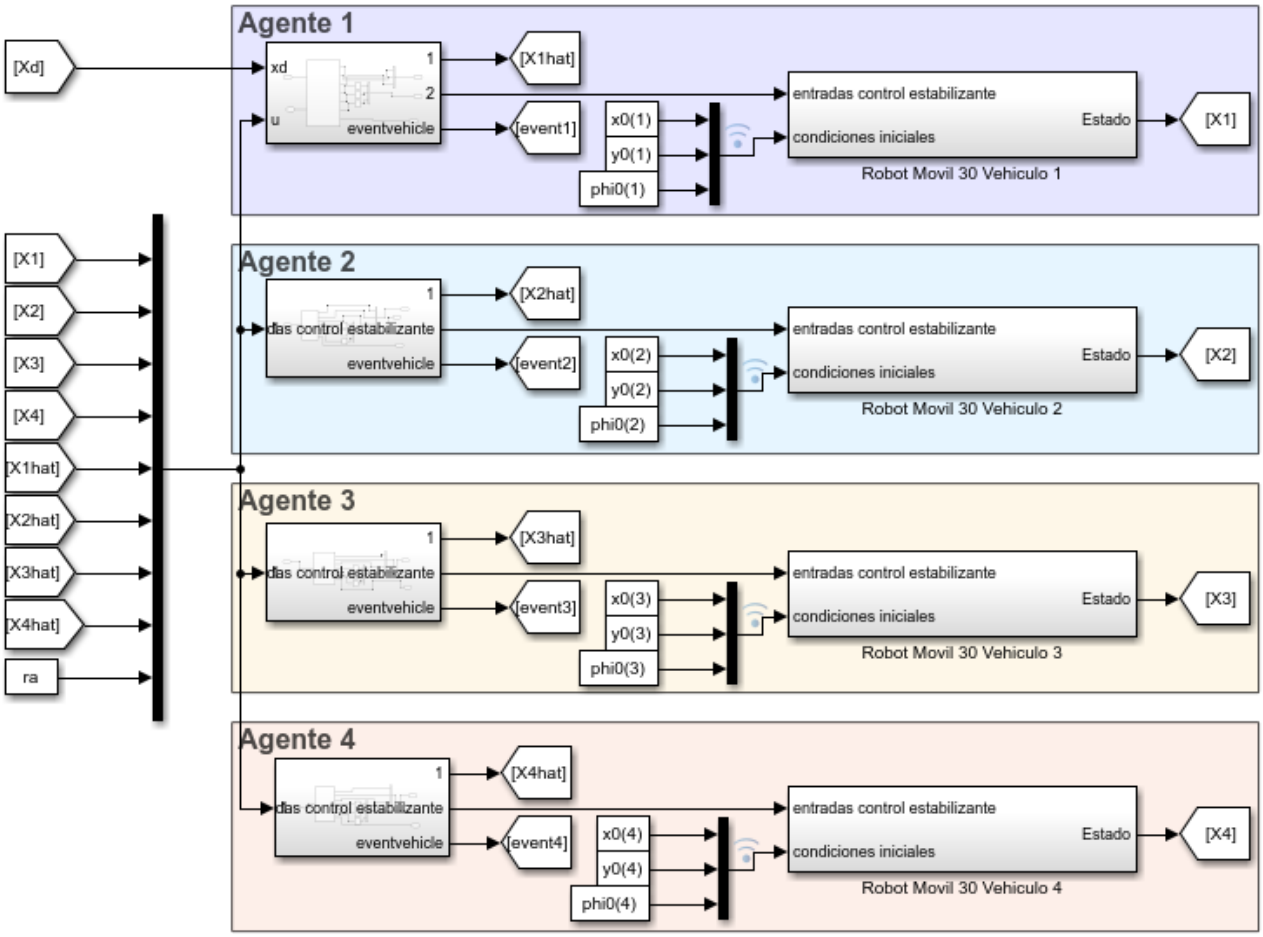

Figura 3.6: Diagrama del control colaborativo para 4 agentes en Simulink.

Para esta simulación se desea lograr la formación presentada en la figura 3.7, donde únicamente el agente 1 realizará consenso con un líder virtual que tiene designado el

conjunto de posiciones fijas de $x_{1,1} = 0$ y $x_{1,2} = 0$ por lo que este agente buscará llegar a esta posición mientras que el resto de los agentes mantendrán una distancia de 1m con respecto a sus vecinos, para completar la formación de la figura 3.7. Se considera un radio de seguridad para la evasión de colisiones de ra=0.6m y se generó de manera aleatoria el conjunto de posiciones iniciales por medio de una distribución uniforme como se puede apreciar en la Tabla 3.1. Se asignó una ganancia de $G = 10$ para el consenso entre el agente 1 y el agente virtual, mientras que la ganancia de consenso entre el resto de agentes es de $k = 3$ y la ganancia para la evasión de colisiones es de $k = 1$, todo esto con el fin de que la meta del grupo sea llegar a la posición delimitada por el agente virtual, sin que el consenso ni la evasión de colisiones desvíen por completo al grupo de dicho objetivo. El umbral de activación para los eventos es de $\delta = 0{,}01$, los pasos de la simulación son de 0.05 con un solver ode4 Runge-Kutta y el tiempo de duración de la rutina es de 20 segundos.

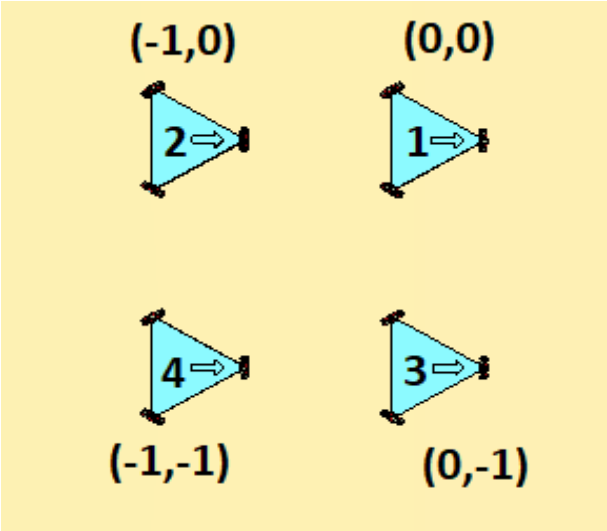

Figura 3.7: Formación deseada para los 4 agentes.

Tabla 3.1: Condiciones iniciales de las variables de estado $x_{i,s}$

Agente i	*1*	*2*	*3*	*4*
$x_{i,1}$	-1.59	-1.37	-0.35	0.24
$x_{i,2}$	-1.2	-1.65	1.74	-0.96
$x_{i,3}$	0.194π	0.555π	1.094π	1.916π

En las figuras 3.8, 3.9 y 3.10 se presenta la evolución de las variables de estado $x_{i,s}$ a lo largo de los 20 segundos de simulación en donde se observa que dichos valores llegan a las referencias deseadas que fueron establecidas en la figura 3.7 como se muestra en la Tabla 3.2, mientras que en la figura 3.11 se presentan los eventos que sucedieron en cada agente

donde si el valor es 1 este indica que ocurrió un evento, es decir, se requirió recalcular la señal de control para ajustar la salida al valor deseado y se activaron las comunicaciones, mientras que si el valor es 0 significa que no hay evento y por lo tanto no se tuvo la necesidad de calcular la señal de control ni transmitir su posición.

Tabla 3.2: Valores finales de las variables de estado $x_{i,s}$

Agente i	*1*	*2*	*3*	*4*
$x_{i,1}$	0	-1	0	-1
$x_{i,2}$	0	0	-1	-1
$x_{i,3}$	0	0	0	0

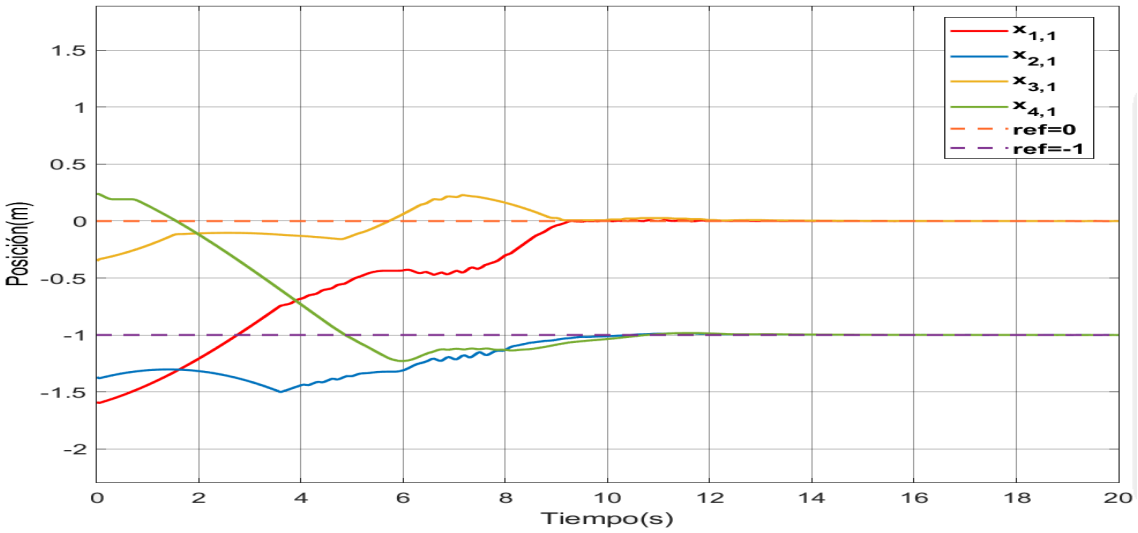

Figura 3.8: Resultados en $x_{i,1}$.

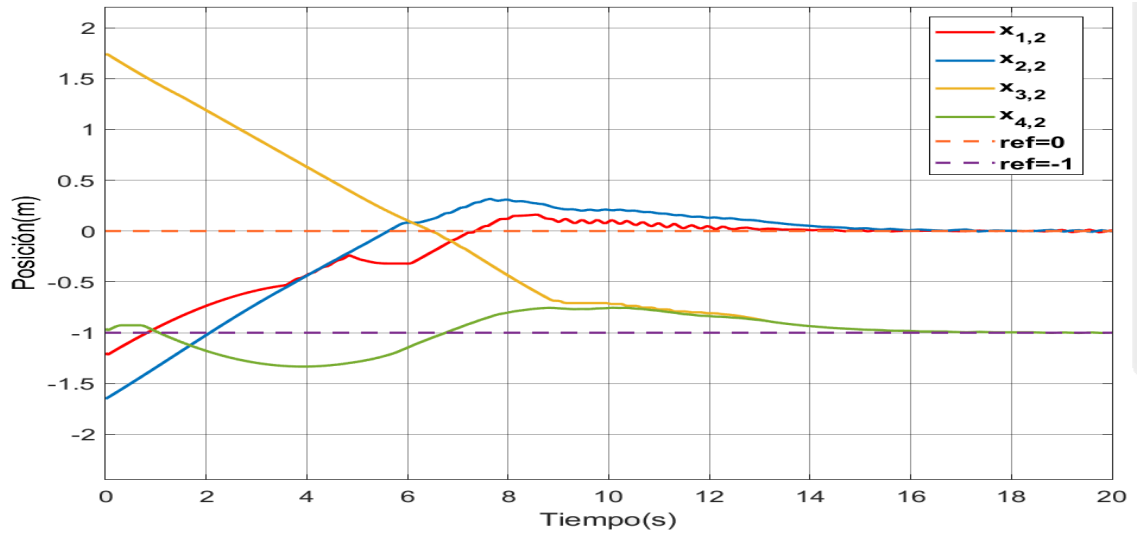

Figura 3.9: Resultados en $x_{i,2}$.

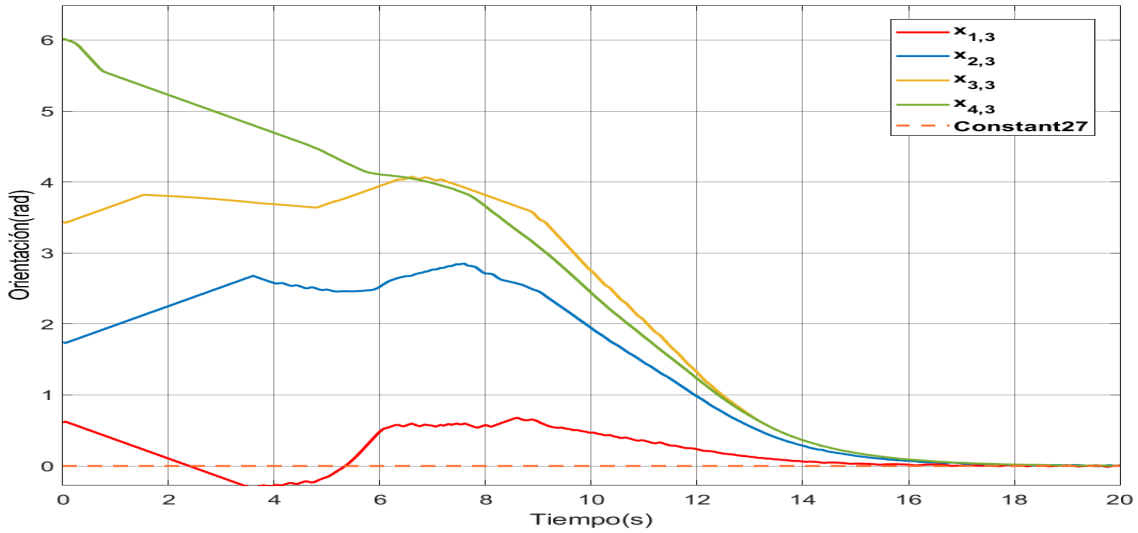

Figura 3.10: Resultados en $x_{i,3}$.

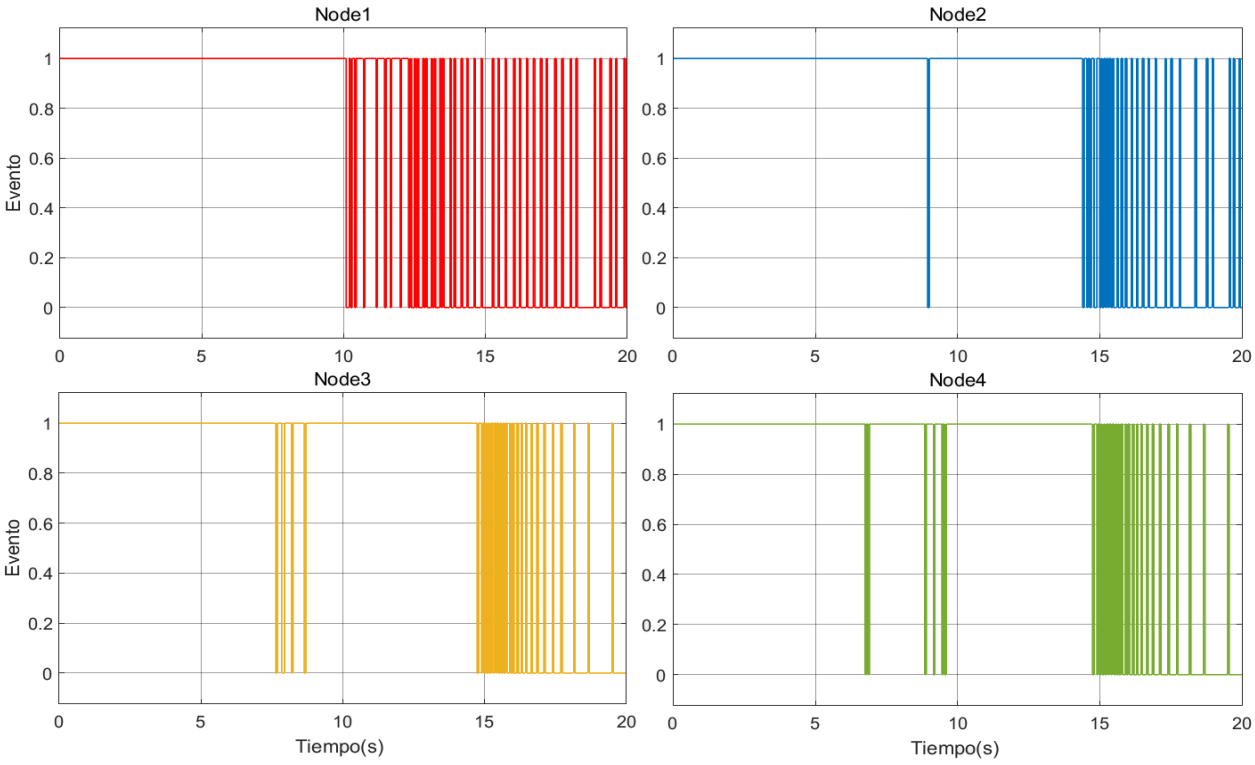

Figura 3.11: Eventos presentados en cada uno de los nodos.

Posteriormente, en la figura 3.12, se presenta en una gráfica XY la trayectoria que tuvo cada uno de los nodos a lo largo de los 20 segundos de simulación, presentando un gradiente de colores en las rutas que representa el tiempo de simulación, esto con el fin de mostrar que no ocurrió ninguna colisión entre los vehículos. De igual manera, los puntos azules corresponden a las posiciones iniciales de los nodos mientras que los puntos rojos corresponden a sus posiciones finales.

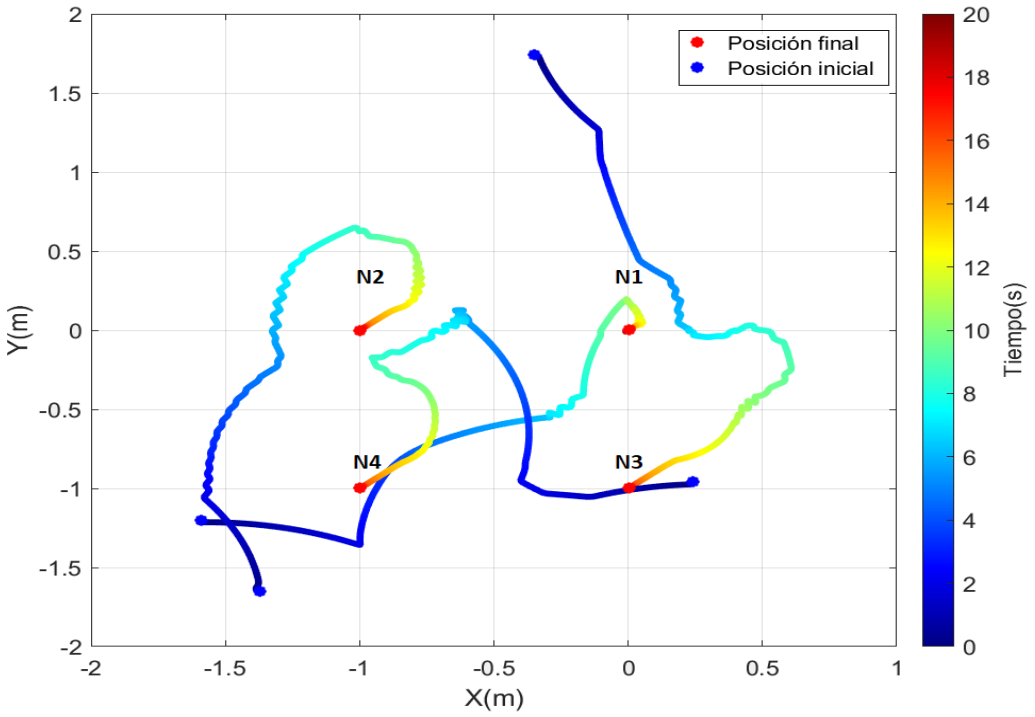

Figura 3.12: Trayectoria seguida de los 4 agentes.

Finalmente, se realiza un análisis de Monte Carlo, donde primero se realizan 30 simulaciones con condiciones iniciales generadas de manera aleatoria por medio de una distribución uniforme, y se calcula el MSE (error cuadrático medio, por sus siglas en inglés) mediante $MSE = \sqrt{(x - x_{ref})^2 + (y - y_{ref})^2}$ en donde "x" corresponde a la variable de estado $x_{i,1}$, mientras que "y" corresponde a $x_{i,2}$ y tanto $xref$ como $yref$ son las referencias deseadas del agente i. En la figura 3.13 se muestra la evolución del MSE del nodo 1 en las 30 iteraciones de la simulación, mientras que en la figura 3.14 se realiza un promedio de estas iteraciones y se delimita por medio de su desviación estándar.

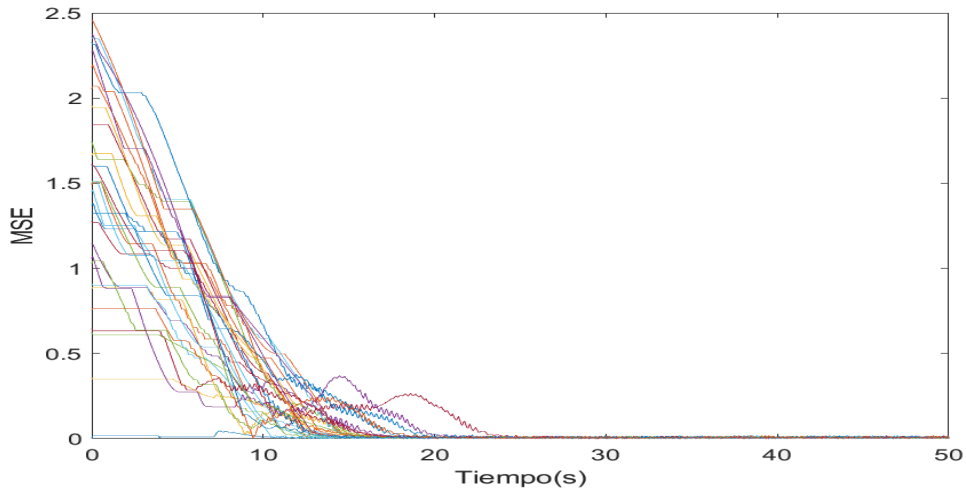

Figura 3.13: Error cuadrático medio en 30 simulaciones.

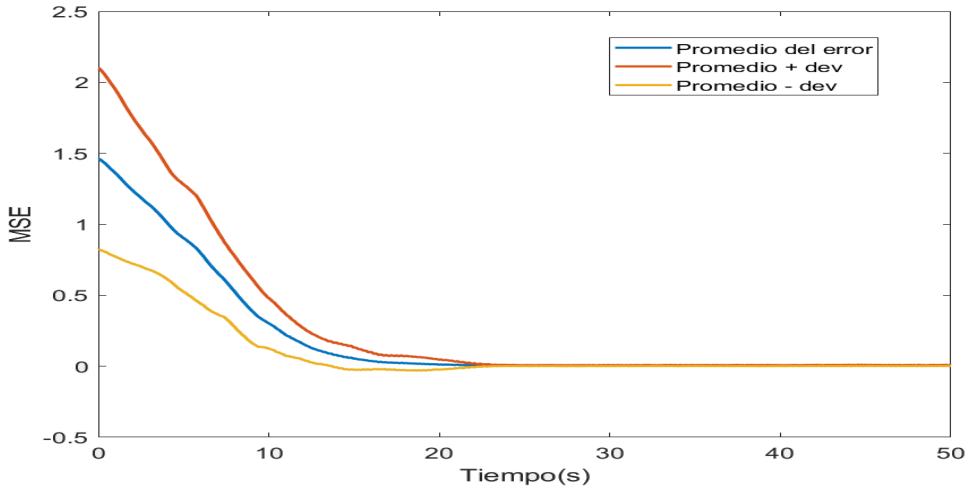

Figura 3.14: Promedio del error cuadrático medio y desviación estándar.

3.3. Simulación de la comunicación entre nodos VANET

En este apartado del capítulo se presentan las simulaciones previas a la implementación del sistema de comunicaciones.

A primera instancia y como prueba preliminar, con base en la topología presentada en capítulos previos, se efectuó la simulación de un algoritmo de enrutamiento en Matlab, donde se plantea la situación de comunicar en el grafo de 4 nodos la posición de estos para después realizar una formación en donde estos se posicionen de manera equidistante de acuerdo a los valores establecidos, considerando únicamente el área de comunicaciones.

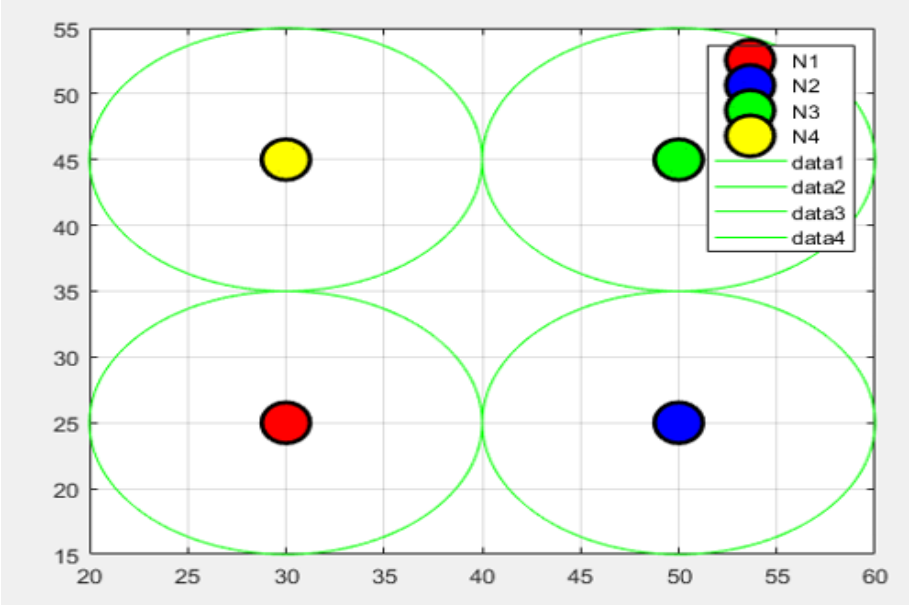

Figura 3.15: Simulación de topología de 4 nodos utilizando enrutamiento Bellman-Ford.

El resultado de la simulación se aprecia en la figura 4.2, en donde se generaron 4 nodos con posiciones aleatorias y se realizó un enrutamiento utilizando el algoritmo de Bellman-ford para hacer que los nodos se comunicaran entre ellos por medio de rutas óptimas entregando la información de sus posiciones deseadas. Este algoritmo permite que todos los nodos sepan la posición de los demás nodos que conforman la red. Al no tener los enlaces entre nodos una ponderación del costo, el enrutamiento se efectuó con base en la cantidad de saltos. En la simulación se requirió que el nodo 4 estuviera en la posición 30,45 (coordenadas en metros) por lo que los demás nodos se posicionaron en las coordenadas correspondientes para mantener la formación. La ruta óptima para el nodo 1 fue la ruta del nodo 4-2-1. Para el nodo 3 fue una conexión directa 4-3. De manera similar al caso anterior, para el nodo 2 solo se requirió un enlace 4-2.

3.4. Simulación de bloques de comunicación y control

En esta sección se proceden a implementar las técnicas de enrutamiento de nodos y estrategias de control colaborativo de manera conjunta de acuerdo a la topología presentada anteriormente. Para esto, se introduce el complemento de Matlab/Simulink denominado True time 2.0. Este consiste en un simulador basado para sistemas de control en tiempo real, el cual facilita la simulación conjunta de sistemas de control kernel en tiempo real, transmisiones de red y dinámicas de planta continuas [31]. Incluye características como:

- Simulación de controladores complejos, calendarización de tareas, comunicaciones alámbricas e inalámbricas de red.
- Posibilidad de escribir tareas como archivos M o funciones de C++.
- Bloques de red.
- Bloques de redes inalámbricas.

En la figura 3.16 se implementa una simulación en una red de 4 nodos en donde se utiliza un algoritmo de enrutamiento AODV para llegar a las posiciones deseadas utilizando los bloques de la herramienta True Time 2.0 para el sistema compuesto por 4 nodos VANET. A su vez, en la figura 3.17 se introducen los subsistemas que contienen el control colaborativo correspondiente de los vehículos omnidireccionales con la topología presentada.

