Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Trabajo Terminal

Plataforma de control cooperativo centralizado para sistemas multiagente conformado por robots móviles omnidireccionales

---

que para obtener el título de:

Ingeniero en Mecatrónica

Presentan los alumnos:

García Mercado Faustino
López Sayeg Fuad Alejandro
Ugalde Sánchez Jaime Zuriel

Asesores:

Villarreal Cervantes Miguel Gabriel
Cuervo Pinto Victor Darío

9 de