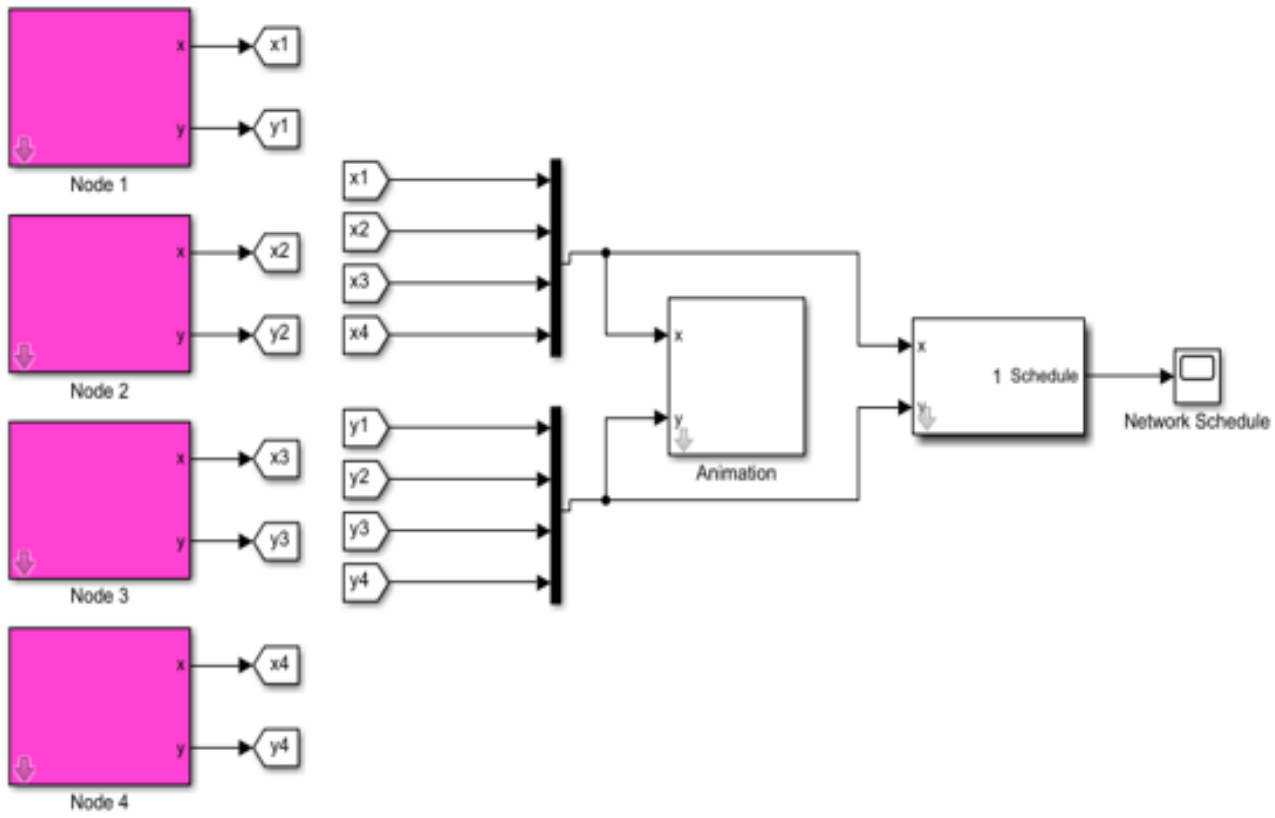

Figura 3.16: Diagrama de bloques de una red de comunicaciones de 4 nodos.

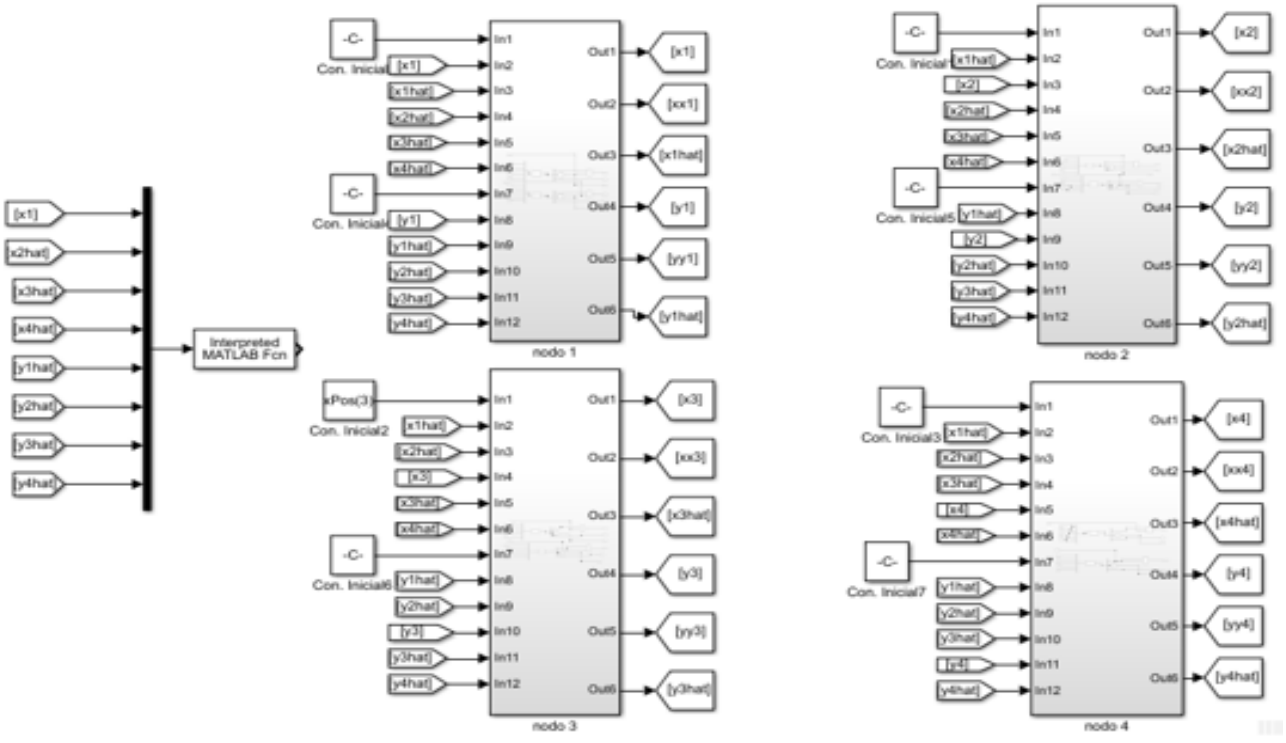

Figura 3.17: Subsistemas de control colaborativo para cada nodo.

Para esta simulación se configuran los parámetros de la red como se muestra en la figura 3.18. En esta, se utiliza como base el estándar IEEE802.11b para realizar el proceso de comunicación entre la topología presentada de 4 nodos, con una tasa de transmisión de 1Mbits/s, tamaño de frame mínimo de 24 bits, potencia de transmisión pow = -8dB (.16mW), un umbral de señal recibida de thres = -48dB (15.8nW), un exponente de pérdida de 3.5, Ack timeout = 40us.

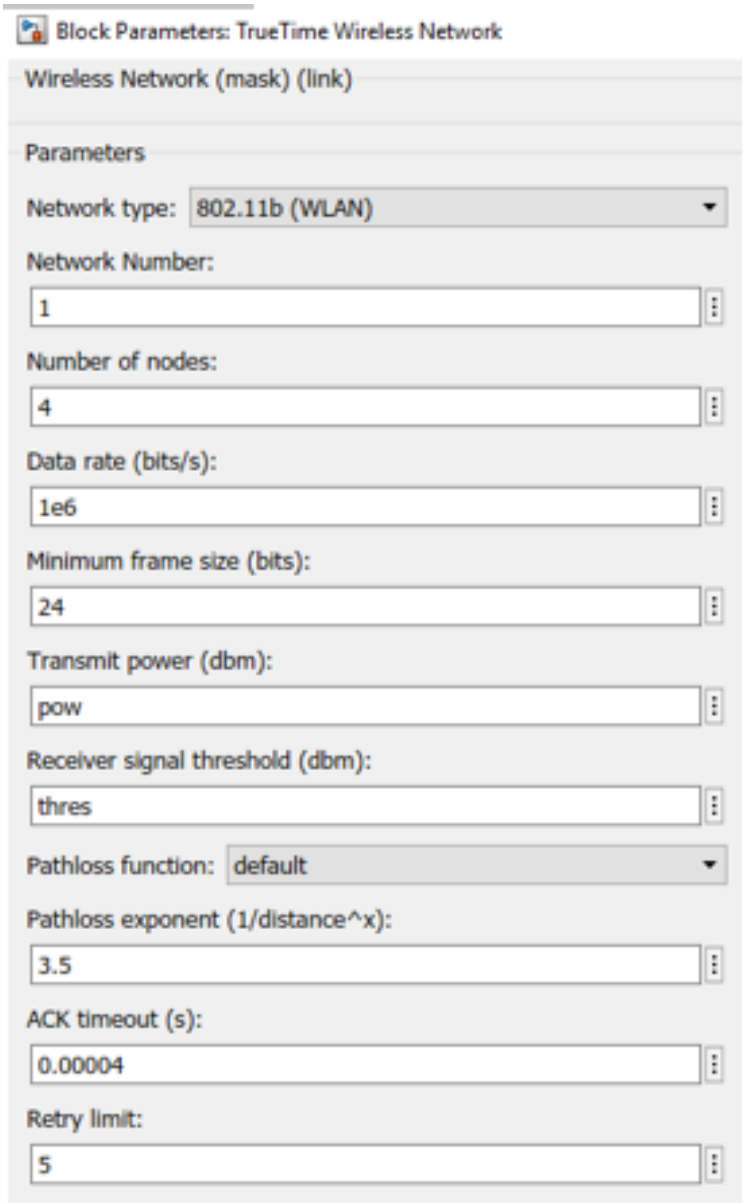

Figura 3.18: Parámetros para llevar a cabo la comunicación y el enrutamiento.

Al llevar a cabo la simulación, los nodos se posicionan aleatoriamente en un área de 10x10m como se muestra en la figura 3.19 para después llegar al consenso por medio de los bloques de control.

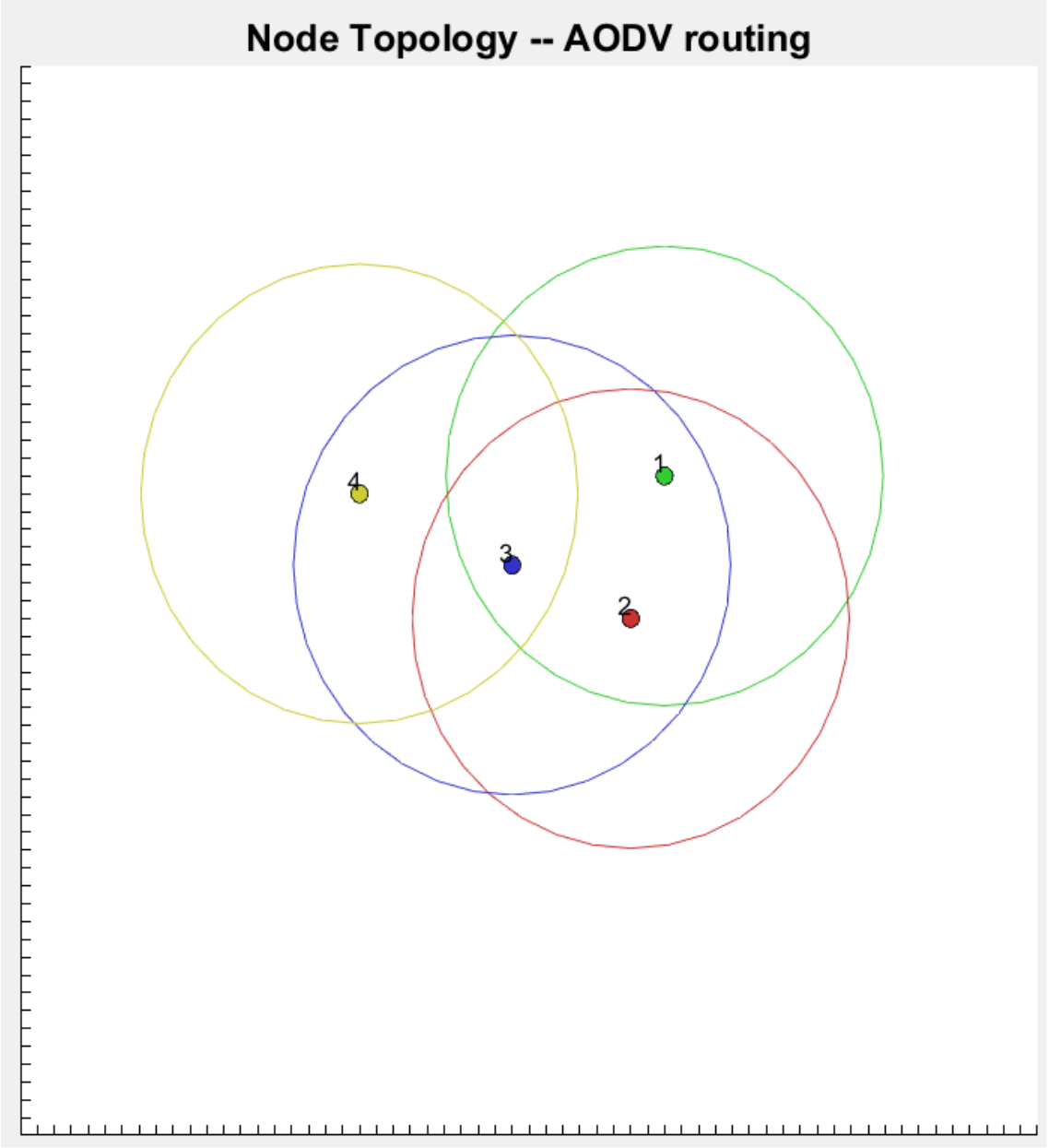

Figura 3.19: Generación pseudo-aleatoria de 4 nodos en un área de 10x10m por medio de una distribución uniforme.

El algoritmo de los datos se presenta en el diagrama de flujo de la figura 3.20. En este se muestra que en principio se establecen los parámetros de la red, para después generar de manera aleatoria con una distribución uniforme las posiciones iniciales en $x_{i,1}$ y $x_{i,2}$ de los nodos. Posteriormente, mediante la tabla de enrutamiento, el nodo 1 es el que toma las posiciones de los nodos y las envía al resto para que cada uno pueda calcular su señal de control cuando se genere el disparo de un evento. Para comenzar el proceso de transmisión de las posiciones, primero el nodo 1 manda un mensaje RREQ (requerimiento de ruta) a todos sus vecinos, donde se solicita el ID del nodo destino (como se desea comunicar con cada uno de los otros nodos, hace este proceso para cada uno de ellos). Cuando el

RREQ llega al nodo destino, este envía de vuelta al nodo original un RREP (respuesta de ruta) lo que permite entonces establecer la conexión hasta que sea necesario. Después, se procederá a enviar el mensaje. En caso de no enviar mensaje y querer mantener la ruta, se envía un hello message (mensaje de saludo) para indicar que se desea conservar el vínculo. Seguido de esto, en caso de que el enlace no esté roto (esto ocurre en caso de que los nodos estén fuera del rango de transmisión) comienza el envío de las posiciones de los nodos al resto de los agentes. Una vez que llegue a todos los nodos la información solicitada, estos calculan la señal de control nueva cuando se genere la condición de evento. Después actualiza la posición guardada en la memoria de cada nodo y nuevamente el nodo 1 toma las posiciones de los demás agentes para comenzar el proceso de nuevo.

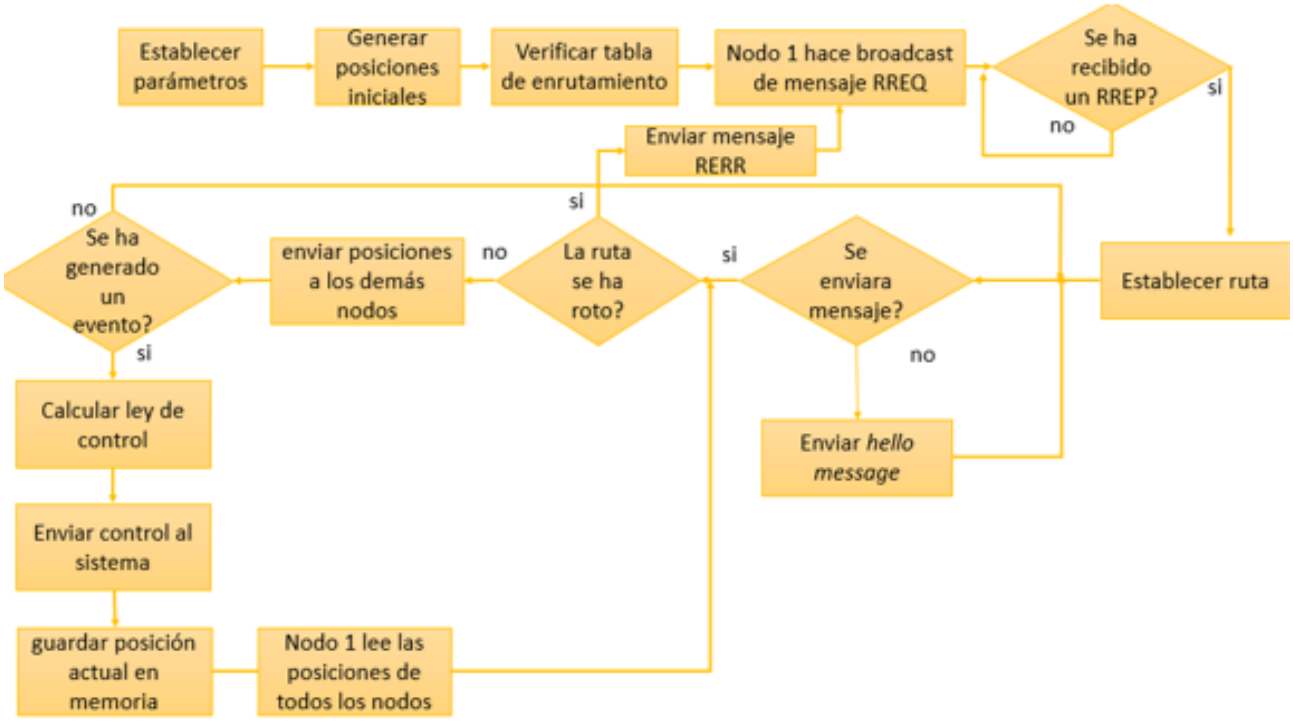

Figura 3.20: Diagrama de flujo del algoritmo de la simulación.

Al llevar a cabo el proceso anterior, se consigue la formación deseada en los nodos como se muestra en las figuras 3.21, 3.22 y 3.23. en donde las primeras dos muestran el valor alcanzado por $x_{i,1}$ y $x_{i,2}$ respectivamente mientras que la última muestra la representación gráfica del nodo.

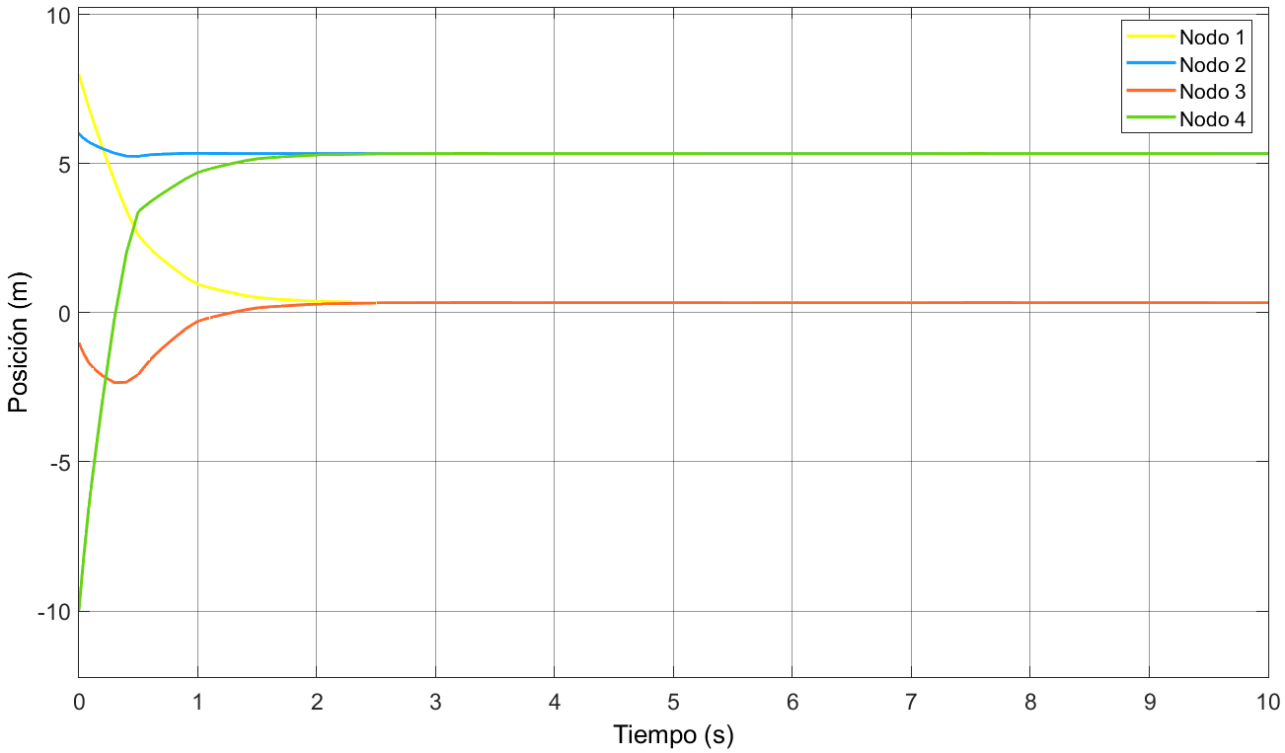

Figura 3.21: Consenso alcanzado en $x_{i,1}$.

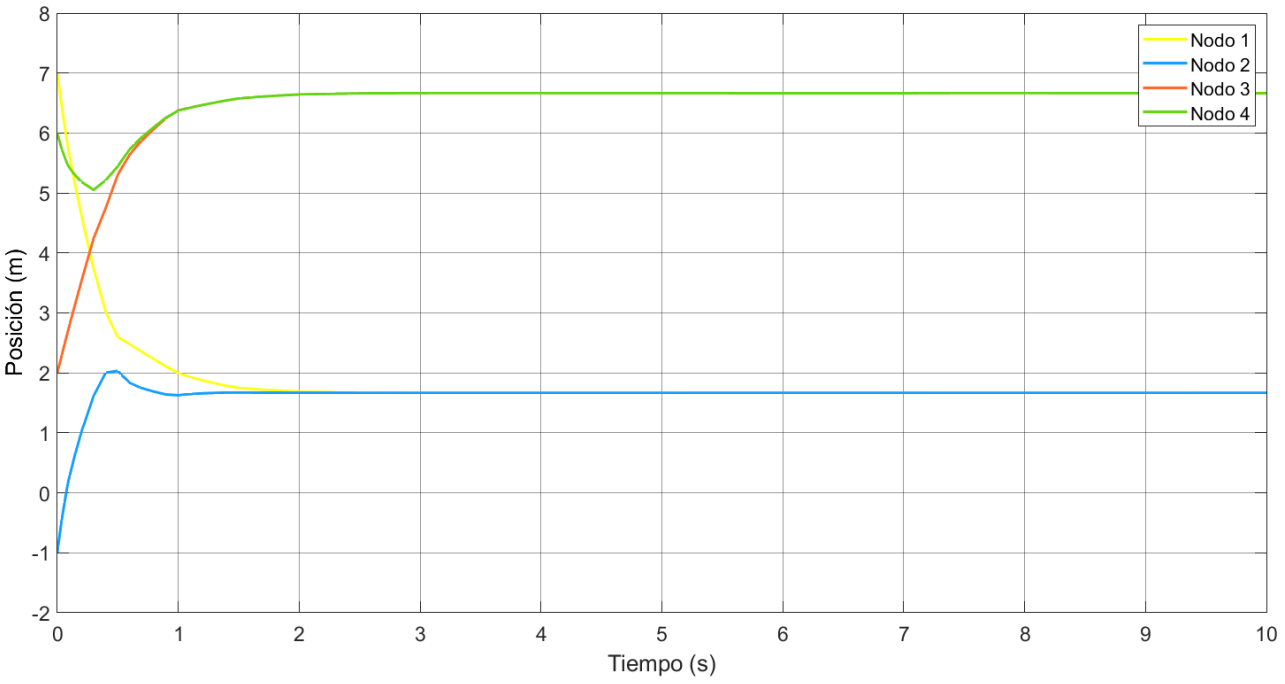

Figura 3.22: Consenso alcanzado en $x_{i,2}$.

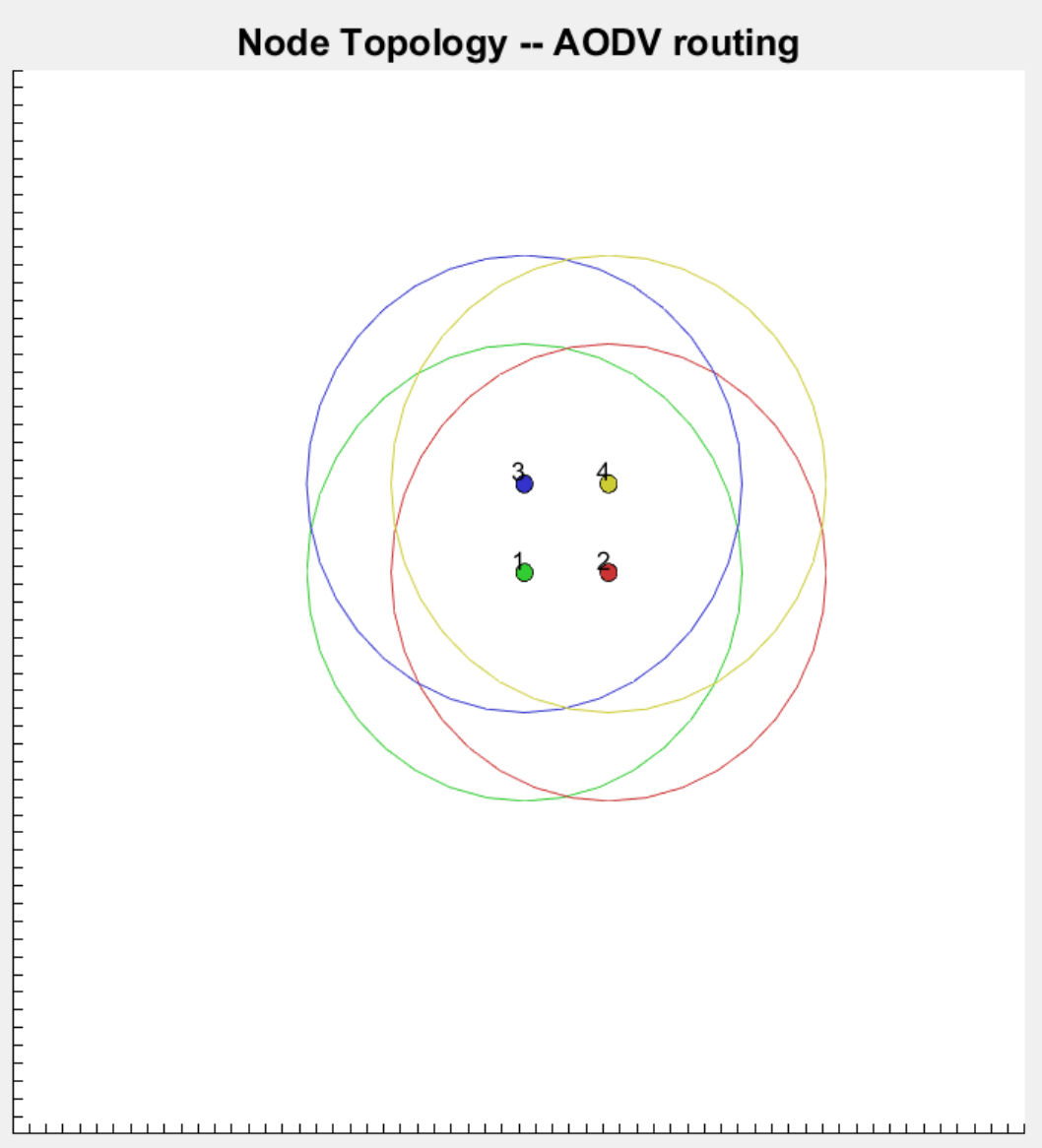

Figura 3.23: Consenso alcanzado por los 4 nodos.

Capítulo 4

Implementación de los nodos VANET

Dentro de este capítulo se describen los elementos que permiten realizar la implementación del nodo VANET que consiste en un vehículo omnidireccional (3,0), utilizando la arena de pruebas descrita en el capítulo 1, y siguiendo las bases y fundamentos teóricos que fueron previamente planteados en el capítulo 2 así como partiendo de las simulaciones realizadas con anterioridad en el capítulo 3 por medio de las herramientas de Matlab/Simulink, el sistema operativo ROS y el simulador Gazebo.

4.1. Esquema de implementación

Para llevar las simulaciones obtenidas en el Capítulo 3 a la práctica se dispone de una herramienta fundamental denominada ROS la cual permite al usuario abstraerse de algunos aspectos muy específicos en cuestión de las comunicaciones entre robots que se utiliza en conjunto con Matlab Simulink para crear los modelos correspondientes a los nodos diseñados junto al control colaborativo propuesto en el Capítulo 2 para, de esta manera, poder crear una estrategia de enrutamiento más especializada. Por tanto, haciendo uso de ROS, existen dos maneras para realizar una implementación de los nodos VANET tal como se muestra en el diagrama de la figura 4.1, en donde se observa que se dispone de las herramientas de ROS y de Matlab/Simulink para de ahí efectuar las implementaciones virtuales haciendo uso del simulador Gazebo y de un modelo URDF correspondiente al vehículo omnidireccional de 3 ruedas, mientras que la implementación física se realiza por medio de los vehículos omnidireccionales físicos controlados cada uno por su propia Raspberry Pi 3B, conectados en una red Wifi y obteniendo sus propias posiciones por medio del sistema de cámaras Optitrack.

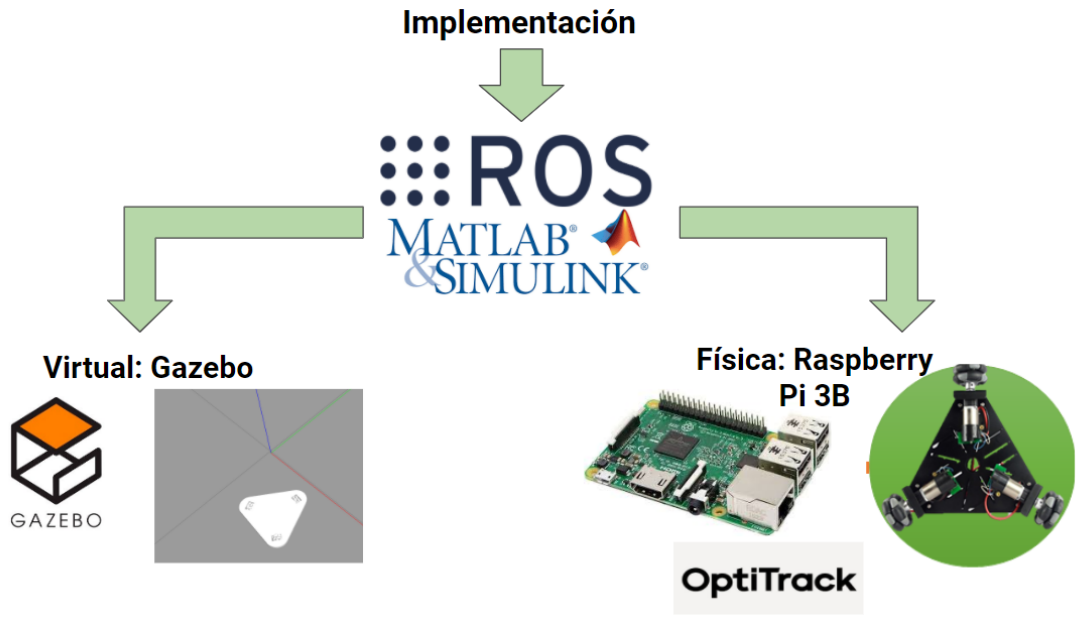

Figura 4.1: Esquema de implementación para los nodos VANET.

4.2. ROS como herramienta de implementación

Consiste en un meta sistema operativo robótico (ROS por sus siglas en inglés) que permite al usuario disponer de un conjunto de herramientas que facilitan el diseño e implementación de sistemas robóticos, mediante un conjunto de paqueterías y bibliotecas, que permiten crear aplicaciones de manera sencilla y estandarizada, para ser utilizadas en ambientes donde se utilicen robots.

De acuerdo con [28] algunas de las características principales que ofrece ROS son:

- Mecanismo de comunicación entre programas: Consiste en un estándar que maneja computación distribuida, lo que permite conectar diferentes programas en un sistema común ya sea de uno o varios equipos.
- Reusabilidad de código: Se conforma por un conjunto de algoritmos que se utilizan comúnmente en sistemas robóticos convertidos en paqueterías estándar y permite manejar la comunicación y gestión de datos de manera directa.
- Testeado rápido: Permite simular dispositivos que serán utilizados en sistemas auxiliándose de las herramientas preestablecidas de comunicación, y es posible aislar la funcionalidad del sistema entre cada una de las partes que lo conforman, para realizar pruebas bajo las mismas condiciones.

La comunicación realizada por ROS se basa en el protocolo XML-RPC, el cual se trata de un protocolo servidor-cliente, que utiliza XML como lenguaje para su codificación y

HTTP (Puerto 80) como medio de transmisión, normalmente abierto. Las características principales de este protocolo son:

- Permite diferentes lenguajes de programación (PHP, java, python, perl, c, c++,etc.).
- Es de fuente abierta.
- Sencillo de utilizar.

Los diferentes elementos que conforman una red de ROS y que permiten llevar a cabo la comunicación de estos abstrayéndose del protocolo XML-RPC [29] son:

- Nodo: Es una unidad de procesamiento que realiza alguna acción de cómputo dentro del sistema.
- Master: Provee un nombre de registro a cada nodo que conforma la red y permite que los nodos se comuniquen entre ellos.
- Mensajes: Estructura de datos simple y estandarizada.
- Tópico: Es el medio que utilizan los nodos para compartir información entre ellos. Los nodos son capaces de publicar información y suscribirse a la información de otros nodos. Este proceso es unidireccional, ya que no existe control de los nodos que se suscriben a un tópico.
- Servicio: Es una comunicación de una sola vez donde un nodo solicita un servicio y otro provee dicho servicio.
- Paquete: El software en ROS se organiza en paquetes, los cuales pueden contener nodos, librerías, conjunto de datos, etc.
- Bags: Formatos para guardar la información de los mensajes de ROS para poder analizarla, reproducirla, etc.

En la figura 4.2 se presenta la manera en la que los nodos ejecutan su proceso de comunicación. Primero, cada nodo se registra por medio del Master, el cual esta al tanto de los elementos que conforman la red, de quien publica mensajes y quien se suscribe a ellos. Para intercambiar información de manera continua, el nodo 1 publica el mensaje a enviar en un tópico para que el nodo 2 pueda suscribirse a este para poder recibir el mensaje. Por otra parte, el nodo 1 puede solicitar un servicio al nodo 2 y este responde con la información requerida.

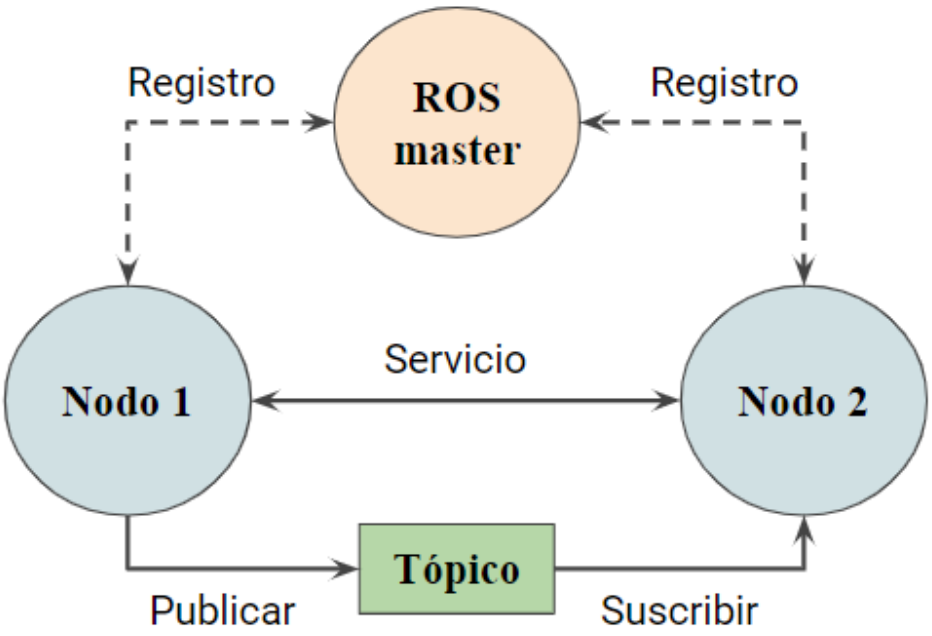

Figura 4.2: Comunicación en ROS.

ROS cuenta con una serie de comandos que permiten utilizar funcionalidades o librerías en el sistema que se esté implementando, algunos de estos comandos son:

- roscd: Cambia a un directorio de paquete o pila (ej. roscd stage).
- roscore: Ejecuta todo lo necesario para dar soporte de ejecución al sistema completo de ROS. Siempre tiene que estar ejecutándose para permitir que se comuniquen los nodos. Permite ejecutarse en un determinado puerto (ej. roscore o roscore -p 1234).
- roscreate-pkg: Crea e inicializa un paquete. Se tiene que ejecutar desde uno de los directorios válidos que contengan paquetes. El formato de ejecución es: roscreate-pkg paquete [depen1 ...] donde depen1 es una dependencia. Por ejemplo, si el paquete que se esta creando va a usar los mensajes estándar y va a usar código por lo que se deben indicar las dependencias std-msgs y roscpp.
- rosnode info nodo: Muestra información sobre el nodo.
- rosnode kill nodo: Termina la ejecución el nodo indicado.
- rosnode list: Muestra los nodos que se están ejecutando.
- rosnode machine máquina: Muestra los nodos que se están ejecutando en la máquina.
- rosnode ping nodo: Comprueba la conectividad del nodo.

- rosrun: Permite ejecutar cualquier aplicación de un paquete sin necesidad de cambiar a su directorio. Es posible pasarle parámetros con -my-param:value (ej. rosrun stage stageros) siendo "stage" el paquete y "stageros" la aplicación que se ejecuta.
- rostopic: Permite obtener información sobre un tópico.
- rostopic echo: Imprime datos del tópico por la salida estándar.
- rostopic info: Imprime información de un tópico.
- rostopic list: Imprime información sobre los tópicos activos.
- rostopic pub: Publica datos a un tópico activo.
- roslaunch: Corre un conjunto de nodos o comandos.

Rqt-graph es una aplicación que permite visualizar la manera en que se están ejecutando los nodos, ver los tópicos que publican y ver los tópicos a los cuales están suscritos.

Una vez instalado ROS, y al hacer uso de las paqueterías existentes que se utilizaron para los crazyflies (drones) se procede a realizar lo siguiente:

1. Asegurarse de que la máquina en la que se levantará ROS esté en la misma red que la máquina que transmite los datos de las cámaras Optitrack e igualmente que la Raspberry Pi y la máquina que se encargue de desplegar los programas en la misma se encuentren en dicha red.
2. Se usa el comando export ROS-MASTER-URI=http://192.168.0.105:11311 en donde la IP introducida es la del equipo que levanta el sistema ROS.
3. Escribir export ROS-IP=192.168.0.114 con la IP del nodo que se utilizará para enviar las posiciones de los agentes creados en Motive.
4. Usar el comando source crazyfile-ws/devel/setup.bash para posicionarse en la carpeta del espacio de trabajo.
5. Lanzar el comando roslaunch vrpn-client-ros sample.launch server:=192.168.0.114 para lanzar el nodo donde se publican las posiciones y orientaciones de los vehículos.

4.3. Gazebo

Gazebo es un simulador de ambientes robóticos para interiores y exteriores que ofrece la posibilidad de simular sistemas complejos de robots con sensores de una manera precisa y eficiente, emulando las variables físicas del mundo real, en una interfaz sencilla de

utilizar como se muestra en la figura 4.3, permitiendo probar algoritmos de control, comunicaciones y diseños propios de robots en escenarios realistas [30]. Tiene una arquitectura cliente/servidor que se basa en las herramientas de ROS de subscripción y publicación de mensajes en los tópicos para generar los procesos de comunicación entre las diferentes aplicaciones robóticas. Algunas de las características principales de Gazebo son:

- Simulación dinámica: Permite acceder a motores de físicas de alto rendimiento.
- Gráficos 3D avanzados: Considera aspectos de iluminación, sombras y texturas de alta calidad.
- Sensores y ruido: Genera sensores de datos con la posibilidad de agregar y ajustar diferentes patrones de ruido.
- Plugins: Desarrolla plugins personalizados para robots, sensores y controles.
- Modelos de robots: Ofrece una variedad predeterminada de diferentes modelos de robots existentes.
- Herramienta de línea de comandos: Facilita la introspección de simulación y control.

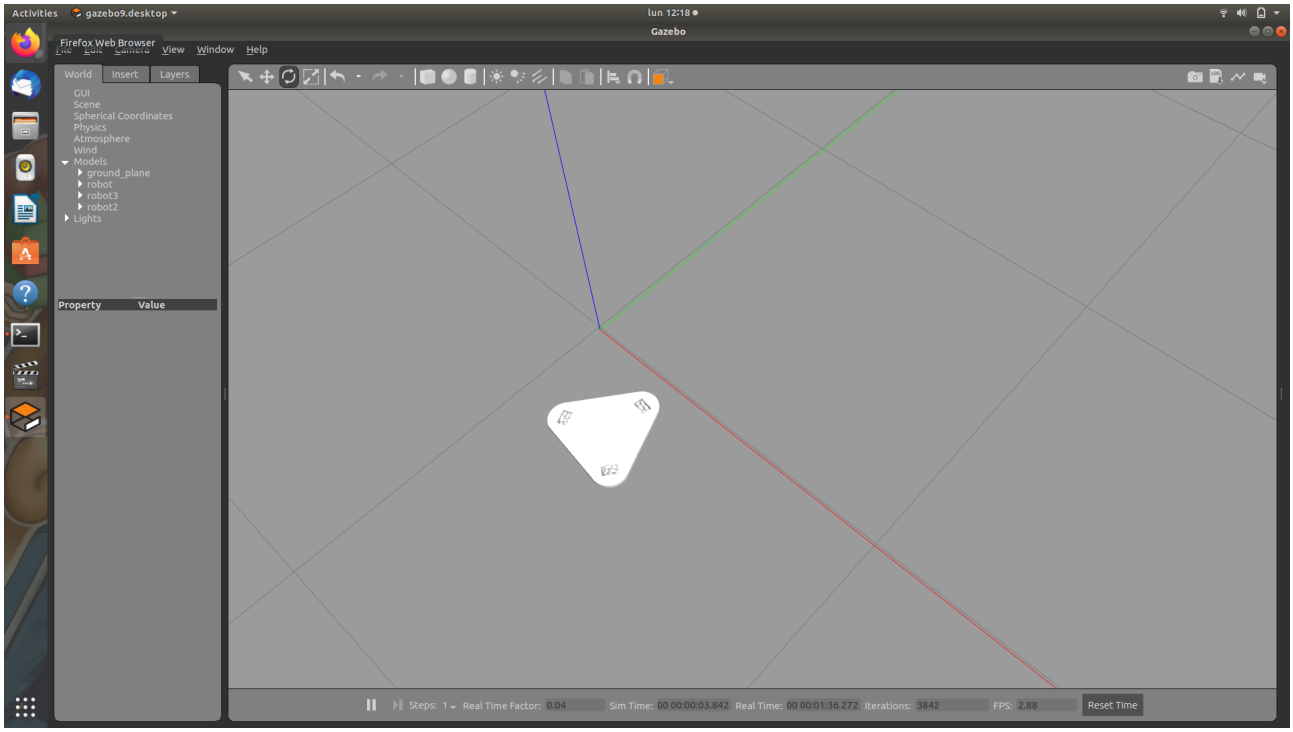

Figura 4.3: Interfaz gráfica de usuario de Gazebo.

4.4. Nodo VANET en Matlab/Simulink

Matlab 2019b introduce ROS Toolbox, que proporciona una interfaz que conecta Matlab-Simulink con el meta sistema operativo, la cual permite construir nodos desde Simulink por medio de una generación de código en C++ e incluso proporciona herramientas de co-simulación con Gazebo.

Algunos de los bloques con los que cuenta esta paquetería son los mostrados en la figura 4.4 que permiten publicar tópicos y suscribirse a ellos, siempre y cuando se haya inicializado el sistema ROS.

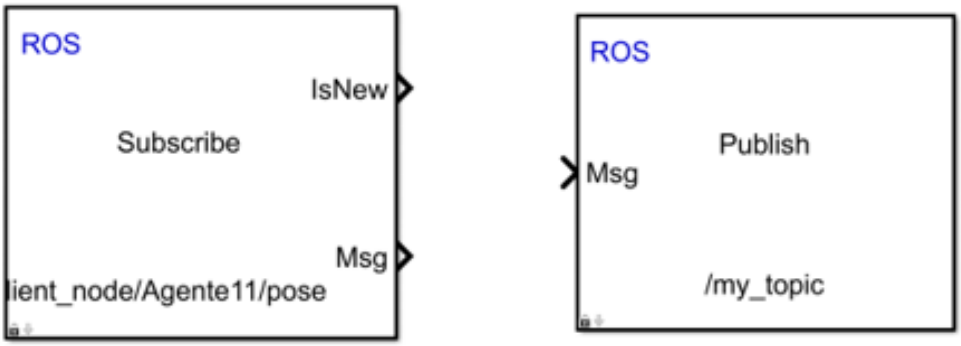

Figura 4.4: Bloques de suscripción y publicación de tópicos en Simulink con ROS.

En la figura 4.5 se presenta el modelo creado de cada uno de los nodos que conforman la red VANET para su implementación en el simulador de Gazebo. En primera instancia, el bloque denominado "Activación del nodo y generación de tabla de enrutamiento" cuenta con un tópico de activación, el cual da inicio a la búsqueda de vecinos del nodo para llenar la tabla de enrutamiento (este bloque es externo y acciona el resto de los bloques pero pasa directamente la información de los vecinos con los cuales se tiene que comunicar el nodo). Posteriormente, el bloque "Posiciones del vehículo en el entorno de Gazebo" obtiene sus propias coordenadas directamente del ambiente de Gazebo. El bloque "Enrutamiento y publicación de posiciones" recibe los valores de la tabla de enrutamiento para suscribirse a los vecinos correspondientes y tomar sus posiciones o para realizar un enrutamiento. En seguida, el bloque "Consenso disparado por eventos" toma los valores de las posiciones para realizar el cálculo del control colaborativo disparado por eventos, con su modelo correspondiente para ser enviado al bloque "Publicación de la posición del agente al entorno de Gazebo" en donde se publicará el valor resultante de la estrategia del control del nodo en un comando de velocidad del vehículo omnidireccional. Adicionalmente, el agente 1 tendrá un bloque que genera la trayectoria a seguir por medio de un consenso con un agente virtual.

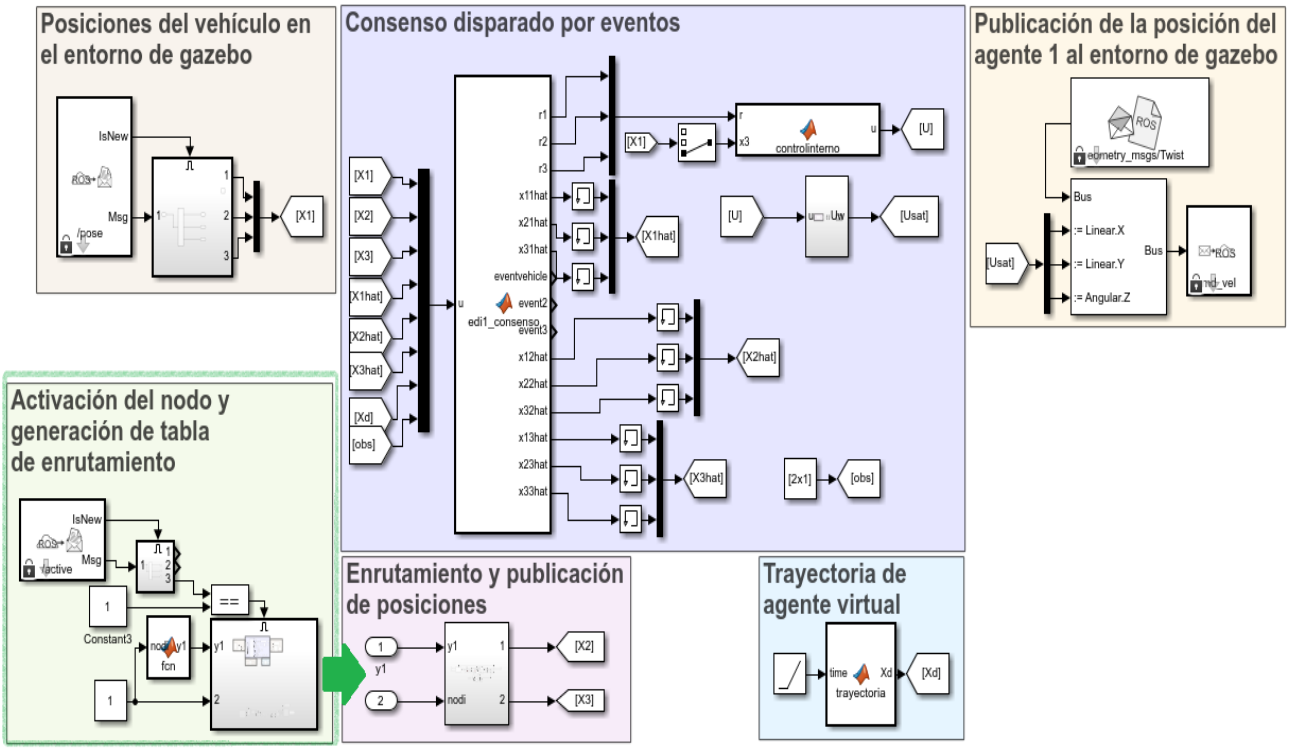

Figura 4.5: Modelo del nodo VANET en Simulink para Gazebo.

4.5. Integración del nodo en ROS y Gazebo

En la figura 4.6 se aprecian los elementos principales que se necesitan para poder efectuar la implementación de los nodos VANET en ROS y Gazebo, donde se crea el nodo desde Simulink inicializando ROS en la ventana de comandos de Matlab a través de la línea de código "rosinit" y posteriormente en la ventana del modelo de Simulink se presiona la combinación de teclas "ctrl + b" para obtener un archivo "nodo.tgz" y otro llamado "model.sh" para posteriormente pasarlos a una máquina con sistema operativo Ubuntu donde se tenga instalado ROS. Aquí, en una terminal posicionada en el directorio donde se encuentra el archivo generado por Simulink, se ejecuta el comando "./build_ros_model.sh nodo.tgz ~/catkin_ws/". Posteriormente es necesario contar con un archivo URDF (Unified Robotic Description Format) para el modelo del vehículo omnidireccional (3,0) como se muestra en la figura 4.7 el cual, de manera simplificada, se compone de una base triangular con 3 ruedas acopladas a esta a través de sus respectivos joints. Por otra parte, Gazebo cuenta con distintos Plugins que permiten el funcionamiento de este en conjunto con ROS y, en este caso, se utiliza el Plugin "/gazebo/model_states" para publicar las posiciones de los nodos que se estén utilizando en el ambiente de Gazebo. Posteriormente se crean scripts adicionales en código Python o C++ con el propósito de ayudar en la ejecución de la implementación. Finalmente, los elementos anterior se integran en un archivo launch donde se asocian los nodos creados con los modelos URDF de los vehículos omnidireccio-

nales (3,0) para ser ejecutados en Gazebo apoyándose también con los scripts realizados. Al contar entonces con un launch file, primero se utiliza el comando "catkin_make" para cargar los nodos creados en Simulink y ,posteriormente, es necesario posicionarse en la carpeta donde se encuentra el launch file por medio de "cd catkin_ws/src/launch" donde "launch" es la carpeta en la que se encuentra el archivo para finalmente ejecutar con el comando "roslaunch archivo.launch".

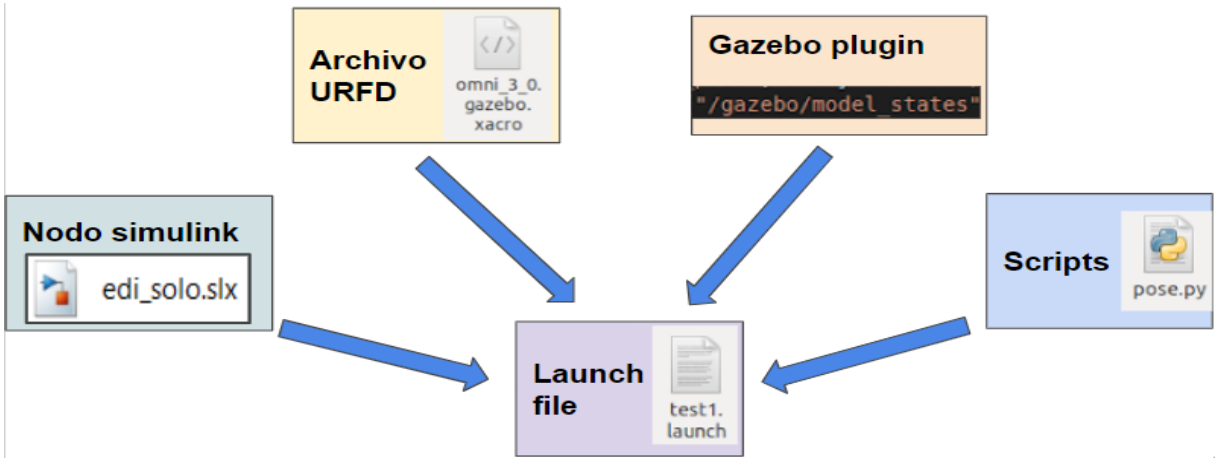

Figura 4.6: Elementos requeridos para la implementación de ROS y Gazebo.

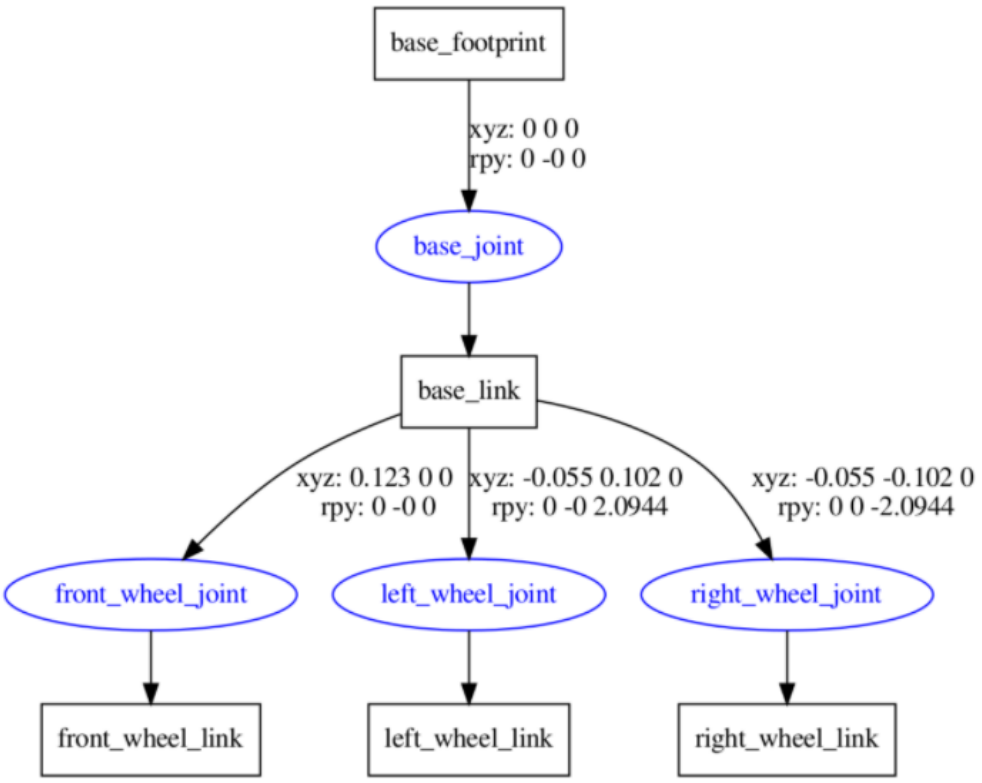

Figura 4.7: Esquema URDF del vehículo omnidireccional.

4.6. Estructura del nodo VANET en implementación física

El nodo VANET se presenta en la figura 4.8 el cual se trata de un robot móvil omnidireccional de tres grados de libertad que está conformado por un chasis con aleación de aluminio, tres ruedas omnidireccionales, tres motores DC 12V a 6W con velocidades de 120RPM hasta 9600RPM (dependiendo de la carga) con consumo de corriente desde 50mA hasta 500mA, una tarjeta Raspberry Pi 3B que sirve para establecer las comunicaciones con los demás nodos y para implementar las leyes de control en el vehículo (el modelo de la tarjeta a utilizar depende de la versión de Matlab con la que se cuente, para el 2018b solo puede ser con RaspberryPi 3B y modelos previos), dos tarjetas H L298N donde cada una consta de dos puentes H que son utilizados para accionar los motores por medio de la tarjeta principal (en total se utilizan tres, uno para cada motor), alimentaciones de 5V (para la alimentación de la tarjeta RaspberryPi 3B) y 12V (para accionar los motores por medio del tren de pulsos generado por los puentes H y el PWM de la RaspberryPi) y, finalmente, un conjunto de marcadores infrarrojos que sirven para que las cámaras Optitrack puedan establecer la posición del nodo.

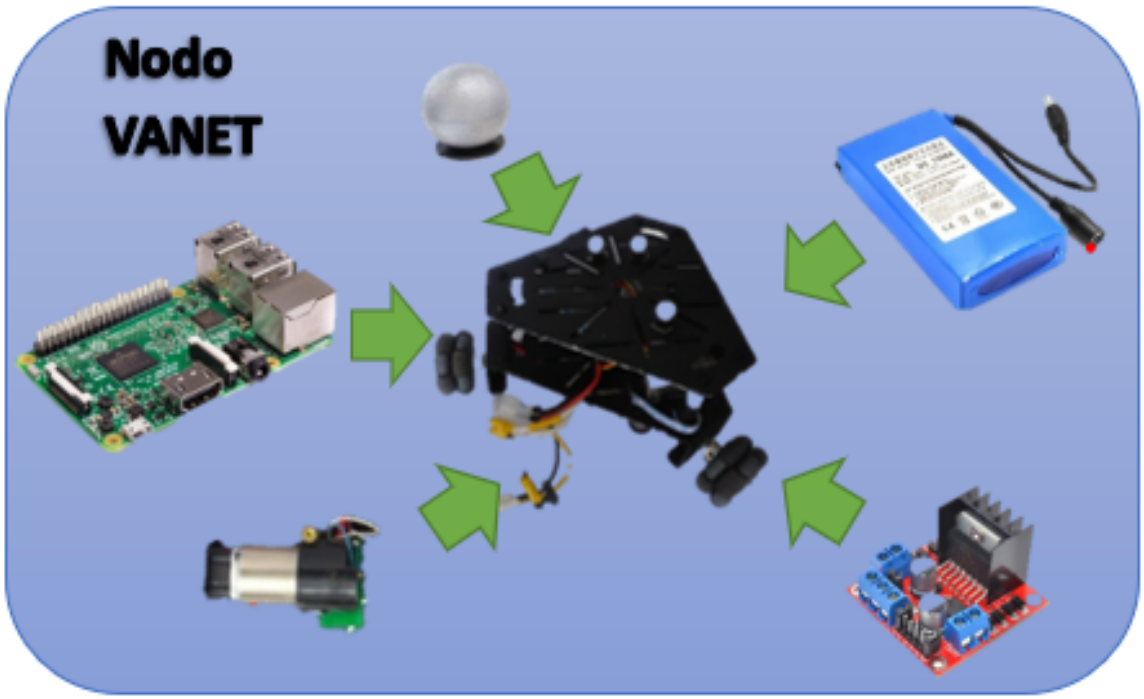

Figura 4.8: Composición del nodo VANET.

4.7. Topología de las comunicaciones entre los nodos y Optitrack

En la figura 4.9 se presenta un diagrama general del esquema de configuración entre los nodos de comunicación y el sistema de visión, donde dicho sistema se compone por el conjunto de 4 cámaras infrarrojas Optitrack las cuales obtienen las posiciones de los vehículos para después ser capturadas en el equipo de cómputo con el software denominado Motive para, consecuentemente, ser publicadas en un tópico de ROS al cual los nodos pueden tener acceso para conocer sus posiciones, mientras que los mismos publican sus posiciones propias (obtenidas por el sistema de visión) en otros tópicos a su vez que dan a conocer dichas posiciones a sus respectivos vecinos para que de esta manera cada nodo sea capaz de generar su ley de control en base dichas posiciones obtenidas.

Figura 4.9: Configuración de la implementación de los nodos VANET.

4.7.1. Sistema de adquisición de las posiciones con Optitrack

Optitrack es un conjunto de sistemas de alta precisión para el rastreo de objetos en tiempo real, optimizado para sistemas robóticos de vehículos terrestres y aéreos, permitiendo obtener su posición y orientación. Las cámaras utilizadas son las Flex 13, las cuales tienen una resolución de 1280x1024 y 120 FPS. El software utilizado como interfaz para el manejo de las cámaras es Motive, en el cual se crea el área de pruebas con base en el conjunto de cámaras, se calibra y se crean los agentes que se utilizan. En la figura 4.10 se observa el ambiente de trabajo de Motive, en donde el Agente11 y el Agente12 son dos VANET que se implementaron para pruebas posteriores.

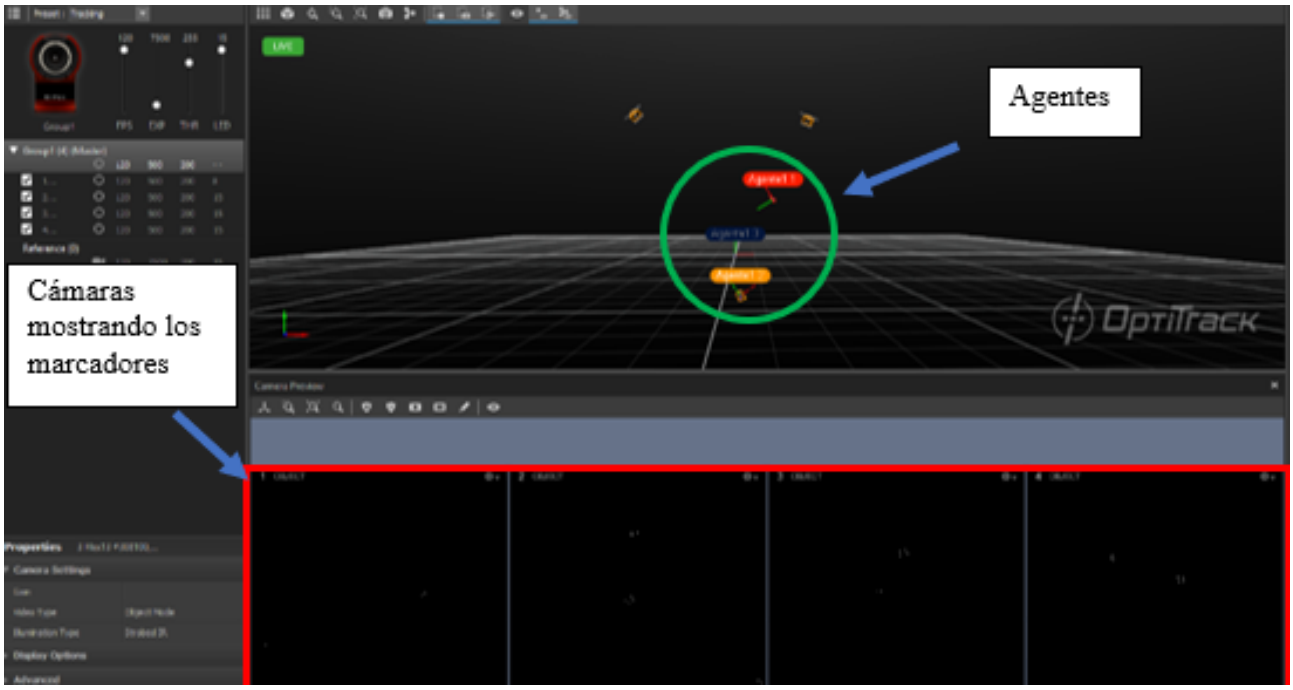

Figura 4.10: Interfaz de Motive.

Estos agentes creados se denominan cuerpos rígidos, los cuales indican la posición y la orientación de los nodos VANET en tiempo real, y al establecer en los parámetros de comunicación introduciendo la IP del equipo donde se esté usando Motive en la ventana de Data Streaming, es posible transmitir dichos datos al ordenador mientras estén conectados en la misma red.

4.7.2. Módulos de Raspberry Pi

Con el fin de implementar los modelos introducidos en el capítulo 3 y generar las comunicaciones por medio de ROS en la Raspberry pi 3B, es necesario contar con tres paqueterías.

- Matlab Support Package for Raspberry Pi hardware.

- Simulink Support Package for Raspberry Pi hardware.
- Robotics system toolbox.

Las primeras dos sirven para contar con el uso de bloques que permiten acceder a funcionalidades de la Raspberry Pi, tal es el caso del bloque generador de PWM mostrado en la figura 4.11, en el cual se selecciona el modelo de la Raspberry que se utiliza y el GPIO que se desea usar para ello.

Figura 4.11: Bloque generador de PWM para Raspberry Pi.

A su vez, estas paqueterías permiten quemar la imagen necesaria para que la Raspberry pueda operar, esto mediante la selección del modelo correspondiente (para Matlab 2018b, el modelo más reciente con el que se puede trabajar es la Raspberry 3B). Una vez que se configura esta imagen en una micro SD y se introduce en la tarjeta, se debe conectar a la misma red que la máquina que la está configurando para entonces, a través de la IP, poder desplegar el programa realizado en Matalab/Simulink a la Raspberry.

Entonces para poder desplegar el script a la tarjeta, es necesario realizar los siguientes pasos:

1. Introducir el siguiente código dentro de la ventana de comandos de Matlab "setenv (ROS_MASTER_URI,http://192,168,0,105:11311)" donde la IP es la correspondiente a la máquina que levanta el sistema ROS.
2. Escribir el comando "setenv(ROS_IP,192,168,0,187)" donde la IP introducida es la de la máquina en la que está trabajando Matlab/Simulink.
3. Utilizar "rosinit(192.168.0.105)" donde la IP corresponde a la máquina que tiene ROS, con el fin de inicializar los comandos de ROS en Matlab-Simulink.

Al realizar todo lo anterior, es posible entonces implementar los modelos introducidos en el capítulo 3, con algunas diferencias mínimas que necesarias para su implementación.

Capítulo 5

Resultados

A lo largo de este capítulo se presentan los resultados relacionados con la implementación que se planeó en el capítulo 1 con base en los fundamentos teóricos del capítulo 2, siguiendo las simulaciones y desarrollo del modelo del capítulo 3 y del capítulo 4. Primeramente, se efectúa el desarrollo del sistema en Gazebo para un sistema multi-agente conformado por 3 vehículos omnidireccionales, mostrando el funcionamiento de la estrategia de enrutamiento de los agentes y la estrategia de control distribuido por eventos a través de los resultados obtenidos. Posteriormente, se presentan los resultados de la implementación física de 2 vehículos omnidireccionales utilizando el sistema de cámaras Optitrack y sus respectivos resultados.

5.1. Resultados de la implementación virtual en Gazebo

En primera instancia, para la prueba en el entorno de Gazebo, se cuenta con 3 nodos VANET conformados por robots omnidireccionales (3,0). El agente 1 tiene una ganancia de $G_1 = 5$ lo que le permite que realice consenso con el líder virtual para definir el seguimiento de una referencia en $x = 0m$ y $y = 2m$ y, a su vez, hace consenso con el agente 2, mientras que el agente 2 efectúa consenso con el agente 3 como se muestra en la figura 5.1 y todos ellos mantienen una distancia de 1m en el eje x entre ellos. El umbral de activación de eventos es $\delta = 0{,}01$, el intervalo de tiempo por cada iteración en Matlab/Simulink es de 0.05s, con el solver Ode4: Runge-Kutta, el radio de seguridad para la evasión de colisiones entre agentes es de $ra = 0{,}5m$, el radio de cobertura de comunicación entre agentes es de $rco = 2m$.

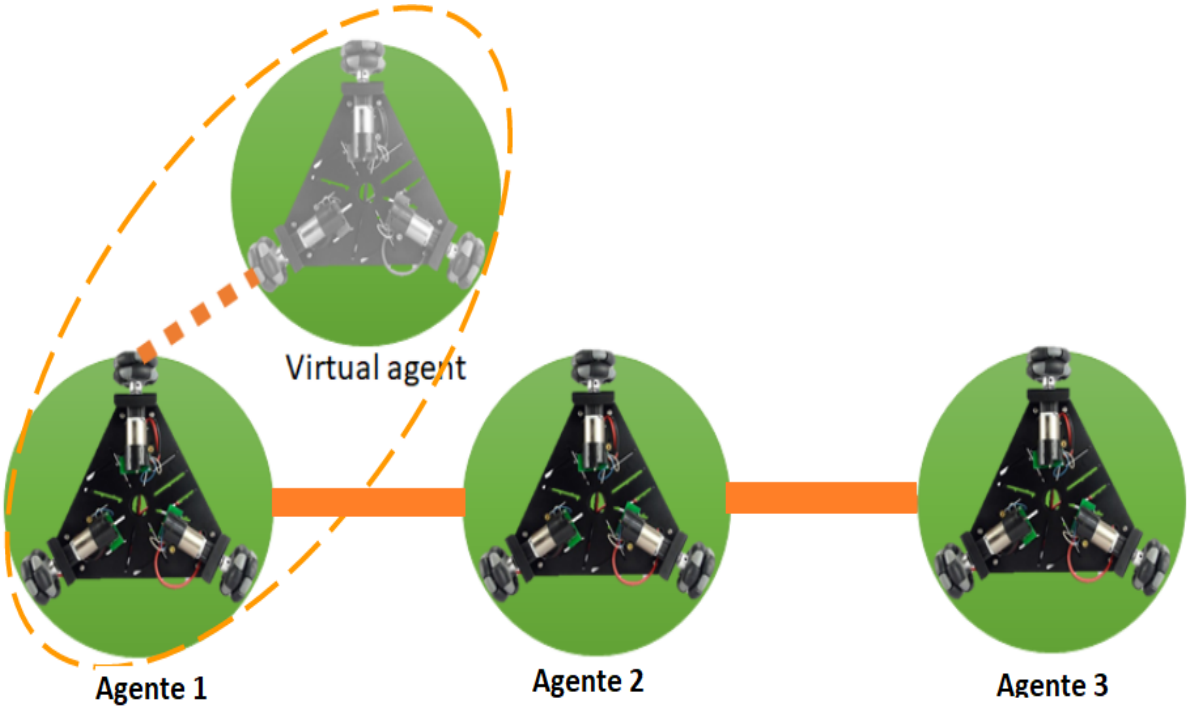

Figura 5.1: Esquema de prueba.

El diagrama de flujo para realizar el enrutamiento y cálculo de la señal de control para cada agente se muestra en la figura 5.2 en donde primeramente, por medio de Gazebo, se obtienen las posiciones iniciales de cada nodo, para realizar un descubrimiento de ruta con base en el radio de cobertura definido previamente para, de esta manera, encontrar la ruta más corta que cada nodo i debe de seguir para que pueda comunicarse con cada vecino j en función al número de saltos para posteriormente accionar el movimiento de los vehículos omnidireccionales por medio de un tópico llamado "/Active" cuando la variable Pose.Orientation.Z sea igual a 1. Posteriormente, se monitorea si existe un evento o no y, en caso de haberlo, se actualizan las memorias de las variables de estado de cada agente y se verifica que exista una ruta válida para requerir las posiciones de los vecinos y, en caso negativo, se vuelve a realizar el descubrimiento de rutas. Si existe una ruta válida, se obtienen las posiciones de los vecinos mediante la subscripción del nodo i al vecino j por medio de su tópico correspondiente en ROS y se actualiza la señal de control. Una vez completados los pasos anteriores, se repite el ciclo dependiendo de la existencia de un evento. Cada agente i siempre publica sus posiciones en su tópico correspondiente, pero cada agente accede a las posiciones de su vecino j únicamente cuando ocurre algún evento.

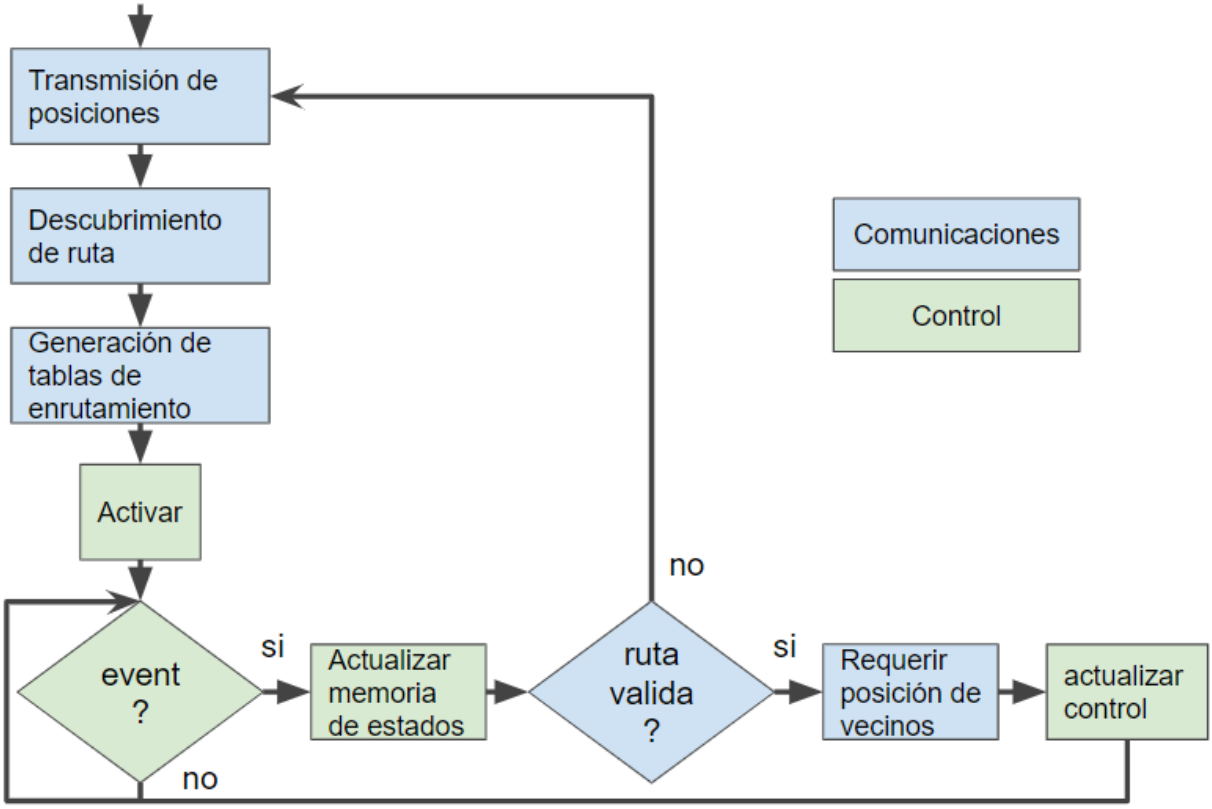

Figura 5.2: Diagrama de flujo de operación de cada agente.

5.1.1. Prueba de consenso de 3 agentes

En la figura 5.3 se presenta la simulación de los 3 agentes realizando consenso, partiendo de las posiciones iniciales de $x_{1,1} = 3m, x_{1,2} = -3m, x_{2,1} = 0m, x_{2,2} = -3m, x_{3,1} = -3m y x_{3,2} = -3m$ hasta que el agente 1 llega a la referencia designada por el agente virtual en $x_{1,1} = 0m$ y $x_{1,2} = 2m$ mientras que el Agente 2 llega a la posición $x_{2,1} = -1m, x_{2,2} = 2m$ y el Agente 3 a $x_{3,1} = -2m, x_{3,2} = 2m$ debido a las inter-distancias designadas de 1m.

La cantidad generada de eventos en cada agente se presenta en la Tabla 5.1 y en la figura 5.4 se muestra el tren de eventos que tuvo lugar en el agente 1 observando que, al irse acercando a la posición final, estos fueron disminuyendo. No obstante, durante toda la trayectoria hubo una gran cantidad de intervalos en los cuales no ocurría ningún evento.

Tabla 5.1: Cantidad de eventos en cada Agente

Agente	***1***	***2***	***3***
Eventos	445	455	325

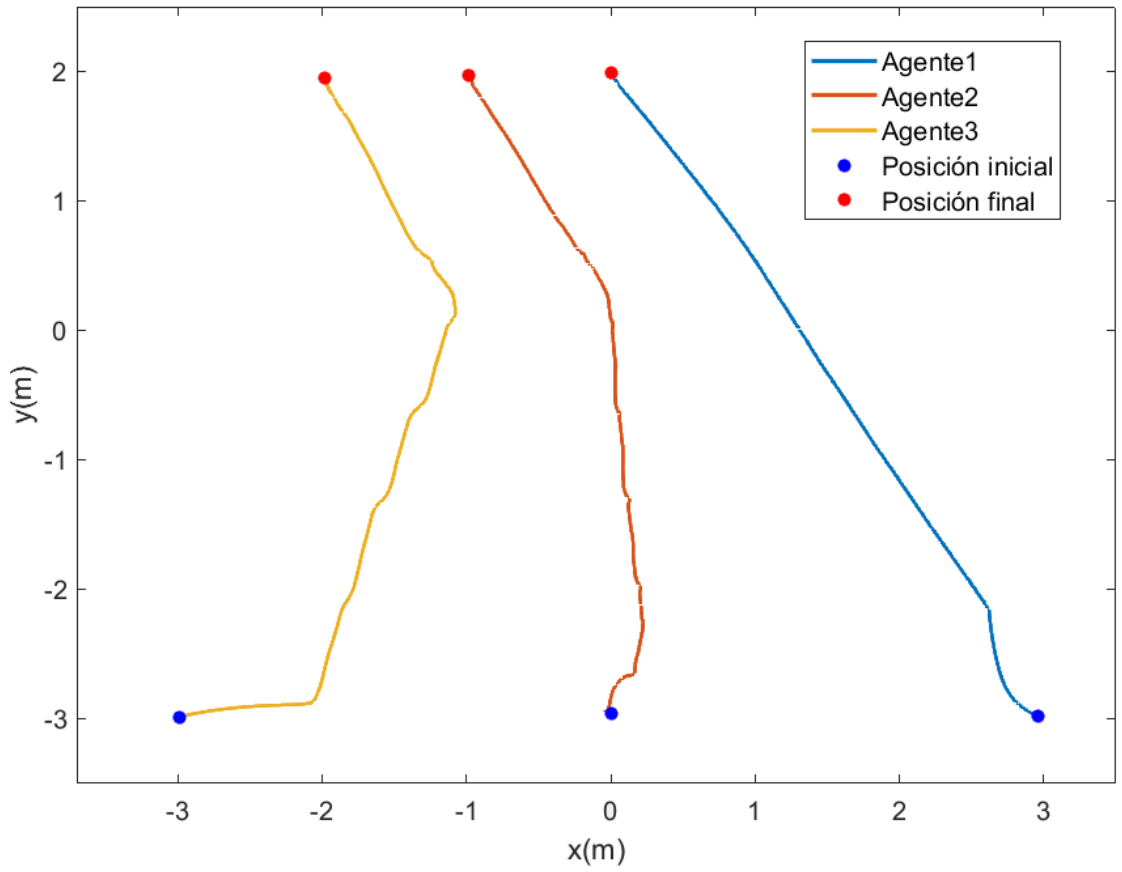

Figura 5.3: Trayectoria de los 3 agentes en Gazebo.

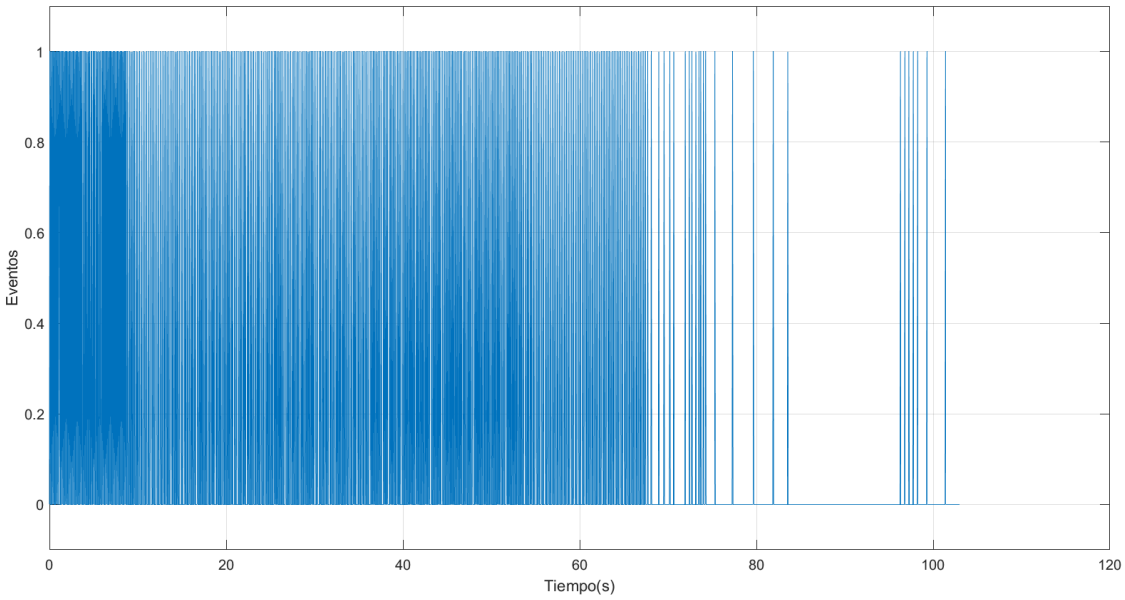

Figura 5.4: Eventos generados en el Agente 1.

Finalmente, en la figura 5.5 se presentan los eventos acumulados durante todo el recorrido del agente 1 comparando esta cantidad de eventos con respecto a una implementación

en tiempo continuo donde no se realiza ninguna estrategia por eventos y en esta, la cantidad de eventos esta representada por el número de iteraciones que corresponde a 2061, por lo que se reduce un 79 % la cantidad de veces que se calcula la señal de control y que se habilitan las comunicaciones del agente i para requerir las posiciones de los agentes j.

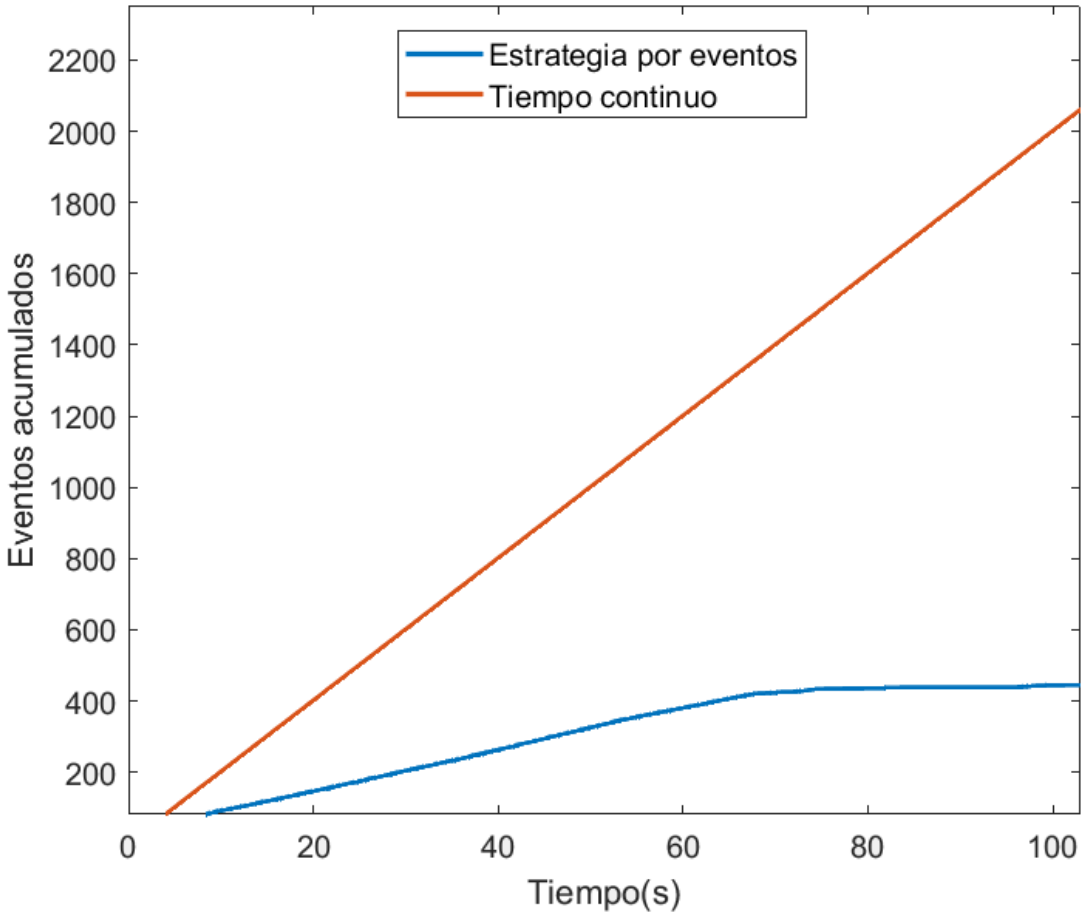

Figura 5.5: Comparación de eventos acumulados contra estrategia continua en el Agente 1.

5.1.2. Prueba de consenso y evasión de colisiones entre agentes

En la figura 5.6 se presenta la simulación de los 3 agentes realizando consenso, partiendo de las posiciones iniciales de $x_{1,1} = -0{,}5m$, $x_{1,2} = -2{,}94m$, $x_{2,1} = 0m$, $x_{2,2} = -2{,}46m$, $x_{3,1} = -1m$ y $x_{3,2} = -2{,}95m$ hasta que el agente 1 llega a la referencia designada por el agente virtual en $x_{1,1} = 0m$ y $x_{1,2} = 2m$ mientras que el agente 2 llega a la posición $x_{2,1} = -1m, x_{2,2} = 2m$ y el agente 3 a $x_{3,1} = -2m, x_{3,2} = 2m$ debido a las inter-distancias designadas de 1m. Cabe recalcar que a pesar de que en la gráfica aparentemente las trayectorias del agente 1 y agente 2 se cruzan en un punto, el tiempo exacto por el cual cada agente cruzó por dicho punto de intersección fue diferente, siendo de 5.59s para el

agente 1 y de 11.18s para el agente 2, por lo que no ocurrió ninguna colisión por parte de ellos e incluso se puede apreciar que el agente 2 esperó a que el agente 1 pasara y por ello en la trayectoria posterior a su posición inicial se observa un movimiento de zig-zag pronunciado.

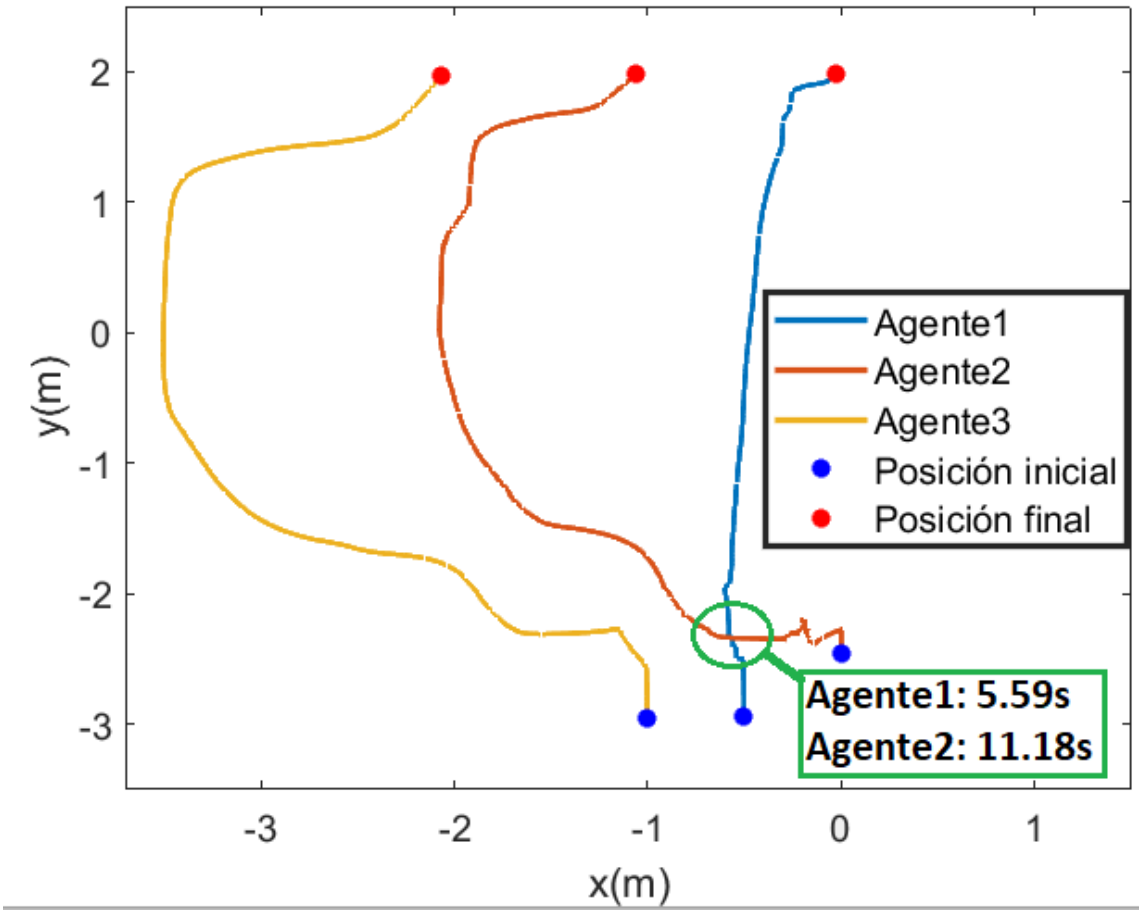

Figura 5.6: Trayectoria de los 3 agentes en Gazebo.

De manera análoga, la cantidad de eventos que fueron generados en cada agente se muestra en la Tabla 5.2, mientras que en la figura 5.4 se contempla el tren de eventos que tuvo lugar en el agente 1, en el cual también se observa una cantidad considerable de intervalos en los cuales no hubo eventos, comparando así en la figura 5.8 nuevamente la cantidad de eventos con respecto a una aproximación del modelo en tiempo continuo para el sistema que evita colisiones entre agentes, donde ocurrieron un total de 2342 iteraciones en dicha estimación, por lo que se decrementó la activación de las comunicaciones y cálculo del control un 82 %, mostrando entonces que aunque se requieran cálculos adicionales para evadir obstáculos, la cantidad de eventos sigue siendo menor, y se ajusta a la naturaleza asíncrona de los sistemas multi-agente.

Tabla 5.2: Cantidad de eventos en cada Agente

Agente	***1***	***2***	***3***
Eventos	438	510	330

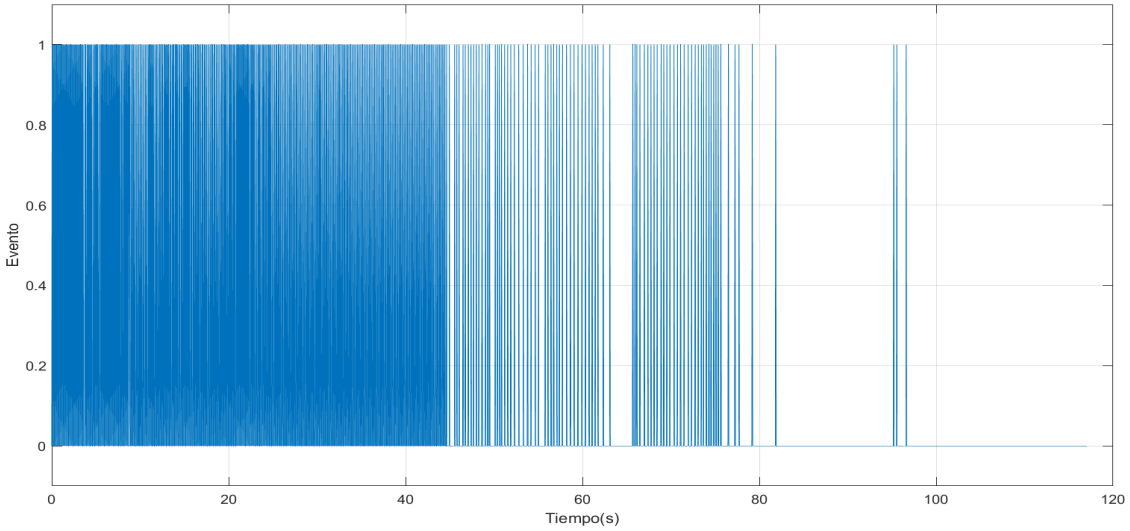

Figura 5.7: Eventos generados en el Agente 1.

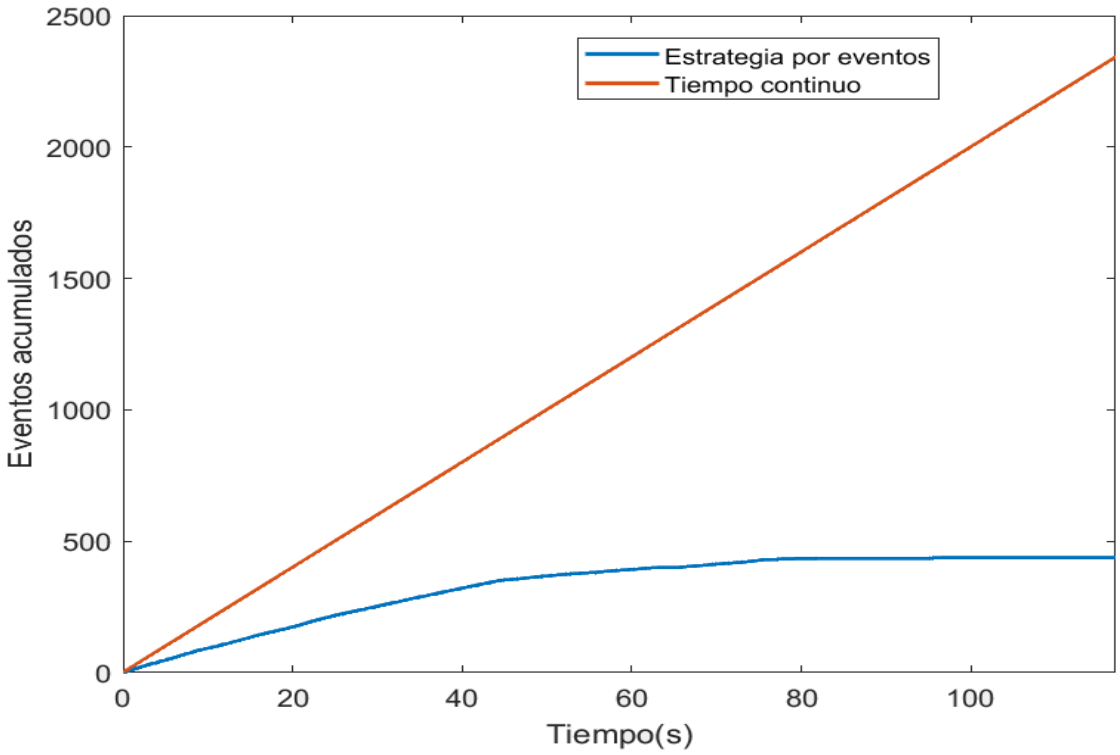

Figura 5.8: Comparación de eventos acumulados contra estrategia continua en el Agente 1.

En el enlace [1] se muestra el video de la simulación realizada con 3 agentes en Gazebo.

5.2. Resultados de la implementación física

En este apartado se utilizan únicamente dos vehículos omnidireccionales. Para poner en marcha los nodos VANET, se implementa el modelo presentado en la figura 5.9 siendo similar al que se mostró en el capítulo 4, con la diferencia de que en lugar de contar con bloques de subscripción de tópicos al entorno de Gazebo, se requieren las posiciones por un tópico que obtiene dichas variables de estado por medio de las cámaras de Optitrack. En este bloque correspondiente al agente número 1, se recibe su propia posición así como su orientación en cuaterniones, para posteriormente ser convertida a radianes. A su vez, también recibe la posición del agente 2, sin embargo, esta es publicada por el mismo agente 2 cada vez que ocurre un evento. De igual manera, este agente hace consenso con un líder virtual que provee una referencia fija o trayectoria designada a la vez que realiza consenso con el agente 2. Entonces, las posiciones de ambos agentes, la trayectoria deseada y los valores de las posiciones previas, son introducidos en el bloque de control distribuido, en donde se introduce la función de consenso y la detección de eventos, donde se designa la formación deseada por medio de un offset en dicha estrategia de consenso para la realización de formaciones. En cuestión a la detección de eventos, se genera uno cada vez que alguno de los valores de posición u orientación tengan un cambio significativo con respecto a su valor previo y en ese instante se calcula de nuevo el control. Esta señal calculada pasa al bloque de conversión de velocidades en el sistema de referencia móvil a las velocidades de las ruedas para entonces ser introducidos al bloque de generación de PWM. En este, se introduce una caracterización lineal que se realizó a cada uno de los motores del vehículo, con el fin de tener una función lineal que permita calcular el ciclo de trabajo del PWM necesario para generar la velocidad que se requiere y se envían a los bloques de PWM de la Raspberry Pi. Finalmente, cada vez que se detecta algún evento, se habilita la publicación de posiciones para que el otro agente pueda actualizar igualmente sus estados.

[1] https://drive.google.com/drive/folders/16SzljHL6FA-RlMbuGIJrY-1uXJYEIMYp?usp=sharing

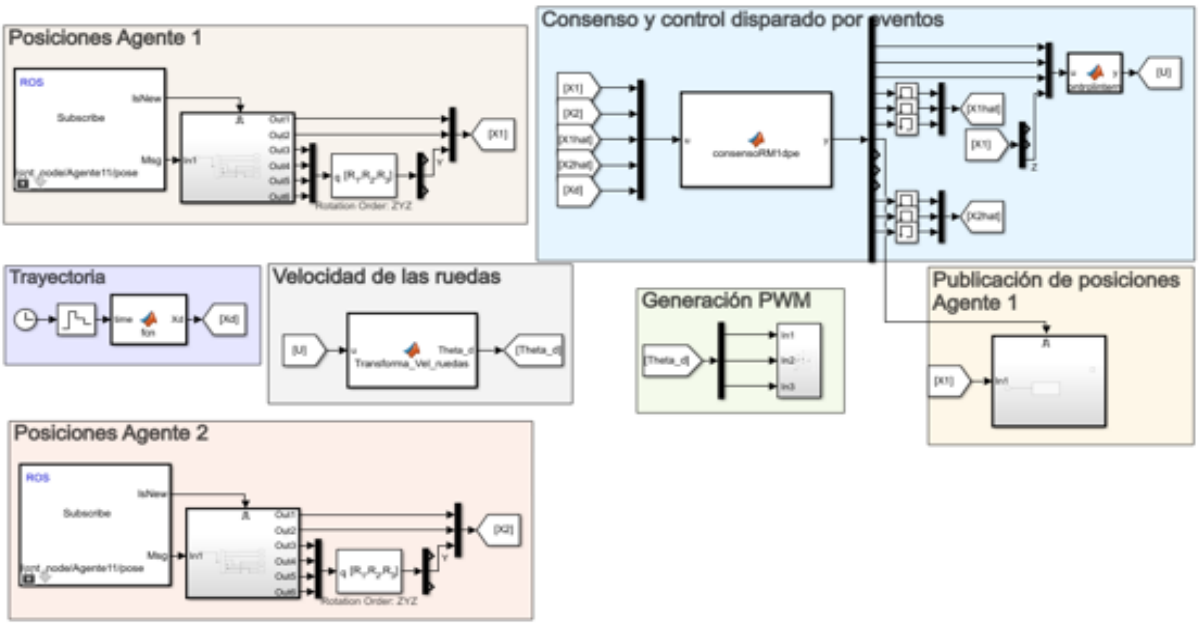

Figura 5.9: Modelo implementado en el agente 1.

La programación de ambos vehículos se realiza de manera inalámbrica por medio del modelo de la figura 5.9, esto se efectúa al conectar la máquina que contiene Matlab/Simulink y los vehículos en la misma red a través del comando “deploy” que se encuentra en la ventana de Simulink en lugar de oprimir el comando habitual “run”.

La conexión lograda para que se comuniquen ambos agentes se presenta en la figura 5.10 y consiste en un diagrama “rqt-graph” que indica la manera en la que los nodos y tópicos que fueron creados dentro de ROS interactúan entre ellos, en donde el agente 1 publica su posición en un tópico denominado “location-edi1” para que el agente 2 pueda acceder a este y así conocer la posición del agente 1 y de manera similar, el agente 2 publica su posición en el tópico “location-edi0’para que el agente 1 conozca el dato.

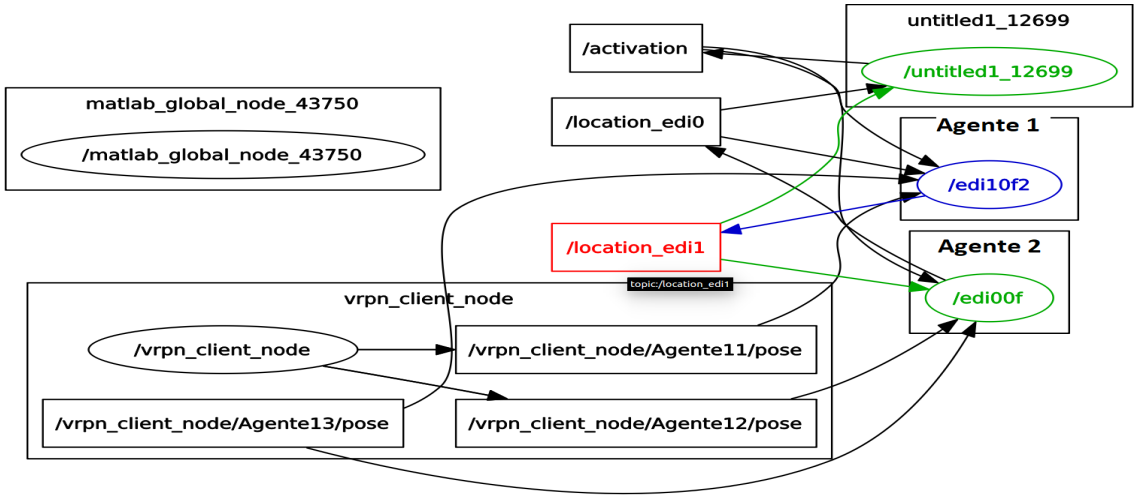

Figura 5.10: Diagrama rqt.

5.2.1. Prueba de dos vehículos con consenso

En la primera prueba se realiza un consenso entre dos nodos VANET, efectuando un distanciamiento en el eje x entre ambos agentes de 70cm. El agente 1 tiene asociado un líder virtual con posición fija asignada de 0.3m y 0.6m (estas posiciones fueron elegidas de acuerdo al perímetro del área de prueba) como destino y referencia, mientras que el agente 2 realiza la función de consenso con el agente 1. La trayectoria seguida se observa en la figura 5.11, la cual consiste en una gráfica XY de dichas variables donde parten de las condiciones iniciales de (-0.5,-1.2) y (0.5,-1.2.). Los pasos que se utilizaron entre cada iteración del procesador son de 0.00125 segundos y se efectuó esta rutina en un tiempo alrededor de 11 segundos.

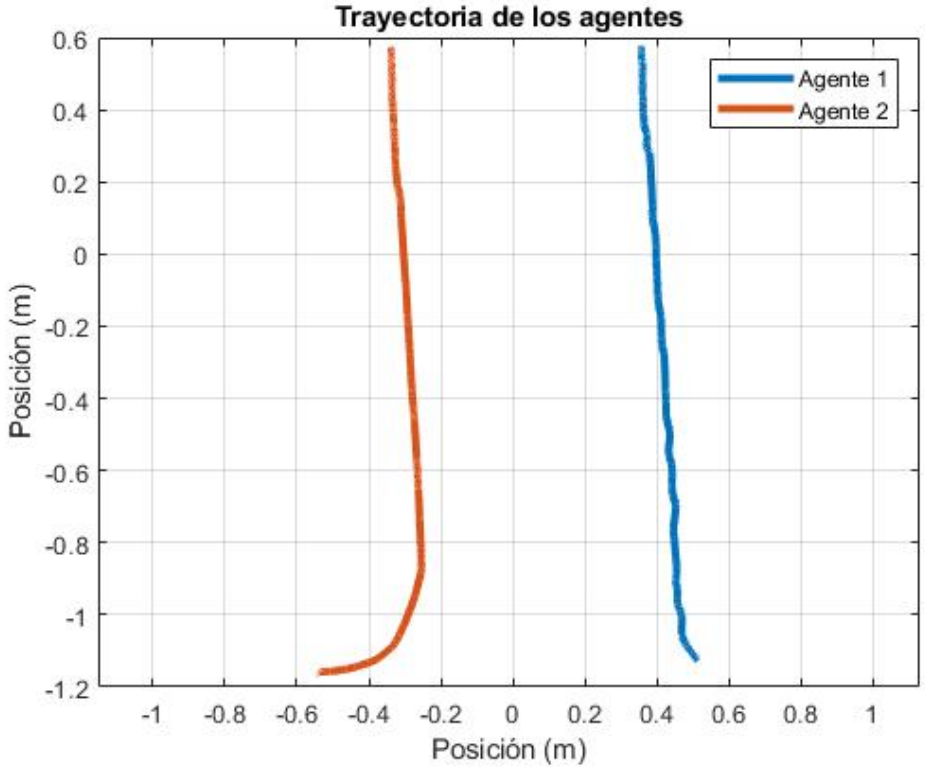

Figura 5.11: Regulación con consenso de dos agentes.

En cuanto a los eventos generados por ambos agentes se aprecian en la figura 5.12 y en la figura 5.13.

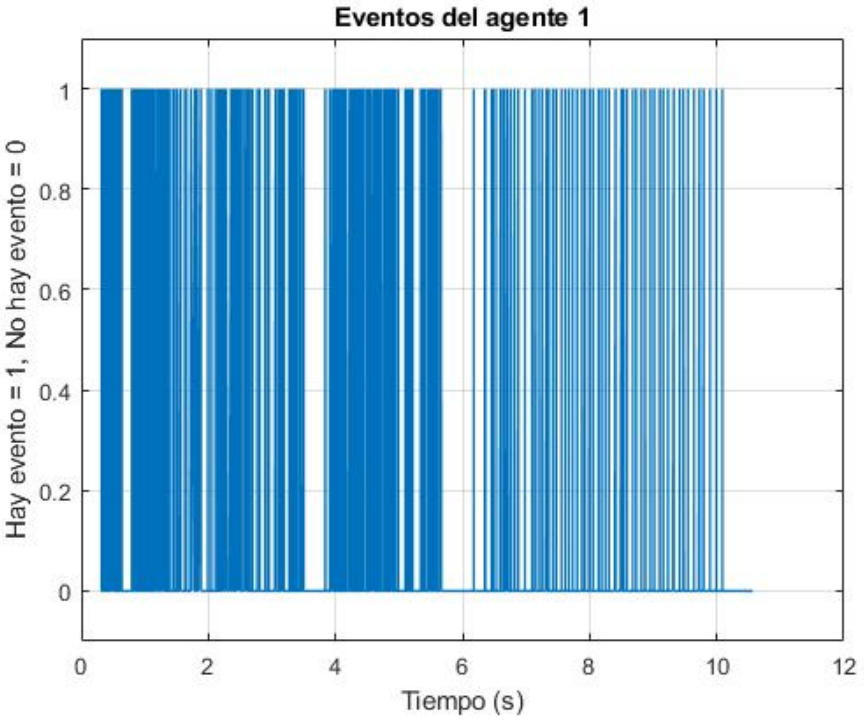

Figura 5.12: Cantidad de eventos en el agente 1.

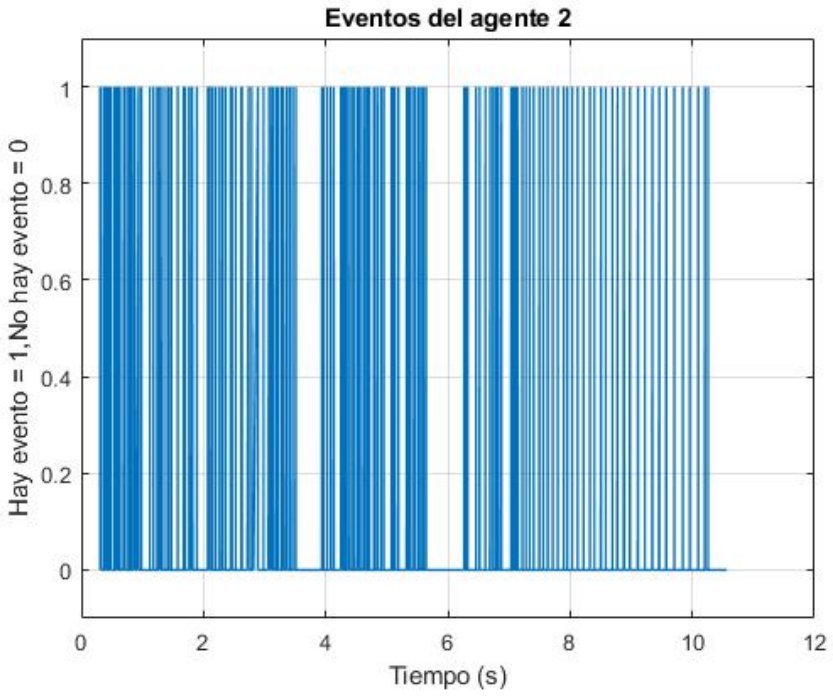

Figura 5.13: Cantidad de eventos en el agente 2.

En las figuras 5.14 y 5.15 se aprecia la comparación de la sumatoria de eventos correspondiente a cada agente con respecto a la cantidad de veces que se deberían realizar las

comunicaciones si se diera el caso de que la estrategia de control no estuviera basada en eventos.

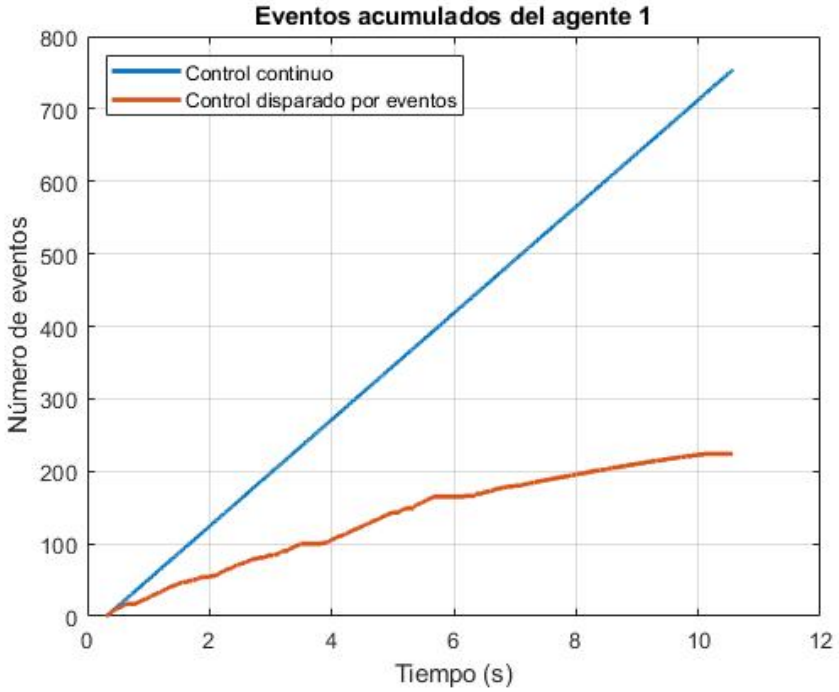

Figura 5.14: Comparación de sumatoria de eventos del agente 1.

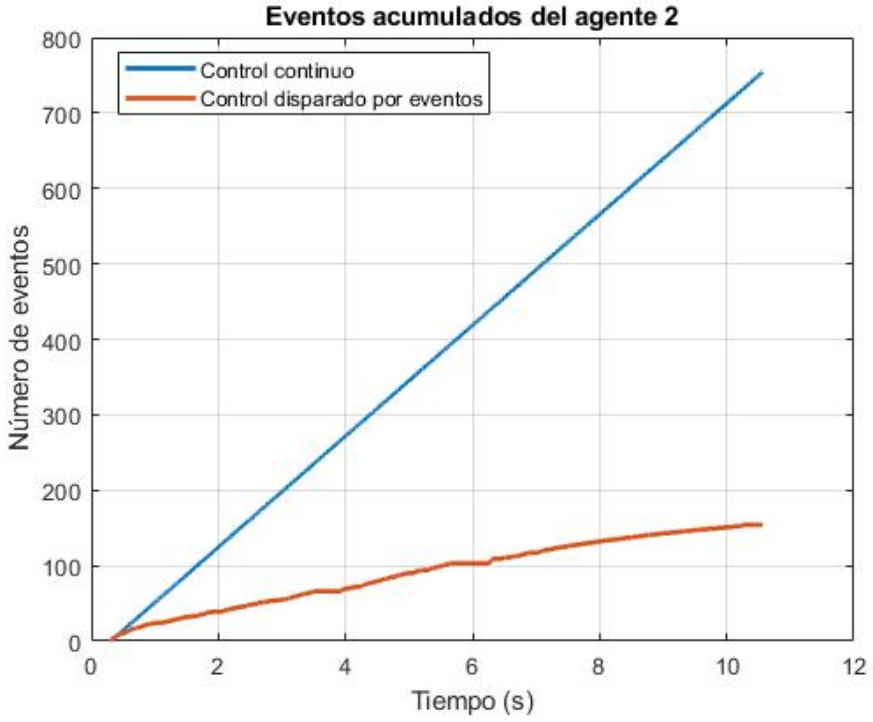

Figura 5.15: Comparación de sumatoria de eventos del agente 2.

5.2.2. Prueba de dos vehículos con consenso y evasión de colisiones entre agentes.

En esta prueba, se realizó un consenso entre ambos agentes con un distanciamiento en x de 70cm y se agregó un algoritmo de evasión entre agentes. El agente 1 funcionó nuevamente tuvo asociado un agente virtual con el que hizo consenso y se le asignó una referencia en las coordenadas (0.3m,0.6m) mientras que el agente 2 únicamente hacia el consenso. El radio de seguridad entre agentes es de ra=0.4m. La trayectoria realizada se observa en la figura 5.16. Los pasos que se utilizaron entre cada iteración del procesador son de 0.00125 segundos.

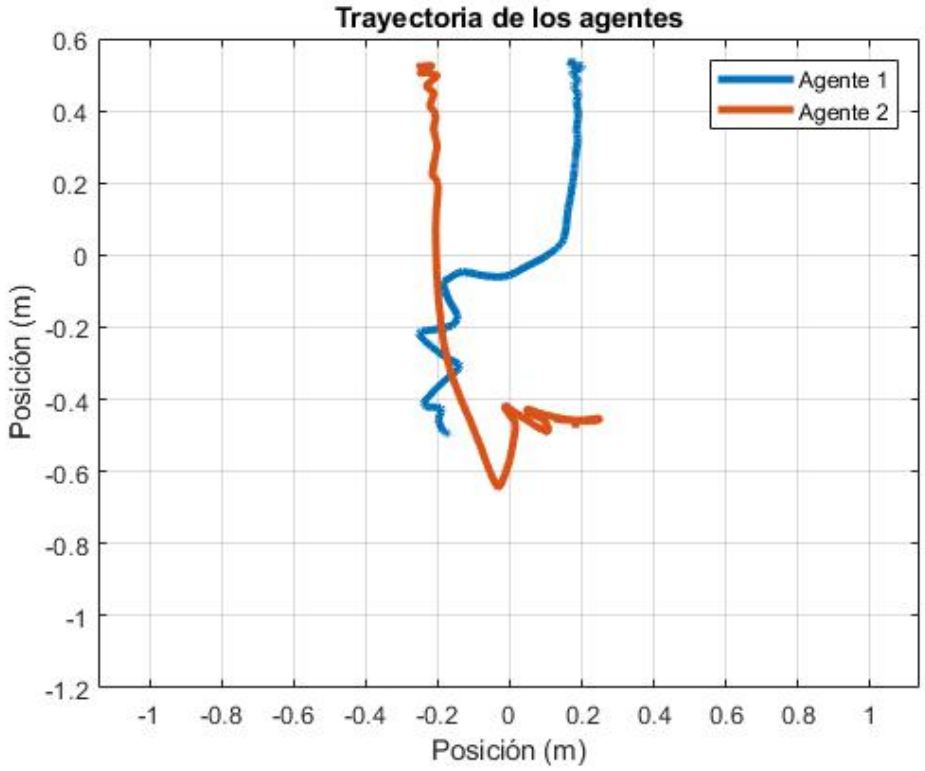

Figura 5.16: Regulación con consenso de dos agentes.

En cuanto a los eventos generados por ambos agentes se pueden apreciar en la figura 5.17 y la figura 5.18.

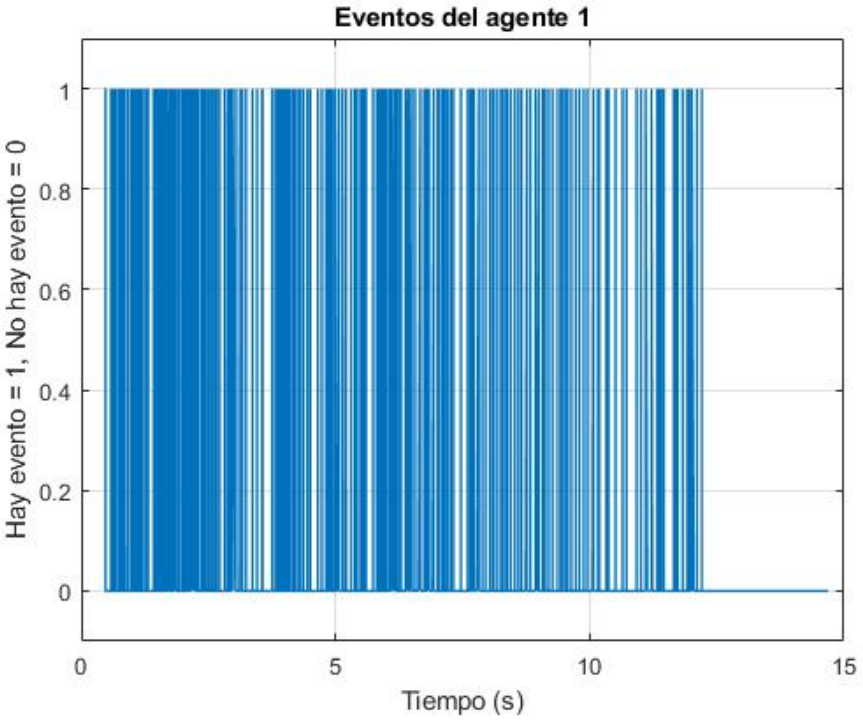

Figura 5.17: Cantidad de eventos en el agente 1.

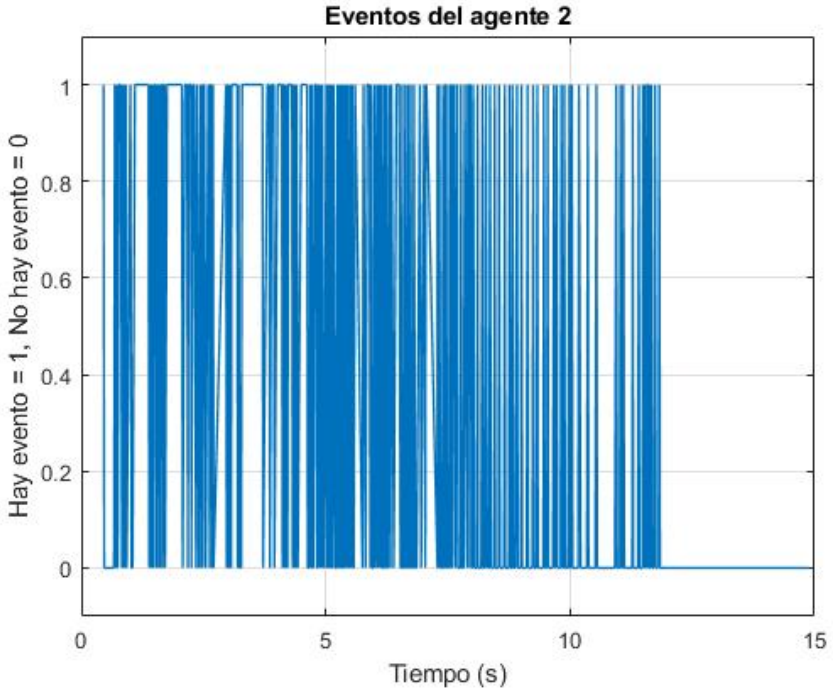

Figura 5.18: Cantidad de eventos en el agente 2.

En las figuras 5.19 y 5.20 se aprecia la comparación de la sumatoria de eventos correspondiente a cada agente con respecto a la cantidad de veces que se deberían realizar las

comunicaciones si se diera el caso de que la estrategia de control no estuviera basada en eventos.

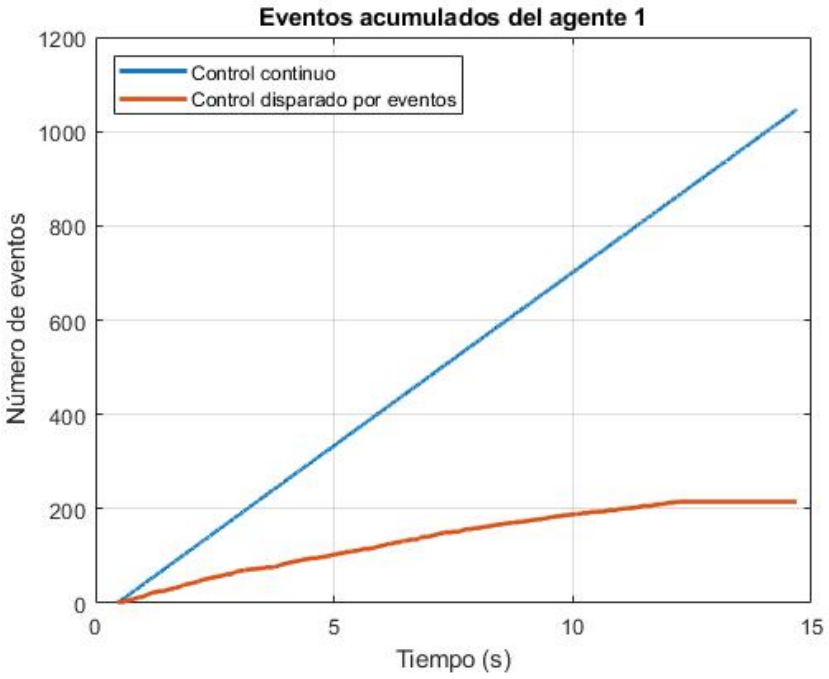

Figura 5.19: Comparación de sumatoria de eventos del agente 1.

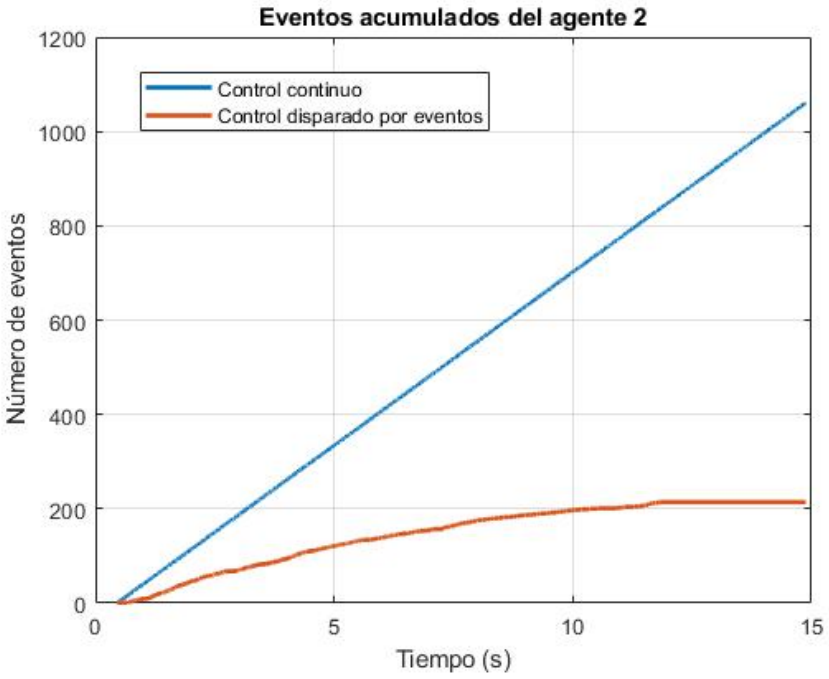

Figura 5.20: Comparación de sumatoria de eventos del agente 2.

Todos los códigos utilizados se encuentran en [2].

[2]https://drive.google.com/drive/folders/1uBRwkGk1vG0S513Thvac-MtF40M1xkHN?usp=sharing

Conclusiones

En este trabajo de tesis se diseñó un nodo VANET basado en un vehículo omnidireccional (3,0) que ejecuta formaciones comunicándose con otros nodos de características similares, por medio de una función de control distribuido basado en eventos y agregando una estrategia que permite realizar el enrutamiento necesario para que todos los nodos puedan entregar su posición a todos los elementos de la red sin importar la topología de la misma. Se realizaron las simulaciones correspondientes y con ayuda de las herramientas de Matlab/Simulink, el meta sistema operativo ROS y el simulador Gazebo, se consiguió realizar una implementación virtual de tres nodos en la que cada uno contaba con la capacidad de generar su tabla de enrutamiento y podía comunicarse con sus vecinos para realizar la estrategia de control que se le fuera designada, siguiendo una referencia y/o trayectoria por medio de un líder virtual que hiciera consenso con alguno de los agentes, logrando que cada nodo tuviera dicha programación en su propio modelo y de esa manera no hubiera una total dependencia de paqueterías externas que realicen este enrutamiento y/o el cálculo del control.

Posteriormente se realizó una implementación física de los nodos a partir de los creados previamente en Matlab/Simulink para la implementación virtual, en donde se utilizaron unos bloques compatibles con el sistema embebido Raspberry Pi 3B que fue utilizado para controlar el vehículo omnidireccional que a su vez se encontraba compuesto por un chasis, tres ruedas omnidireccionales, tres motores DC con encoders y con puentes H, baterías, etc., mientras que la posiciones de los vehículos se obtenían mediante el sistema de cámaras Optitrack y se entregaban por medio del meta sistema operativo ROS, logrando poner en marcha dos vehículos omnidireccionales que ejecutan formaciones y rutinas de seguimiento de trayectorias evitando colisiones entre ellos sin realizar enrutamiento.

Partiendo de los resultados obtenidos, se publicó un artículo en la revista IEEE xplore por parte del congreso “The Ninth International Conference on Advances in Vehicular Systems, Technologies and Applications 2020” con nombre “Event-triggered coordination of omnidirectional robots over Gazebo”. También fue aceptado un artículo en la Publicación de la Revista Digital “Memorias del Congreso Nacional de Control Automático, titulado “Collaborative event-triggered control for omnidirectional robots with obstacle avoidance”.

Trabajo futuro

Como continuación del trabajo de tesis, el primer paso sería implementar en los vehículos físicos el modelo que permita realizar el enrutamiento de los datos en redes compuestas por tres agentes o más. En adición, para emular ambientes más cercanos a los reales, se pueden utilizar diferentes nodos maestros de ROS y habilitar una comunicación entre estos con las técnicas de enrutamiento utilizadas mediante el uso de subscripción y publicación de tópicos así como el uso de servicios, e incluso disponer de paqueterías actualmente existentes para realizar este proceso, pudiendo crear redes heterogéneas con vehículos de diferentes características físicas y utilizar diferentes técnicas para la obtención de los datos del posicionamiento, como lo son el uso de un GPS de alta precisión o técnicas de odometría para determinar la posición de los vehículos y a su vez lograr evitar colisiones con obstáculos.

Bibliografía

[1] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. *Mobile ad hoc networking*. John Wiley & Sons, 2004.

[2] Sarita Singh Bhadauria and Laxmi Shrivastava. A wireless ad-hoc network: Design and performance evaluation of dynamic routing model. In *2008 International Conference on Recent Advances in Microwave Theory and Applications*, pages 546–549. IEEE, 2008.

[3] JF Guerrero-Castellanos, A Vega-Alonzo, N Marchand, S Durand, J Linares-Flores, and G Mino-Aguilar. Real-time event-based formation control of a group of vtol-uavs. In *2017 3rd International Conference on Event-Based Control, Communication and Signal Processing (EBCCSP)*, pages 1–8. IEEE, 2017.

[4] Anil D Devangavi and Rajendra Gupta. Routing protocols in vanet, a survey. In *2017 International Conference on Smart Technologies for Smart Nation (SmartTechCon)*, pages 163–167. IEEE, 2017.

[5] Shaikhul Islam Chowdhury, Won-Il Lee, Youn-Sang Choi, Guen-Young Kee, and Jae-Young Pyun. Performance evaluation of reactive routing protocols in vanet. In *The 17th Asia Pacific Conference on Communications*, pages 559–564. IEEE, 2011.

[6] Mayank Satya Prakash Sharma and Ranjeet Singh Tomar. Vanet based communication model for transportation systems. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pages 1–6. IEEE, 2016.

[7] Chandresh Pathak, Anurag Shrivastava, and Anjana Jain. Ad-hoc on demand distance vector routing protocol using dijkastra's algorithm (aodv-d) for high throughput in vanet (vehicular ad-hoc network). In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 355–359. IEEE, 2016.

[8] Haibo Du, Jun Zhou, Di Wu, and Guanghui Wen. Consensus for second-order nonlinear leader-following multi-agent systems via event-triggered control. In *2017 International Workshop on Complex Systems and Networks (IWCSN)*, pages 301–305. IEEE, 2017.

[9] Yang Zhang, Yuan Fan, Chengxiao Zhang, and Haohao Chen. Multi-agent event-triggered consensus based on distributed state observer. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 4684–4688. IEEE, 2018.

[10] Dimos V Dimarogonas, Emilio Frazzoli, and Karl H Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2011.

[11] Dimos V Dimarogonas and Karl H Johansson. Event-triggered control for multi-agent systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7131–7136. IEEE, 2009.

[12] Koya Nambo and Seiichiro Katsura. Event-triggered formation control of leader-follower multi-agent system for reducing the number of information transmission. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 7269–7274. IEEE, 2017.

[13] Alejandro Garcia-Santiago, Josefina Castaneda-Camacho, Jose F Guerrero-Castellanos, Gerardo Mino-Aguilar, and V Yair Ponce-Hinestroza. Simulation platform for a vanet using the truetime toolbox: Further result toward cyber-physical vehicle systems. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2018.

[14] Gárcia Santiago A. Diseño e implementación de una red de comunicaciones para el control de formación de vehículos aéreos no tripulados. Master's thesis, Benemérita universidad Autónoma de Puebla, Puebla,México, 2018.

[15] J Sánchez-Santana, J Guerrero-Castellanos, and M Villarreal-Cervantes. Control distribuido y disparado por eventos para la formación de robots móviles tipo (3, 0).

[16] Justin M Bradley and Ella M Atkins. Optimization and control of cyber-physical vehicle systems. *Sensors*, 15(9):23020–23049, 2015.

[17] Kyoung-Dae Kim and Panganamala R Kumar. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100(Special Centennial Issue):1287–1308, 2012.

[18] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in vanets: Protocols, applications and challenges. *Ad Hoc Networks*, 44:90–103, 2016.

[19] Sunil Taneja and Ashwani Kush. A survey of routing protocols in mobile ad hoc networks. *International Journal of innovation, Management and technology*, 1(3):279, 2010.

[20] Rolf Isermann. *Mechatronic systems: fundamentals*. Springer Science & Business Media, 2007.

[21] R Sell and M Tamre. Integration of v-model and sysml for advanced mechatronics system design. In *The 6th International Workshop on Research and Education in Mechatronics REM*, pages 276–280, 2005.

[22] Aníbal Ollero Baturone. *Robótica: manipuladores y robots móviles*. Marcombo, 2005.

[23] Saeed B Niku. *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.

[24] Javaid Khurshid and Hong Bing-Rong. Military robots-a glimpse from today and tomorrow. In *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, volume 1, pages 771–777. IEEE, 2004.

[25] Santiago Martínez and Rafael Sisto. Control y comportamiento de robots omnidireccionales. *Montevideo, Uruguay: Universidad de la República*, 2009.

[26] Frank L Lewis, Hongwei Zhang, Kristian Hengster-Movric, and Abhijit Das. *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[27] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[28] Manual de ros. `https://moodle2017-18.ua.es/moodle/mod/book/tool/print/index.php?id=2046#`. Accessed: 2020-04-07.

[29] Ros wiki. `http://wiki.ros.org/`. Accessed: 2020-04-07.

[30] Gazebosim. `http://gazebosim.org/`. Accessed: 2020-04-07.

[31] Cory Beard and William Stallings. *Wireless communication networks and systems*. Pearson, 2015.

[32] Subir Kumar Sarkar, Tiptur Gangaraju Basavaraju, and C Puttamadappa. *Ad hoc mobile wireless networks: principles, protocols and applications*. CRC Press, 2007.

[33] Prasant Mohapatra and Srikanth Krishnamurthy. *AD HOC NETWORKS: technologies and protocols*. Springer Science & Business Media, 2004.

[34] Karl-Erik Åarzén. A simple event-based pid controller. *IFAC Proceedings Volumes*, 32(2):8687–8692, 1999.

[35] Sebastián Dormido, Jose Sánchez, and Ernesto Kofman. Muestreo, control y comunicación basados en eventos. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 5(1):5–26, 2008.

[36] Wi-Fi Primer. Overview of the 802.11 physical layer and transmitter measurements. *Beaverton: Tektronix Inc*, pages 4–7, 2013.

[37] A Cervin, D Henriksson, and M Ohlin. Truetime 2.0 beta 5-reference manual, department of automatic control, lund university, sweden, june 2010. cit: 10.10. 2011.

[38] Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. Simulation environment for mobile robots testing using ros and gazebo. In *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 96–101. IEEE, 2016.

Apéndice A

Publicaciones

Articulo 1. *Event-triggered coordination of omnidirectional robots over Gazebo* publicado en 2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP) en Krakow, Poland, Poland del 23 al 25 septiembre del 2020. ISBN: 978-1-7281-9581-0.

Articulo 2. *Collaborative event-triggered control for omnidirectional robots with obstacle avoidance* publicado en Memorias del Congreso Nacional de Control Automático 2020, revista digital. ISSN: 2594-2492.

Event-triggered coordination of omnidirectional robots over Gazebo

M. Longinos-Garrido, N. A. Tolentino-Medrano, J. F. Guerrero-Castellanos, J. Castañeda-Camacho and R. Álvarez-González
Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Electrónica,
Av. San Claudio y 18 Sur, Col. San Manuel,
Puebla, Pue. México.C.P. 72570
Email: longinos.200918526@gmail.com, nestor.tolentino16@gmail.com, fermi.guerrero@correo.buap.mx
josefina.castaneda@correo.buap.mx, ricardo.alvarez@correo.buap.mx

***Abstract*—This work presents a Gazebo Simulation of a Multi-agent system conformed by Omnidirectional (3,0) mobile robots. The model consists of integrating a collaborative control that allows agents to achieve consensus and avoid collisions among them and against obstacles. The control strategy for the communication and agreement is event-based to adapt the simulation to the asynchronous behavior of this type of system, while the collision avoidance is in continuous time. The implementation is carried out relying on ROS and MATLAB/Simulink tools to implement the vehicle model and allow the exchange of information between agents. A virtual leader is introduced to provide the agents with a reference and a trajectory to follow.**

***Index Terms*—ROS, Gazebo, CPS, Multi-Agent systems, Cooperative control, Event-triggered control, (3,0) Omnidirectional mobile robot**

I. INTRODUCTION

A. Background and literature review

Cyber-physical systems (CPS) consist in computing-based systems that acquire signals with sensors and process them relying in adaptable and intelligent software resources, all of this in a feedback loop with the purpose of controlling these processes, allowing to achieve stability, performance and robustness [1], [2]. Cyber-physical vehicular systems (CPVS) emerge as a particular area of interest among CPS focusing in generating the necessary mobility of these autonomous agents considering aspects in control, communications and computational procedures [2]. Given the tendency of Industry 4.0 [3] CPVS need to be adapted to interact with other systems of similar nature in order to accomplish specific goals creating in this way Multi-Agent Systems (MAS) which gives the advantage of providing parallel processing, asynchronous operation, scalability, among other traits [4]. In most of the cases elements forming a MAS must exchange information of its physical states variables and dynamics with other agents of the group in order to achieve an agreement in such variables allowing to perform coordinated actions, requiring then a collaborative control strategy for the complete system [5].

In a distributed cooperative control, consensus between an agent and its surrounding neighbors permits to create formations by reaching certain positions with a desired offset between them that helps the MAS to carry out different tasks [5]. It is possible as well to provide either changing trajectories or references to the agents in a flocking behavior or by adding a virtual leader to one of the agents. Given the fact that the elements of a MAS have an asynchronous nature, is necessary to introduce an event-triggered control paradigm to allow the control strategy and the communications to be executed within this non-periodic scheme. In this principle, only if an error from the controlled signal exceeds a certain threshold or when a condition is met, the control will be updated and thereby the exchange of information with externals agents will be activated, reducing like this the usage of computational resources [6], [7], [8]. In addition, for autonomous vehicles that form a wireless mobile network, the event-based approach is highly related to a distance-based send-on-Delta strategy, where the position of the vehicle is tracked, and only when a significant change in its value takes place, the new signal value is transmitted to the other agents in the network, reducing the number of messages sent, the network traffic and reducing as well the energy consumption as communications are the major source of it [9], [10]. In MAS involving CPVS there are different kinds of mobile robots that can be implemented, but two typical types of unmanned vehicles that are usually integrated consist in aerial vehicles (UAVs) and ground vehicles (UGVs), which suits in tasks requiring surveillance, area recognition, rescue operations, agriculture support for field operations, among other applications and can work together in heterogeneous implementations [11], [12], [13]. UGVs take an important role when the terrain must be explored, in military applications and their dynamics are more simple than aerial vehicles [14]. Omnidirectional mobile robots allow to have a most efficient mobility since these vehicles can freely move in spite of its current orientation, allowing to fulfill a better performance for tasks with space restriction [15], [16]. An omnidirectional mobile robot (3,0) can coordinate its movement with its 3 mecanum wheels, by a combination of each wheel's velocity allowing the robot to freely move in the plane without any dependency of its angular positioning [17], [18], [19].

978-1-7281-9581-0/20//$31.00 ©2020 IEEE

Robotic Development Environments (RDEs) have become an increasingly important piece in the robotics research field in general and for the development of architectures for mobile robots in particular [20], since the design workflow for robotic systems usually starts by using a simulation software to create virtual environments for test settings that require robot-environment interaction [21]. One of the most popular RDEs in the present is Gazebo, which presents many advantages as making the simulation world more realistic, not only for humans but for robots by including many sensors and actuators in its increasingly open source library and the compatibility with other projects like the Player and Stage one. Therefore, Gazebo is designed to accurately reproduce the dynamic environments that a robot may encounter [22].

In parallel with this development, a new flexible framework for robot applications has emerged in the recent years; the Robot Operating System (ROS) environment, an open source for robot operating systems that provides a communication layer and a data programming structure [23], [24]. This tool helps to manage different aspects of a robotic system implementation, such as the hardware abstraction layer and management of the communications [25]. It also provides some advantages in order to use the development across different software/hardware implementations.

B. Contributions

The main contribution from this work is the integration of an event based consensus control for communications between agents and a collision avoidance continuous control in a Gazebo simulation using ROS and MATLAB/Simulink tools to create the models and network, while contrasting the performance of these approaches with a non-event model.
The rest of the paper is organized as follows. Section II presents the mathematical background, graph theory, system's model, and control strategy. Section III is devoted to giving the simulation environment, as well as the simulator modules. In section IV, the results are presented and discussed. Finally, the conclusion and future work are given in Section V.

II. Preliminaries

A. Graph theory

A graph $G = (V, E)$ consists in a set of nodes $V = \{v_1 ... v_N\}$, where N is the total number of agents, and a set of edges $E(i, j)$, where $i, j \in V$ and it defines the interactions between the nodes that conform a MAS. In a leader-following configuration, the leader is represented by an extra vertex 0 and information is exchanged between the leader and the following agents which are in its neighborhood. The leader is a *virtual* system in the present study. Then, such a configuration can be defined with a graph $\bar{G}$, which consists of graph G, vertex 0 and edges between the leader 0 and its neighbors. Furthermore, let $\mathbf{G} = \text{diag}(g_1, \ldots, g_N)$ be the diagonal matrix of pinning gains, $g_i > 0$, describing the connections between the leader and the follower nodes [5], [26]. In this paper, if $g_i = 1$ the node i is said to be pinned to the leader, i.e. the node i observes the leader and an edge $(0, i)$ is said to exist.

B. Omnidirectional (3,0) mobile robot

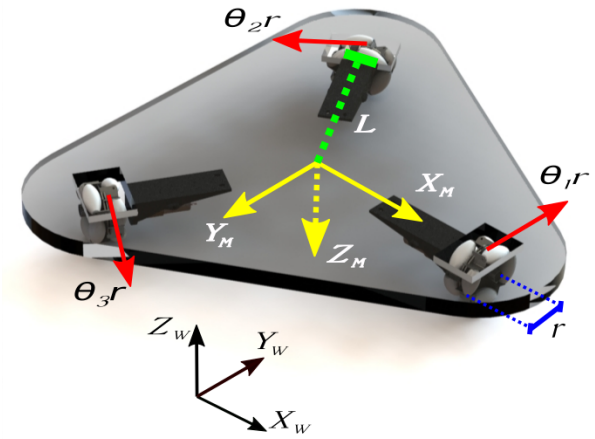

Fig. 1. Omnidirectional (3,0) mobile robot model

The omnidirectional (3,0) mobile robot model is shown in Fig. 1 where, to achieve the movement of the vehicle in the inertial plane W, is necessary to transform the set of inertial linear and angular velocities $\dot{\eta}_w = \begin{bmatrix} \dot{x}_w & \dot{y}_w & \dot{\phi}_w \end{bmatrix}^T$ to the set of mobile linear and angular velocities $\dot{\eta}_m = \begin{bmatrix} \dot{x}_m & \dot{y}_m & \dot{\phi}_m \end{bmatrix}^T$ corresponding to the mobile plane M. Then, the kinematic model of the (3,0) robot is [17], [18]

$$\begin{aligned} \dot{X}_w &= \dot{X}_m cos\phi_w - \dot{Y}_m sin\phi_w \\ \dot{Y}_w &= \dot{X}_m sin\phi_w + \dot{Y}_m cos\phi_w \\ \dot{\phi}_w &= \dot{\phi}_m \end{aligned} \tag{1}$$

It is considered that the omnidirectional (3,0) mobile robot has a rigid structure with non-deformable and non-slipable wheels and with a movement over an horizontal plane with only one contact point between the wheel and the plane. The mapping of the mobile velocities to the angular velocities of the wheels is represented by

$$\begin{aligned} \dot{\theta}_1 r &= \tfrac{1}{2}\sqrt{3}\dot{X}_m + \tfrac{1}{2}\dot{Y}_m + L\dot{\phi}_m \\ \dot{\theta}_2 r &= -\dot{Y}_m + L\dot{\phi}_m \\ \dot{\theta}_3 r &= -\tfrac{1}{2}\sqrt{3}\dot{X}_m + \tfrac{1}{2}\dot{Y}_m + L\dot{\phi}_m \end{aligned} \tag{2}$$

This mapping allows us to design local controls (PI controllers) to regulate each wheel's angular velocity.

C. Event-triggered control

An event based communication system between agents starts by an event generator, with an event function e_i that will monitor the current state and a memory that stores the state's value the last time the event generator was activated. If the difference between the current state and the memory value exceeds a threshold c the memory value will be updated with the current state with $e_i \geq c$, otherwise, the memory will not be updated with $e_i < c$. The next element is a static distributed

control function $u_i = k_i(x_i, m_i, m_j)$ which depends in the actual state, its own memory value and the memories from the neighbors j.

Let us consider N (3,0) omnidirectional mobile robots connected with a graph G, and with $V = \{1, 2, ..., N\}$. Let $x_i = \begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \end{pmatrix}^T$ be the state vector of the ith robot in the inertial frame W, being $x_{i,1}, x_{i,2}$ the positions in the coordinate plane and $x_{i,3}$ the orientation. In addition, $m_i = \begin{pmatrix} m_{i,1} & m_{i,2} & m_{i,3} \end{pmatrix}^T$ represents the memory of the state x_i, the last time an event took place. Furthermore, let $u_i = (u_{i,1}\ u_{i,2}\ u_{i,3})^T$ the ith vehicle's linear and angular position in the body frame, which will be considered the control. Then, the ith vehicle's kinematic model is given by

$$\begin{aligned} \dot{x}_{i,1} &= u_{i,1}cos(x_{i,3}) - u_{i,2}sin(x_{i,3}) \\ \dot{x}_{i,2} &= u_{i,1}sin(x_{i,3}) + u_{i,2}cos(x_{i,3}) \\ \dot{x}_{i,3} &= u_{i,3} \end{aligned} \tag{3}$$

Now, let us consider the following variable's change

$$\begin{aligned} u_{i,1} &= cos(x_{i,3})\bar{u}_{i,1} - sin(x_{i,3})\bar{u}_{i,2} \\ u_{i,2} &= sin(x_{i,3})\bar{u}_{i,1} + cos(x_{i,3})\bar{u}_{i,2} \\ u_{i,3} &= \bar{u}_{i,3} \end{aligned} \tag{4}$$

Then, the model for the ith robot becomes

$$\begin{aligned} \dot{x}_{i,1} &= \bar{u}_{i,1} \\ \dot{x}_{i,2} &= \bar{u}_{i,2} \\ \dot{x}_{i,3} &= \bar{u}_{i,3} \end{aligned} \tag{5}$$

Remark 1: The variable's change represents an inner controller's implementation to regulate the robot's wheels' speed. This internal controller must be sufficiently fast, such that the mathematical procedure (4) has sens.

Remark 2: The inner controller is implemented via three PI controllers to regulate the speed of each wheel. The desired wheel's speed is obtained through (2), that is,

$$\begin{aligned} \dot{\theta}^d_{i,1} &= \frac{1}{2r}\sqrt{3}u_{i,1} + \frac{1}{2r}u_{i,2} + \frac{L}{r}u_{i,3} \\ \dot{\theta}^d_{i,2} &= -\frac{1}{r}u_{i,2} + \frac{L}{r}u_{i,3} \\ \dot{\theta}^d_{i,3} &= -\frac{1}{2r}\sqrt{3}u_{i,1} + \frac{1}{2r}u_{i,2} + \frac{L}{r}u_{i,3} \end{aligned} \tag{6}$$

Inspired in [27], the following strategy (13) allows the leader-Following Consensus control for the multi robot system using an event-triggered communication.

$$\bar{u}_{i,s} = \kappa \left[\sum_{j \in \mathcal{N}_i} (m_{j,s} - m_{i,s}) + g_i(\xi_{0,s} - m_{i,s}) \right] \tag{7}$$

with $s \in \{1, 2, 3\}$, $\kappa > 0$, g_i the pinning gains ($g_i = 1$ if the node i observes the leader and $g_i = 0$ otherwise), $\xi_{0,s}$ denotes the way-points (reference positions) assigned by the virtual leader. The event function that depends on x_i and m_i and allows the i^{th} robot broadcasts its state to its neighbors is

$$e_i(x_i, m_i) = e_{i,1} \wedge e_{i,2} \wedge e_{i,3} \tag{8}$$

where the symbol $\wedge$ denotes the "or" logical connective. Moreover:

$$e_{i,s} = \begin{cases} 1 & \text{if } |m_{i,s} - x_{i,s}| - c \geq 0 \\ 0 & \text{elsewhere} \end{cases} \tag{9}$$

$c \in \mathbb{R}_+$ is a threshold variable that allows regulating the number of events, *i.e.*, the traffic flow on the communication network. The event function (9) will rule the update of the memory values with their current state variables by

$$m_{i,s} = \begin{cases} x_{i,s} & \text{for } e_{i,s} = 1 \\ m_{i,s} & \text{for } e_{i,s} = 0 \end{cases} \tag{10}$$

The consensus strategy (13) can be extended to the formation one. For that, the desired shape is described through an associated location set F given by

$$F = \{\zeta_1, \zeta_2, ...\zeta_N\},\ \zeta_i \in \mathbb{R}^3,\ i = 1, .., N \tag{11}$$

where

$$\|\zeta_i - \zeta_j\| = \varrho_{ij} \tag{12}$$

Then, the control strategy (13) becomes:

$$\bar{u}_{i,s} = \kappa \left[\sum_{j \in \mathcal{N}_i} (m_{j,s} - m_{i,s}) - (\zeta_{j,s} - \zeta_{i,s}) + g_i(\xi_{0,s} - m_{i,s}) \right] \tag{13}$$

$d_i = (\zeta_{j,s} - \zeta_{i,s})$ represents the plane inter-distance and the angular offset between the ith and jth robot allowing formations.

D. Collision avoidance

In robotic applications, it is essential to consider the environment to avoid collision between neighbors and obstacles. The work in [28] presents an algorithm for avoiding obstacles and agents collisions. Adapting the mathematical models from [28] using the state's memories, and consolidating all terms results in

$$u_{i,s} = \bar{u}_{i,s} + u^{\alpha}_{i,s} + u^{\beta}_{i,s} \tag{14}$$

where $\bar{u}_{i,s}$ is given by (13), $u^{\alpha}_{i,s}$ involves the collision avoidance algorithm against agents and $u^{\beta}_{i,s}$ is the collision avoidance algorithm against obstacles both in continuous time which are not developed in detail here due to space's restrictions.

E. Robot operating system (ROS)

As it has been mentioned earlier ROS is a framework that brings many advantages for robot implementation. To make it easy to maintain and scale, ROS works as a set of interconnected programs or processes, these are seen by ROS as nodes. There also exist the topics, these are named buses through which the nodes can communicate. A node in the network can publish its own data through a topic or can subscribe to one to receive all the data published by other nodes.

The most important part of the ROS network is the ROS master, which is an intermediate program that handles the communication between the nodes and registers all the nodes and connections in the network. An illustration on how does the ROS network is set can be seen in Fig. 2

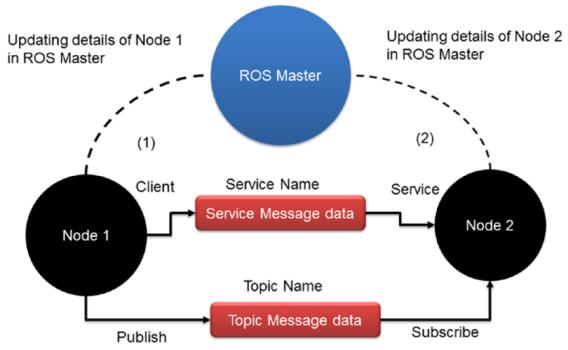

Fig. 2. ROS communication block diagram [29]

III. Simulation

A. Overview

A simulation of a MAS formed by two omnidirectional (3,0) mobile robots in Gazebo is presented. The scheme consists of a virtual leader associated with Agent 1 and will provide a reference and trajectory. Agent 1 and Agent 2 will follow the path, seeking to achieve a formation keeping a distance of 0.7m from each other, avoiding collisions among them and against an obstacle. The model of each vehicle is built in MATLAB/Simulink and introduced as ROS nodes to Gazebo platform, allowing to observe its behavior based on the created model. To make the system accurate regarding the communications between agents, we proposed that they get his partner's position from the theme instead of getting both; the own position and the partner one from the gazebo environment.

B. Simulink models

MATLAB/Simulink has add-ons that allows to work with ROS applications by incorporating blocks for publishing messages or subscribing to topics. Fig. 3 presents the Simulink model for one agent. It contains 3 subscribe topic blocks, where the first one allows to obtain its own position from Gazebo's environment, the second gets the position that the other Agent is publishing while the last one subscribes to a topic that contains the position of the predefined and well known obstacle. The trajectory generator will only be included to the Agent that has the virtual leader associated, giving the reference points or the desired function for the trajectory. The event based consensus and collision avoidance establishes the control laws from (14) using the values that comes from the subscribing blocks and the previous states from the memories.The robot dynamics block introduces the mathematical model that corresponds from the omnidirectional (3,0) mobile robots in (1) and its output its converted to each wheel's velocity by (2) to enter in a continuous PI control where the output is saturated at a maximum speed of 12 rad/s and is converted again by the inverse of (2) to the velocities in the mobile reference system, while the two remaining blocks allow to publish the Agent's own position for Gazebo's environment and for the other Agent respectively. To incorporate the Simulink model to the simulator, ROS is initiated in the Matlab command window and the Simulink diagram is built generating like this a "$.tgz$" file that can be incorporated in the ROS workspace by executing the script provided at build time.

C. Gazebo implementation

The use of the Gazebo RDE, starts by designing the robot it self and can be archived by the description of simple geometries, or the inclusion of meshes, designed in an CAD software in the main URDF file, which is based in the XML structure, where it is described the relation between the robot components as it is shown in Fig. 4. Here, the black rectangles represents the links of the robot while the blue ovals represents the joints that correlate every parent link with its son, this relation is driven by the offset values presented above the joints and correspond to a position offset along the X,Y, and Z axes and a rotation along $roll$, $pitch$ and yaw all this with respect to the parent link.
It is a common practice to start every mobile robot with a empty joint so it can be related to the plugin that make it possible to move the robot in the Gazebo environment, in this case it is the planar move plugin the one that allows a robot to make any linear or planar move in one plane.

Since the meshes simulation requires a lot of resources from the computer, it has been decided to use a simplified geometrical representation, that is recommended for a low specs computer system.

From the Gazebo environment can be obtained the state of the elements in the simulation through an API called "$robotstate\ \ publisher$", which publishes a topic called "$model\ \ state$", from here it can be obtained the position and orientation of the elements in the environment, for this case just the agent_1, agent_2 and one obstacle, after every robot gets its parameters each one computes its own linear and angular velocity, which closes the loop to the gazebo environment through the "$planar\ \ move$" so it updates the agents positions in the RDE as shown in Fig. 5.

In the graph presented in the Fig. 5, the big rectangle at the top represents the namespace that holds the gazebo environment, the smaller rectangles including the one inside the namespace represents the topics and finally the ovals represents the nodes that publish to a topic trough the arrows coming out and subscribe with the arrows coming in.

D. Launch files

To simplify the execution of a simulation ROS provides a tool called "$roslaunch$", which is used to execute different programs at the same time, it also triggers the Gazebo spawn

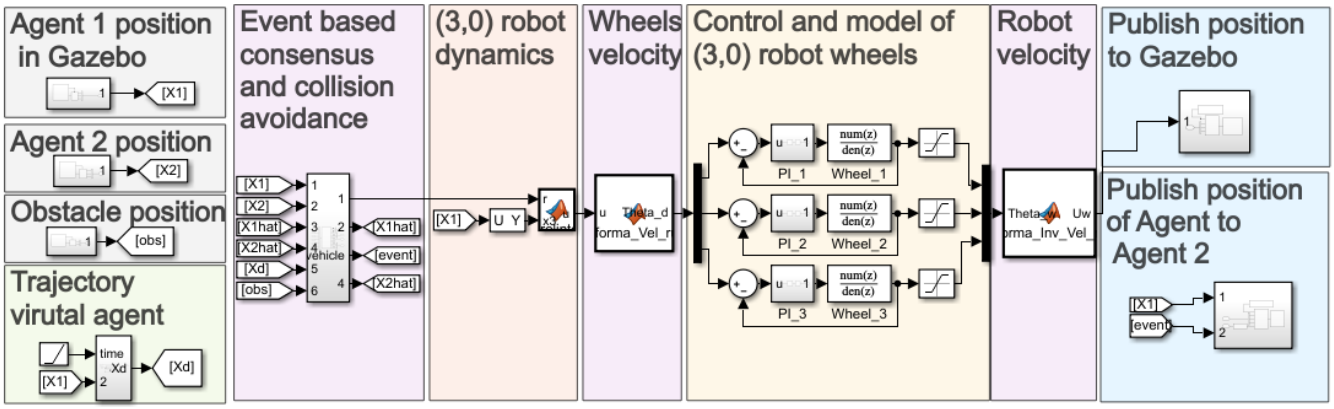

Fig. 3. Simulink model of (3,0) robot with ROS blocks

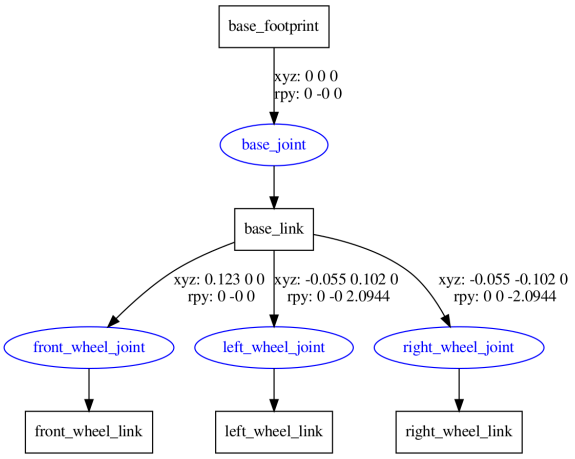

Fig. 4. Robot graph described in the URDF file.

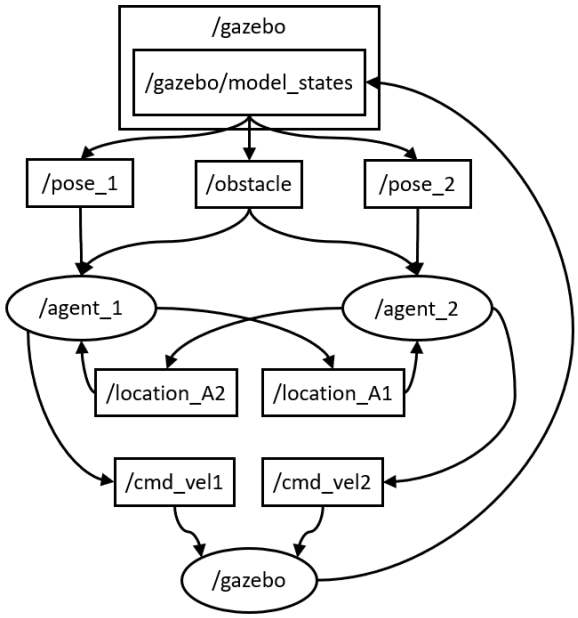

Fig. 5. ROS nodes communication graph.

service, that is responsible for positioning the agents and the obstacle at their initial positions. The launch files called with those tool have XML Format, and with the tag $< node >$ it is specified the nodes to be launched, indicating its package and name, that for the simulink build nodes are the same as the $*.slx.$ file.

E. ROS bags

One important tool for the evaluation of these systems is the ROS bag, which is a tool that makes a timed log of all the messages trough an specified topic or all of them. Later on, this data can be processed for a further analysis.

IV. Results

For this test, it is implemented a formation where the Agent 2 will keep a distance of 0.7m in the Y direction and no offset in the X direction with respecting of Agent 1 and vice versa, while the Agent 1 will reach consensus with the virtual leader. The route of this virtual leader has been decided to be divided in two parts, the first one corresponds to a regulation of the desired position in the coordinates (2.0m,0.0) with an obstacle located at (0.0,0.0), in order to test the obstacle avoidance and also avoiding collisions from each other and once the regulation has been reached, it will start the trajectory following stage, this part of the test is driven by the circle parametrization described by the equations $\xi_{0,1} = Asin(\omega t)$ and $\xi_{0,2} = Acos(\omega t)$ where $A = 2$ and $\omega = 0.1$ evaluated in a range from $t = 0$ to $t = 4\pi$ making this way two full turns.

In order to test the performance of the event-triggered control it is proposed to measure the Integral Square Error (ISE) of the test using different values of the C constant. Taking 3 different scenarios with $C = 0.01$, $C = 0.055$ and $C = 0.1$ for all $e_{i,s}$ adding as well a continuous time model without event-triggered control to contrast with the other cases.

From the experiment driven it can be assumed that the obtained performance corresponding to a $C = 0.01$ value is equal to the case of a continuous time control (denoted as CT).

The path traveled by the two Agents using the $C = 0.01$ is shown in Fig. 6 while Fig. 7 shows the worst case with $C = 0.1$.

By making a closer examination of the results, we can observe in Fig. 8 and Fig. 9, the individual performance that has been reached by each agent in the different c values observing the square error graphs while Fig.10 and Fig. 11 presents the performance achieved during the trajectory following section, showing its square error.

From the previous graphs it can be calculated the ISE, by applying a trapezoidal integral function to the discrete values, obtaining this way the results from TABLE I and TABLE II

TABLE I
ISE FROM REGULATION TEST

Agent	*CT*	*c=0.01*	*c=0.055*	*c=0.1*
Agent 1	46.973	47.320	48.780	60.845
Agent 2	101.708	102.943	106.002	117.090

TABLE II
ISE FROM TRAJECTORY TEST

Agent	*CT*	*c=0.01*	*c=0.055*	*c=0.1*
Agent 1	21.060	20.545	18.349	19.977
Agent 2	47.385	46.625	43.331	45.764

It is observed that the values of ISE have just a slightly variation in contrast to the different threshold tests, being the greatest difference of 13.872 units for Agent 1 with respecting of the continuous time model in regulation stage in the worst case of $c = 0.1$ while the best case of $c = 0.01$ only has a difference of 0.347 units,and for the Agent 2, in regulation stage as well, there is a difference of 15.0382 for $c = 0.1$ and 1.235 for $c = 0.01$ while for the trajectory stage, the greatest difference in Agent 1 was of 2.38 units and 3.41 for Agent 2 .

Afterwards, Fig. 12 and Fig. 13 shows the accumulated events from each agent for the 3 different values of c in comparison to the continuous time model, representing the times that the control signal is updated. The total amount of events for each case are shown in TABLE III.

TABLE III
NUMBER OF EVENTS PER TEST

Agent	*CT*	*c=0.01*	*c=0.055*	*c=0.1*
Agent 1	15350	2295	454	265
Agent 2	15350	2373	473	266

It can be noticed that the proportion in the numbers of events triggered with a threshold value of $c = 0.01$ from respect to the continuous time model where around 15%, while for the value of $c = 0.055$ were 3% and around 2% for $c = 0.1$.

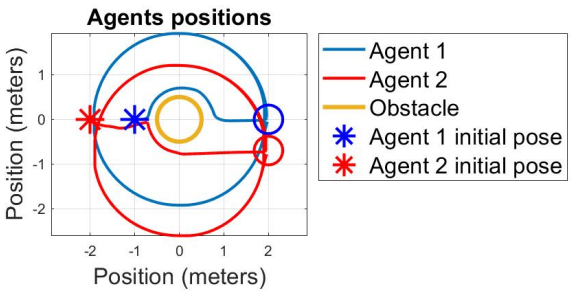

Fig. 6. Agents trajectories with $c = 0.01$ and continuous time.

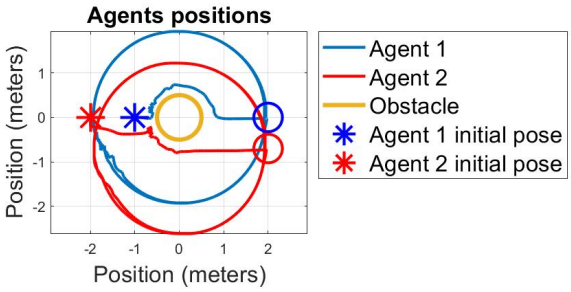

Fig. 7. Agents trajectories with $c = 0.1$.

Fig. 8. Regulation error square in Agent 1 for ISE calculation.

Finally, the simulation of these tests is presented in the videos stored in the drive [30].

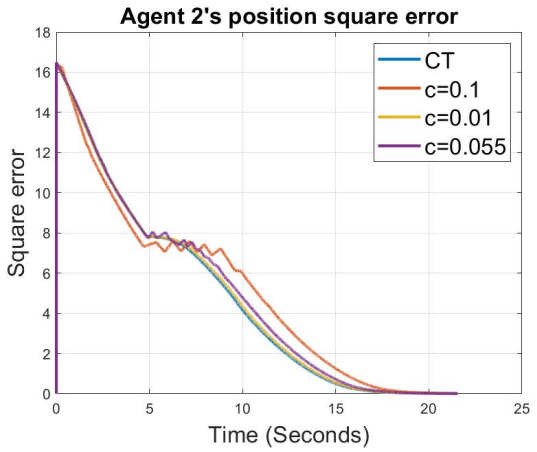

Fig. 9. Regulation error square in Agent 2 for ISE calculation.

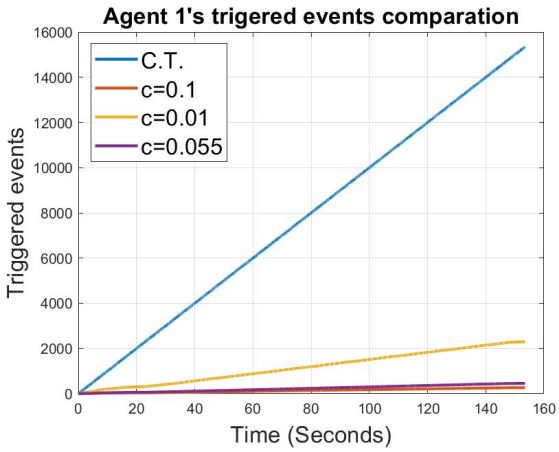

Fig. 12. Events comparison in Agent 1.

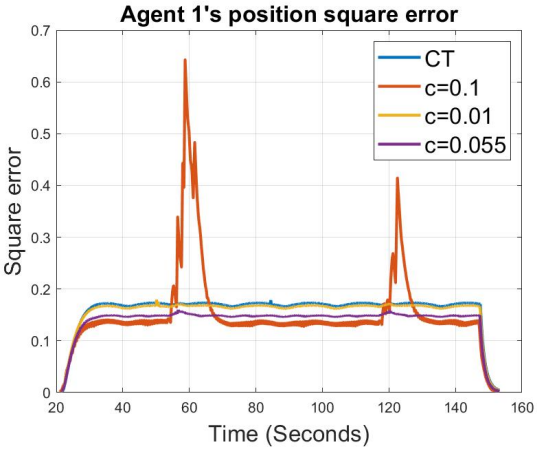

Fig. 10. Trajectory square error in Agent 1.

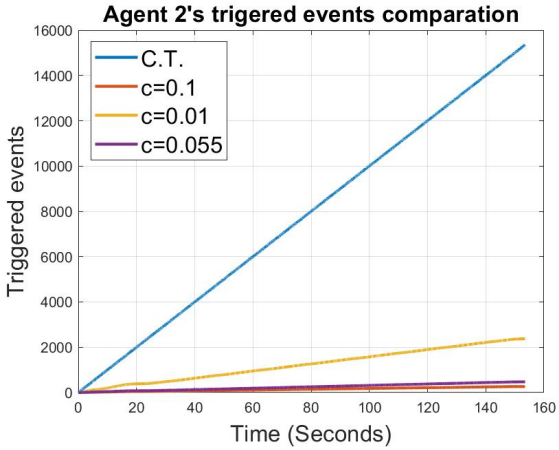

Fig. 13. Events comparison in Agent 2.

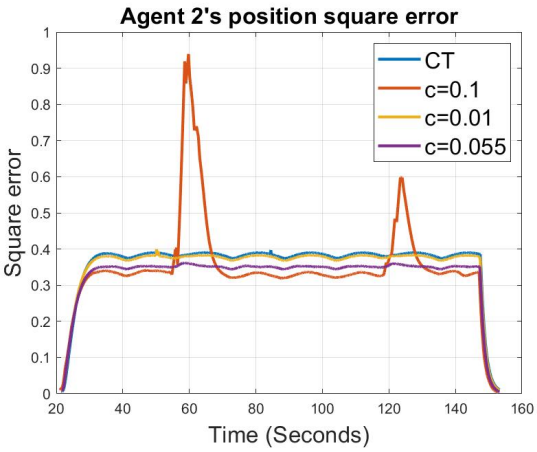

Fig. 11. Trajectory square error in Agent 2.

V. CONCLUSIONS

A Gazebo's simulation of an event-triggered multi-agent system was implemented to show the advantages and disadvantages of an event-triggered system. A quantitative comparison of the controllers' efficiency at different threshold values for a continuous-time controller was made. ISE index was used to compare the different cases, showing just a slight difference among them. This study allows us to conclude that implementing an event-based paradigm in this class of systems doesn't adversely affect the performance, allowing less use of the available resources. As future work, we'll implement the presented methodology and control strategy in real-time using a set of omnidirectional robots provided of a Raspberry board and a sensor suite to obstacle avoidance. The vehicles' position will be obtained via an Optitrack system, and the communication layer will be established via ROS.

REFERENCES

[1] E. A. Lee, "Computing foundations and practice for cyber-physical systems: A preliminary report," *University of California, Berkeley, Tech. Rep. UCB/EECS-2007-72*, vol. 21, 2007.

[2] J. M. Bradley and E. M. Atkins, "Optimization and control of cyber-physical vehicle systems," *Sensors*, vol. 15, no. 9, pp. 23 020–23 049, 2015.

[3] S. Patnaik, *New Paradigm of Industry 4.0: Internet of Things, Big Data & Cyber Physical Systems*. Springer Nature, 2019, vol. 64.

[4] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[5] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[6] M. Miskowicz, *Event-Based Control and Signal Processing*. CRC Press, 2015.

[7] K.-E. Åarzén, "A simple event-based PID controller," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 8687–8692, 1999.

[8] N. Marchand, S. Durand, and J. F. G. Castellanos, "A general formula for event-based stabilization of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 5, pp. 1332–1337, 2012.

[9] M. Diaz-Cacho, J. F. Pereira, E. Delgado, and A. Barreiro, "Send-on-delta sampling strategies for vehicle position tracking," in *2017 3rd International Conference on Event-Based Control, Communication and Signal Processing (EBCCSP)*. IEEE, 2017, pp. 1–7.

[10] M. Miskowicz, "Send-on-delta concept: An event-based data reporting strategy," *sensors*, vol. 6, no. 1, pp. 49–63, 2006.

[11] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[12] S.-C. Choi, J.-H. Park, and J. Kim, "A networking framework for multiple-heterogeneous unmanned vehicles in FANETs," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2019, pp. 13–15.

[13] A. Bechar and C. Vigneault, "Agricultural robots for field operations: Concepts and components," *Biosystems Engineering*, vol. 149, pp. 94–111, 2016.

[14] I. Khan and M. Spenko, "Dynamics and control of an omnidirectional unmanned ground vehicle," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4110–4115.

[15] G. A. H. Pawitan, K. Mutijarsa, and W. Adiprawita, "Three-wheeled omnidirectional robot controller implementation," in *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. IEEE, 2016, pp. 1–4.

[16] C. Ye, S. Ma, B. Li, and R. Sun, "Kinematic analysis on a mobile robot composed of three wheeled units," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2007, pp. 485–489.

[17] J. Sánchez-Santana, J. Guerrero-Castellanos, M. Villarreal-Cervantes, and S. Ramırez-Martınez, "Control distribuido y disparado por eventos para la formación de robots móviles tipo (3, 0)," in *Congreso Nacional de Control Automático*, vol. 2017, 2018.

[18] M. G. Villarreal-Cervantes, J. P. Sánchez-Santana, and J. F. Guerrero-Castellanos, "Periodic event-triggered control strategy for a (3, 0) mobile robot network," *ISA transactions*, vol. 96, pp. 490–500, 2020.

[19] J. G. Castellanos, M. V. Cervantes, J. S. Santana, and S. R. Martínez, "Seguimiento de trayectorias de un robot móvil (3, 0) mediante control acotado," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 11, no. 4, pp. 426–434, 2014.

[20] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[21] S. Vorapojpisut, M. Lhongpol, R. Amornlikitsin, and T. Phuapaiboon, "A Robot Augmented Environment Based on ROS Multi-Agent Structure," in *2019 4th International Conference on Control, Robotics and Cybernetics (CRC)*. IEEE, 2019, pp. 52–56.

[22] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[23] Z. Ma, L. Zhu, P. Wang, and Y. Zhao, "ROS-Based Multi-Robot System Simulator," in *2019 Chinese Automation Congress (CAC)*. IEEE, 2019, pp. 4228–4232.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[25] P. Estefo, J. Simmonds, R. Robbes, and J. Fabry, "The robot operating system: package reuse and community dynamics," *Journal of Systems and Software*, vol. 151, pp. 226–242, 2019.

[26] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 6698–6703.

[27] J. Guerrero-Castellanos, A. Vega-Alonzo, S. Durand, N. Marchand, V. Gonzalez-Diaz, J. Castañeda-Camacho, and W. Guerrero-Sánchez, "Leader-Following Consensus and Formation Control of VTOL-UAVs with Event-Triggered Communications," *Sensors*, vol. 19, no. 24, pp. 1–26, 2019.

[28] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[29] L. Joseph, *Robot Operating System (ROS) for Absolute Beginners*. Springer, 2018.

[30] Ebccsp 2020, date = 2020, organization = Google drive, author = M. Longinos-Garrido, url = https://drive.google.com/folderview?id=1FFVBQijDdJ1FLfiskSfq0AmfZxBKqVAf,.

Memorias del Congreso Nacional de Control Automático ISSN: 2594-2492

Collaborative event-triggered control for omnidirectional robots with obstacle avoidance

N. A. Tolentino-Medrano* **M. Longinos-Garrido****
J. F. Guerrero-Castellanos*** **J. Castañeda-Camacho** †
and R. Álvarez-Gonzalez ‡
Benemérita Universidad Autónoma de Puebla - FCE-BUAP,
Av. San Claudio y 18 Sur, Col. San Manuel,
Puebla, Pue. México.C.P. 72570
* *nestor.tolentino16@gmail.com* ** *longinos.200918526@gmail.com*
*** *fguerrero@ece.buap.mx* †*jcastane@ece.buap.mx*
‡*ricardo.alvarez@correo.buap.mx*

Abstract
In this paper, a collaborative event-triggered control strategy for a group of mobile agents is introduced. The communications between agents are asynchronous due to the event-triggered approach proposed. In this context, the agents broadcast their state information only when a specific condition is accomplished. The elements that compose the network consist of a set of omnidirectional (3,0) mobile robots. The event-triggered control strategy is designed to perform formations while avoiding collisions among them and against identified obstacles. Simulation results performed in MATLAB/Simulink of 10 omnidirectional vehicles are presented to corroborate that this control allows agents to reach the requested formation. Finally, a Montecarlo analysis is included to ensure the repeatability of the results.

Keywords: CPS, VCPS, VANET, mobile robot (3,0), event-triggered control, collaborative control, multi-agent system, collision avoidance.

1. INTRODUCCIÓN

Los sistemas ciber-físicos (CPS) están basados en la integración de diferentes subsistemas como lo son los recursos computacionales y las comunicaciones con el propósito de adquirir variables físicas y ejercer una acción sobre ellas de acuerdo a Kim and Kumar (2012). Los sistemas ciber-físicos vehiculares (CPVS) son una sub-categoría de los CPS, los cuáles cuentan con procesos físicos que recaen en acciones como la locomoción, control, estabilidad, rendimiento, robustez y están estrechamente ligados con los vehículos autónomos como se ve en Bradley and Atkins (2015). Los CPVS pueden considerarse redes ad-hoc vehiculares (VANETs) si se comunican y colaboran con otros CPVS de la misma naturaleza segun Basagni et al. (2004).
La apertura de nuevos retos en el mercado, como lo son la industria 4.0 y las ciudades inteligentes, ha generado necesidades y prospectos para explorar en áreas de redes vehiculares entre las que figuran aplicaciones industriales, usos militares, provisión de asistencia en situaciones de emergencia como recopilación de información sobre terrenos de difícil acceso para personas y para enviar provisiones como se ve en Noussaiba and Rahal (2017), Grocholsky et al. (2006). Por tanto, surge la necesidad de trabajar con sistemas multi-agentes (MAS), que permitan a los elementos que los componen intercambiar entre ellos información de sus estados y de su dinámica para la ejecución de tareas coordinadas a través de una estrategia de control distribuida de acuerdo con Dignum (2009), Lewis et al. (2013).

Las comunicaciones en tiempo real entre VANETs usando una estrategia de control continua e incluso periódica no es totalmente conveniente debido principalmente al alto consumo de energía y recursos computacionales, así como retardos en la información, pérdida de paquetes de información, latencia de la red, etc. Por tal motivo, una estrategia de control que sea asíncrona y basada en eventos es ideal para implementar VANETs. El paradigma basado en eventos surge como una opción para reducir el uso del ancho de banda de las comunicaciones en el sistema en concordancia con Miskowicz (2015). Contrario al paradigma periódico, en el control basado en eventos la señal de control se calcula y actualiza en los actuadores solo cuando una condición específica se satisface, a esto se le denomina evento.

El interés en estudiar topologías de robots omnidireccionales surge en la necesidad de un mecanismo seguro, adaptativo y flexible, Ullrich (2015); Kraetzschmar et al.

(2014). A diferencia de los robots diferenciales tradicionales, los robots ominidireccionales son capaces de satisfacer exigencias en las industrias en cuanto a movimiento y facilidad en el diseño de estrategias de control, Shneier and Bostelman (2015) y Pawitan et al. (2016).

El presente artículo retoma las ideas puestas en Sánchez-Santana et al. (2018) y presenta una estrategia de control colaborativo y distribuido para la formación de un conjunto de robots móviles omnidireccionales. La propuesta consiste en desarrollar un algoritmo para que las comunicaciones entre agentes se realice de manera asíncrona, es decir, los agentes se comunicarán con sus vecinos (de acuerdo a la topología de la red), si y solo si, un evento ocurre. En ese sentido la propuesta de control entra en el contexto de control basado en eventos. Además, inspirándose del trabajo seminal de Olfati-Saber (2002) donde se asume que la comunicación entre agentes se lleva a cabo en tiempo continuo, la estrategia de control propuesta contempla el prevenir colisiones entre los propios robots móviles, así como obstáculos presentes en el ambiente. Debido a que se usa el modelo cinemático de los robots móviles omnidireccionales Guerrero-Castellanos et al. (2014), únicamente sus posiciones y orientación son necesarias en la estrategia de control, lo cual la hace relativamente fácil a implementar, cuestión sumamente necesaria en aplicaciones reales. Además, en este trabajo se asume que la topología de comunicación para el conjunto de vehículos es representada mediante un grafo conectado y no direccionado. Puesto que el artículo tiene la intención de mostrar la idea central, pruebas analíticas no son desarrolladas, no obstante un análisis de Montecarlo es llevado a cabo para mostrar la efectividad de la propuesta.

El resto del artículo está estructurado de la siguiente forma. La sección 2 presenta el modelo matemático del los robots móviles, un recordatorio sobre la teoría de grafos, así como definiciones importantes en el contexto de control basado en eventos. La sección 3 se dedica a explicar la estrategia de control, tanto para la comunicación entre agentes disparado por eventos, como la estrategia de evasión de obstáculos y colisiones entre vehículos. Los resultados en simulación utilizando una conjunto de diez vehículos son presentados en la sección 4. Finalmente las conclusiones y trabajos futuros son expuestas en las sección 5.

2. PRELIMINARES

2.1 Modelado matemático del robot móvil (3,0)

Como establecen Ullrich (2015) y Shneier and Bostelman (2015) el robot móvil (3,0) consiste en un vehículo holónomo que posee tres grados de libertad debido a las 3 ruedas omnidireccionales que le permiten desplazarse en cualquier dirección mediante la combinación de las velocidades de cada una de sus ruedas sin importar la orientación del robot. En la Figura 1 se muestra la representación esquemática del robot (3,0) donde se establece el sistema coordenado inercial w y el sistema coordenado móvil m en el vehículo. Sean $\dot{\eta}_w = \begin{bmatrix}\dot{x}_w & \dot{y}_w & \dot{\phi}_w\end{bmatrix}^T$ las velocidades lineales y angulares inerciales mientras $\dot{\eta}_m = \begin{bmatrix}\dot{x}_m & \dot{y}_m & \dot{\phi}_m\end{bmatrix}^T$ son las velocidades lineales y angular del sistema móvil, tomando en consideración que el robot móvil tiene una estructura rígida con ruedas no deformables y no deslizantes con un solo momento sobre el plano horizontal que solo hace un punto de contacto entre la rueda y el plano, el modelo cinemático del robot (3,0) es:

$$\begin{aligned}\dot{x}_w &= \dot{x}_m cos\phi_w - \dot{y}_m sin\phi_w \\ \dot{y}_w &= \dot{x}_m sin\phi_w + \dot{y}_m cos\phi_w \\ \dot{\phi}_w &= \dot{\phi}_m\end{aligned} \tag{1}$$

En adición, el mapeo entre las velocidades angulares de las ruedas con la velocidad angular y lineal del sistema coordenado de un robot (3,0) se obtiene por:

$$\begin{aligned}\dot{\theta}_1 r &= \tfrac{1}{2}\sqrt{3}\dot{x}_m + \tfrac{1}{2}\dot{y}_m + L\dot{\phi}_m \\ \dot{\theta}_2 r &= -\dot{y}_m + L\dot{\phi}_m \\ \dot{\theta}_3 r &= -\tfrac{1}{2}\sqrt{3}\dot{x}_m + \tfrac{1}{2}\dot{y}_m + L\dot{\phi}_m\end{aligned} \tag{2}$$

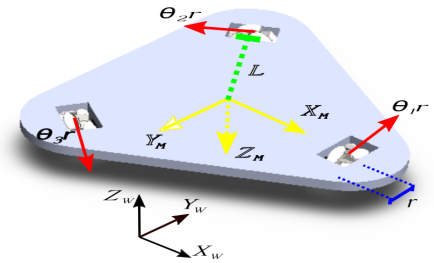

Figura 1. Ilustración del robot (3,0).

donde r es el radio de la rueda y L es la longitud desde la rueda hasta el centro del vehículo.

2.2 Teoría de grafos

El grafo $G = (V, E)$ representa el conjunto de elementos $V = \{v_1, \cdots, v_N\}$ llamados nodos y por el conjunto de bordes denotados por E. Un grafo bidireccional se dice que es uno no dirigido. La matriz de conectividad $A \in \mathbb{R}^{N\times N}$ es aquella en la cual sus elementos están dadas por $a_{ij} = 1$ si $(i, j) \in E$ y $a_{ij} = 0$ en caso contrario. La matriz D es diagonal y sus elementos d_i corresponden al conjunto de vecinos del nodo i denotados por $N_i = \{j \in V : (i, j) \in E\}$. La diferencia entra la matriz diagonal y la matriz de conectividad resulta en la matriz laplaciana $L(G) = D - A$ siendo esta semi-definida positiva para grafos no dirigidos. La suma resultante de los elementos de cada fila de L es igual a cero. En consecuencia, el vector $1 = (1, \cdots, 1)^T \in \mathbb{R}^N$ es un eigenvalor asociado con $\lambda_1(G) = 0$, Lewis et al. (2013).

2.3 Comunicación disparada por eventos

Según Guerrero-Castellanos et al. (2019), un esquema disparado por eventos permite la comunicación de manera asíncrona entre agentes al integrar una función de evento $e_i : X \times X \to \mathbb{R}$.

La función de evento para esta implementación se define por

$$e_{i,s} = \begin{cases} 1 & \text{if } |m_{i,s} - x_{i,s}| - \delta \geq 0 \\ 0 & \text{en caso contrario} \end{cases} \tag{3}$$

donde $i \in V$, $s \in \{1, 2, 3\}$, $\delta > 0$ y se encuentra compuesta por el valor actual del nodo $i, x_{i,s}$ y una memoria $m_{i,s}$ del valor de $x_{i,s}$ la ultima vez que $e_{i,s}$ resultó en un valor positivo permitiendo así que i se comunique con j solo cuando $(e_{i,s} > 0)$ y en caso contrario, que no se habilite la transmisión de información. Con base en la función de evento, un MAS cuenta con un control distribuido en donde el agente i usa sus propias variables de estado y la información obtenida por los agentes vecinos j que son almacenadas en las memorias m_i y m_j respectivamente, teniendo una forma $u = k(x_i, m_i, m_j)$.

3. DISEÑO DE UN CONTROL DISTRIBUIDO BASADO EN EVENTOS PARA ROBOTS MÓVILES (3,0)

3.1 Definición del problema

Para implementar un MAS formado por robots móviles (3,0) es necesario tomar en cuenta aspectos del tráfico de la red para prevenir la pérdida de información, retardos, desfasamiento en los datos, etc. Dado el hecho de que se tratan de robots móviles es necesario tomar en cuenta el consumo energético, los recursos computacionales y las comunicaciones. Por ello, es necesario introducir un control asíncrono que permita tomar en consideración estos aspectos por lo que, se debe de encontrar una condición que determine el momento en el que cada agente necesita transmitir su estado a sus vecinos cercanos.

Considere $N(3, 0)$ como robots móviles conectados al grafo G con $V = \{1, 2, \cdots, N\}$. Sean $x_i = (x_{i,1}\ x_{i,2}\ x_{i,3})^T \in \mathbb{R}^3$ las velocidades lineal y angular del sistema coordenado inercial $\{w\}$ y $u_i = (u_{i,1}\ u_{i,2}\ u_{i,3})^T \in \mathbb{R}^3$ las velocidades lineal y angular del sistema coordenado móvil $\{m\}$ con $i \in V$. A su vez, sean los vectores globales $\bar{x} = (x_{1,1} \cdots x_{N,1}\ x_{1,2} \cdots x_{N,2}\ x_{1,3} \cdots x_{N,3})^T \in \mathbb{R}^{N*3}$ $\bar{u} = (u_{1,1} \cdots u_{N,1}\ u_{1,2} \cdots u_{N,2}\ u_{1,3} \cdots u_{N,3})^T \in \mathbb{R}^{N*3}$ $\bar{m} = (m_{1,1} \cdots m_{N,1}\ m_{1,2} \cdots m_{N,2}\ m_{1,3} \cdots m_{N,3})^T \in \mathbb{R}^{N*3}$ donde $m_{i,s}$ representa la memoria que guardia el valor de los estados del agente $x_{i,s}$ con $i \in V$ y $s \in \{1, 2, 3\}$. Entonces el modelo para el vehículo i dado su modelo de espacio de estados es denotado por:

$$\begin{aligned} \dot{x}_{i,1} &= u_{i,1}cos(x_{i,3}) - u_{i,2}sin(x_{i,3}) \\ \dot{x}_{i,2} &= u_{i,1}sin(x_{i,3}) + u_{i,2}cos(x_{i,3}) \\ \dot{x}_{i,3} &= u_{i,3} \end{aligned} \tag{4}$$

El objetivo es integrar la función de evento $e_{i,s}$ para una función de control distribuido estático que permita a los elementos de V llegar a una referencia dada mediante el consenso de un agente de la red con un agente virtual mientras que el resto de agentes llegarán a una formación, evitando colisiones entre ellos y contra obstáculos.

3.2 Estrategia de control distribuido

Considere un MAS de N robots móviles (3,0) que interactúan de acuerdo al grafo no dirigido G. La función de control distribuido estático $u_i = k_i(x_i, m_i, m_j) \in \mathbb{R}^3$ se compone por:

$$u_i = M_i(u_i^c + u_i^\alpha + u_i^\beta) \tag{5}$$

donde u_i^c es el término de consenso, u_i^α es el término correspondiente a la evasión de colisiones entre agentes y u_i^β es el término de evasión de colisiones con obstáculos. Adicionalmente, M_i es una matriz de desacoplamiento para el vehículo móvil (3,0) dada por:

$$M_i = \begin{bmatrix} \cos(x_{i,3}) & \sin(x_{i,3}) & 0 \\ -\sin(x_{i,3}) & \cos(x_{i,3}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

3.3 Consenso y formación

Considere un sistema multiagentes de N robots (3,0) representados por (4) los cuales interactúan de acuerdo al grafo G. La función de control distribuido estático vinculada con el consenso entre agentes se define por:

$$u_i^c = C_i + \Xi_i \tag{7}$$

con C_i siendo las variables de control que están establecidas por el consenso el cual es:

$$C_i = \begin{bmatrix} -\sum_{j \in N_i}(m_{i,1} - m_{j,1}) \\ -\sum_{j \in N_i}(m_{i,2} - m_{j,2}) \\ -\sum_{j \in N_i}(m_{i,3} - m_{j,3}) \end{bmatrix} \tag{8}$$

y se tiene Ξ como las distancias que mantendrán los agentes entre ellos tal que $\Xi = \xi_{((i,s),(j,s))} \in \mathbb{R} | \xi_{((i,s),(j,s))} > 0, i, j \in V$ y $s \in 1, 2, 3$ y Ξ es una formación factible, el arreglo que representa este desfasamiento es:

$$\Xi_i = \begin{bmatrix} -\sum_{j \in N_i} \xi_{((i,1),(j,1))}) \\ -\sum_{j \in N_i} \xi_{((i,2),(j,2))}) \\ -\sum_{j \in N_i} \xi_{((i,3),(j,3))}) \end{bmatrix} \tag{9}$$

Permitiendo así poder realizar formaciones entre los N agentes con un desfasamiento entre ellos.

La función de evento que permite accionar el control distribuido del MAS, se define por:

$$e_i(x_i, m_i) = e_{i,1} \cup e_{i,2} \cup e_{i,3} \tag{10}$$

donde $e_{i,s}$ fue definida en (3).

3.4 Evasión de colisiones entre agentes

En Olfati-Saber (2006) se introduce un método para implementar un algoritmo de evasión de colisiones. Este algoritmo considera un comportamiento en "flocking" de los agentes que conforman el grupo. Cada nodo es considerado como un agente de segundo orden, usando los valores de sus posiciones para el término que previene las colisiones, mientras que las velocidades son utilizadas para el término que permite el consenso, sin embargo, en este trabajo los agentes son definidos como un sistema de primer orden y por tanto, el consenso es calculado usando también las posiciones y orientaciones de los vehículos. Los términos para la evasión de colisiones, según Olfati-Saber (2006), son u_i^α usado para evitar colisiones con sus vecinos y u_i^β que previene impactos con obstáculos en el área circundante. Ambos términos están basados en funciones potenciales atractivas/repulsivas que permiten calcular un término adicional cuando se puede dar una colisión. Los vehículos cercanos se denominan α-agentes los cuales, en consecuencia del enfoque basado en eventos, tienen un conjunto de posiciones previas $\hat{m}_j = (m_{j,1}\ m_{j,2})^T \in \mathbb{R}^2$ mientras que el conjunto de posiciones actuales del nodo son $\hat{m}_i = (m_{i,1}\ m_{i,2})^T \in \mathbb{R}^2$. Los nodos considerados como α-agentes son:

$$N_i = j \in V : \|\hat{m}_j - \hat{m}_i\| < r \tag{11}$$

donde r es el rango de interacción entre agentes. La ley de control introducida para la evasión de colisiones con α-agentes es:

$$u_{i,s}^\alpha = c_\alpha \textstyle\sum_{j \in N_i} \phi_\alpha(\|\hat{m}_j - \hat{m}_i\|_\sigma)\hat{n}_{ij} \tag{12}$$

donde la norma sigma $\|z\|$ de la matriz z es calculada por:

$$\|z\|_\sigma = \tfrac{1}{\epsilon}[\sqrt{1 + \epsilon \|z\|^2} - 1] \tag{13}$$

además $c_\alpha > 0$ y $\hat{n_{ij}}$ es un vector definido como:

$$\hat{n}_{ij} = \tfrac{\hat{m}_j - \hat{m}_i}{\sqrt{1+\epsilon\|\hat{m}_j - \hat{m}_i\|}} \tag{14}$$

con $\epsilon > 0$. El término ϕ_α es una función de acción para $z = \|\hat{m}_j - \hat{m}_i\|$ que se desvanece siempre que los α-agentes se encuentran lo suficientemente lejos el uno del otro. La función de acción se define como:

$$\phi_\alpha = \rho_h(z/r_\alpha)\phi(z - d_\alpha) \tag{15}$$

donde $\rho_h(z)$ es una función "bump" que consiste en una función escalar que varia suavemente desde 0 a 1 y se define por:

$$\rho_h(z) = \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2}[1 + cos(\pi \frac{(z-h)}{1-h})] & z \in [h, 1] \\ 0 & de\, otra\, manera \end{cases} \tag{16}$$

y $\phi(z)$ es una función sigmoidal desigual definida por:

$$\phi(z) = \tfrac{1}{2}[(a+b)\sigma_1(z+c) + (a-b)] \tag{17}$$

donde:

$$\sigma_1 = z/\sqrt{1+z^2} \tag{18}$$

con $0 < a \leq b, c = a - b/\sqrt{4ab}$ y donde $r_\alpha = \|r\|_\sigma$ y $d_\alpha = \|d\|_\sigma$ son la σ-norma del radio r y de la distancia d entre los agentes respectivamente. El valor d se obtiene con:

$$d = \|\hat{m}_j - \hat{m}_i\| \tag{19}$$

3.5 Evasión de colisiones con obstáculos

Para la evasión de obstáculos, cada vez que el vehículo i esta cerca de los obstáculos O_k un término adicional será agregado a la ley de control (9). Los obstáculos serán considerados como círculos o paredes infinitas y estos serán considerados como β-agentes en los límites del obstáculo O_k con sus estados denotados como $\hat{x}_{i,k} \in \mathbb{R}^2$ donde esta posición satisface $\hat{x}_{i,k} = argmin_{y \in O_k} \|y - \hat{x}_i\| = (x_{i,1,k}\ x_{i,2,k})^T \in \mathbb{R}^2$ con y siendo parte del área O_k. Sea $V_k = \{1', 2', \cdots, n'\}$ el conjunto de nodos que representa a los β-agentes y los β-vecinos del nodo $i \in V$ son:

$$N_i^\beta = \{k \in V_\beta : \|\hat{x}_{i,k} - \hat{m}_i\| < r'\} \tag{20}$$

donde $r' > 0$ y representa el rango de interacción entre el nodo i y los β-agentes. El término correspondiente para la evasión de colisiones con obstáculos es:

$$u_{i,s}^\beta = c_\beta \textstyle\sum_{k \in N_{i,k}} \phi_\beta(\|\hat{x}_{i,k} - \hat{m}_i\|_\sigma)\hat{n}_{i,k} \tag{21}$$

con $c_\beta > 0$ y

$$\hat{n}_{i,k} = \tfrac{\hat{x}_{i,k} - \hat{m}_i}{\sqrt{1+\epsilon\|\hat{x}_{i,k} - \hat{m}_i\|}} \tag{22}$$

La función de acción ϕ_β para el β-agentes:

$$\phi_\beta(z) = \rho_h(z/r_\beta)(\sigma_1(z - d_\beta) - 1) \tag{23}$$

donde $\sigma_1(z)$ es presentado en (17), $\rho_h(z)$ en (15), $r_\beta = \|r'\|_\sigma$ y $d_\beta = \|d'\|$ donde:

$$d' = \|\hat{x}_{i,k} - \hat{m}_i\| \tag{24}$$

4. RESULTADOS EN SIMULACIÓN

Para implementar el control propuesto, se consideran 10 nodos VANETs conectados de acuerdo al grafo de la Figura 2. Cada nodo consiste en un robot móvil (3,0)

y mantiene una distancia de 1m con respecto sus vecinos. La simulación se efectúa usando Matlab/Simulink y comienza con un conjunto aleatorio de posiciones iniciales para los 10 VANETs que corresponden a los valores del Tabla 1. Un obstáculo previamente definido con un radio de seguridad de $r_b = 0{,}6$ es introducido en el origen. Los agentes cuentan con un radio de seguridad de $r_a = 0{,}85$. Para la activación de los eventos, se tiene un umbral de error $\delta = 0{,}03$. Se introduce un líder virtual, que hace consenso únicamente con el nodo 1 para que este llegue a una referencia de $x_{i,1} = -0,5m$ y $x_{i,2} = -1{,}5m$, mientras los demás nodos mantienen su formación correspondiente respecto este punto. La simulación se ejecuta por 15 segundos con un tiempo de 0.01 segundos para cada iteración.

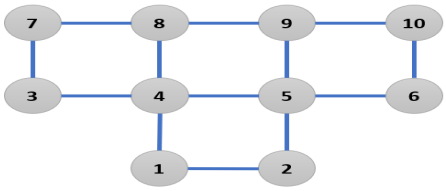

Figura 2. Grafo no dirigido de los 10 VANETs.

Tabla 1. Condiciones iniciales de las posiciones

nodo	$x_{i,1}$	$x_{i,2}$	*nodo*	$x_{i,1}$	$x_{i,2}$
1	-1.59	-1.2	6	1.14	-1.81
2	-1.37	-1.65	7	-0.45	0.43
3	-0.35	1.74	8	-1.88	0.19
4	0.24	-0.96	9	0.34	-1.62
5	-0.93	-1.19	10	0.23	0.55

En la Figura 3 y en la Figura 4 se presenta la evolución de las variables $x_{i,1}$ y $x_{i,2}$, con el transcurso del tiempo, alcanzando la formación propuesta como se muestra en la Figura 5 donde los puntos azules representan las posiciones iniciales de los 10 nodos en una gráfica XY mientras que los rojos indican sus posiciones finales y cada linea de color representa la ruta que tomo cada nodo para alcanzar su posición final. Adicionalmente, la Figura 6 compara los eventos que fueron disparados en cada nodo durante la simulación, mientras que la linea de "control continuo" representa la cantidad de veces que la comunicación sería realizada durante el proceso si la estrategia de control fuera continua y no basada en eventos, cambiando únicamente la condición de activación y calculando el control con los valores actuales de las posiciones de los agentes, contrastando así el control basado en eventos y sin eventos.

A su vez se llevó a cabo un análisis de Montecarlo de 30 simulaciones con 50 segundos de duración usando un conjunto de posiciones iniciales generadas aleatoriamente con una distribución normal uniforme en un rango de -2m a 2m para los valores tanto de $x_{i,1}$ y como de $x_{i,2}$, y se calcularon los promedios y desviaciones estándar de la norma del error, siendo esta $|e| = \sqrt{(x - x_d)^2 + (y - y_d)^2}$, obteniendo los resultados mostrados en la Figura 7.

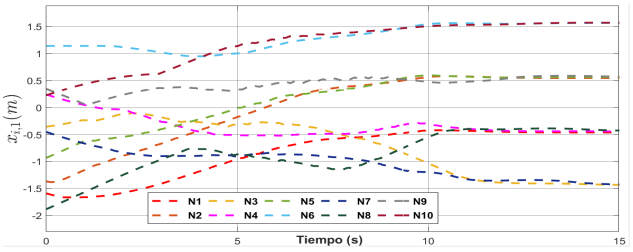

Figura 3. Posición $x_{i,1}$ de los nodos.

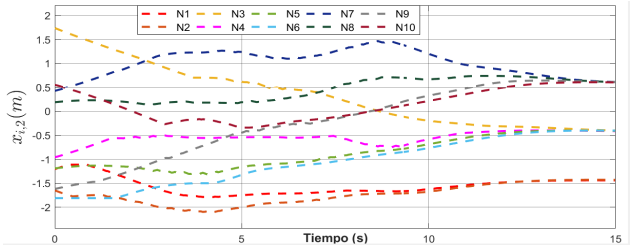

Figura 4. Posición $x_{i,2}$ de los nodos.

Finalmente se presenta en video [1] simulación de los 10 robots móviles y el obstáculo usando el modulo de Simulink VR sink para mostrar que los agentes no colisionen entre ellos ni con el obstáculo ubicado en el origen de acuerdo a los parámetros establecidos.

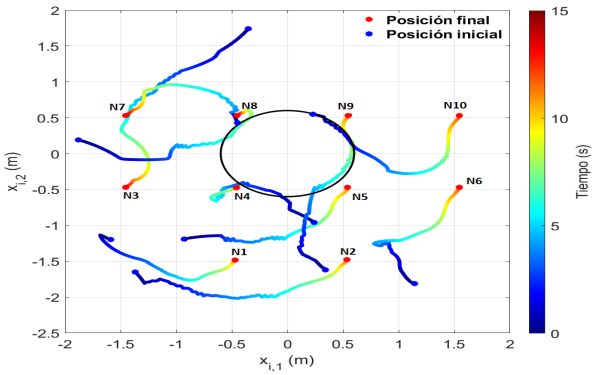

Figura 5. Trayectoria de los nodos en el plano XY.

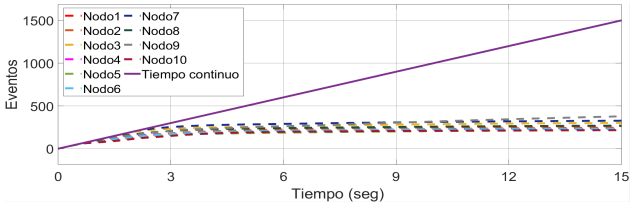

Figura 6. Eventos acumulados de cada nodo.

[1] https://drive.google.com/file/d/125674VrqSzUHXykzP6sJuRtmvIbCeaeh/view?usp=drivesdk

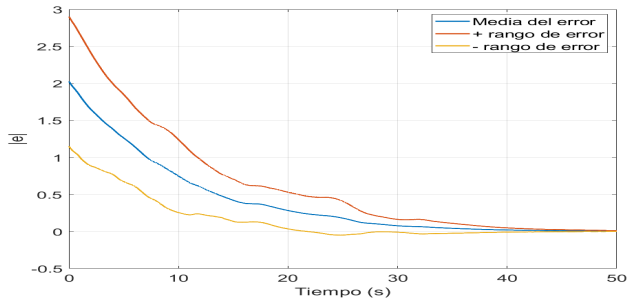

Figura 7. Promedio de la norma del error de la posición de los agentes con análisis Montecarlo.

5. CONCLUSIONES

En este artículo se presentó una estrategia de control distribuido disparado por eventos para un conjunto de robots móviles omnidireccionales. Se planteó que el control de cada agente se compone por 3 términos, el primero para que un nodo realizara consenso con agente virtual mientras que los demás nodos realizaban una formación respecto a este, el segundo para prevenir colisiones contra otros agentes y el tercero para evitar colisiones contra obstáculos identificados en el área circundante. Se llevo a cabo la simulación de este control para un grupo de 10 agentes modelados como robots (3,0) y un obstáculo previamente identificado, mostrando que los agentes lograron posicionarse en los puntos deseados para crear la formación propuesta. En el futuro se planea implementar en tiempo real la propuesta.

REFERENCIAS

Basagni, S., Conti, M., Giordano, S., and Stojmenovic, I. (2004). Mobile ad hoc networking. URL `http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10114079`. Electronic reproduction, Somerset, New Jersey : Wiley InterScience.

Bradley, J.M. and Atkins, E.M. (2015). Optimization and control of cyber-physical vehicle systems. *Sensors*, 15(9), 23020–23049.

Dignum, V. (2009). *Handbook of research on multi-agent systems: semantics and dynamics of organizational models*. Information Science Reference Hershey.

Grocholsky, B., Keller, J., Kumar, V., and Pappas, G. (2006). Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3), 16–25.

Guerrero-Castellanos, J., Vega-Alonzo, A., Durand, S., Marchand, N., Gonzalez-Diaz, V., Castañeda-Camacho, J., and Guerrero-Sánchez, W. (2019). Leader-Following Consensus and Formation Control of VTOL-UAVs with Event-Triggered Communications. *Sensors*, 19(24), 1–26.

Guerrero-Castellanos, J., Villarreal-Cervantes, M., Sánchez-Santana, J., and Ramírez-Martínez, S. (2014). Seguimiento de trayectorias de un robot móvil (3,0) mediante control acotado. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 11(4), 426–434. doi:10.1016/j.riai.2014.09.005.

Kim, K.D. and Kumar, P.R. (2012). Cyber physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100, 1287–1308. doi:10.1109/JPROC.2012.2189792.

Kraetzschmar, G.K., Hochgeschwender, N., Nowak, W., Hegger, F., Schneider, S., Dwiputra, R., Berghofer, J., and Bischoff, R. (2014). Robocup@work: Competing for the factory of the future. In R.A.C. Bianchi, H.L. Akin, S. Ramamoorthy, and K. Sugiura (eds.), *RoboCup 2014: Robot World Cup XVIII [papers from the 18th Annual RoboCup International Symposium, João Pessoa, Brazil, July 15*, volume 8992 of *Lecture Notes in Computer Science*, 171–182. Springer. doi:10.1007/978-3-319-18615-3_14.

Lewis, F.L., Zhang, H., Hengster-Movric, K., and Das, A. (2013). *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media.

Miskowicz, M. (2015). *Event-Based Control and Signal Processing*. CRC Press.

Noussaiba, M. and Rahal, R. (2017). State of the art: Vanets applications and their rfid-based systems. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, 0516–0520. IEEE.

Olfati-Saber, R. (2002). Global configuration stabilization for the vtol aircraft with strong input coupling. *IEEE Transactions on Automatic Control*, 47(11), 1949–1952. doi:10.1109/TAC.2002.804457.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51, 401–420. doi:10.1109/TAC.2005.864190.

Pawitan, G.A.H., Mutijarsa, K., and Adiprawita, W. (2016). Three-wheeled omnidirectional robot controller implementation. In *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*, 1–4. IEEE.

Sánchez-Santana, J., Guerrero-Castellanos, J., Villarreal-Cervantes, M., and Ramırez-Martınez, S. (2018). Control distribuido y disparado por eventos para la formación de robots móviles tipo (3, 0). In *Congreso Nacional de Control Automático*, volume 2017.

Shneier, M. and Bostelman, R. (2015). Literature review of mobile robots for manufacturing. doi:10.6028/nist.ir.8022.

Ullrich, G. (2015). Automated guided vehicle systems.

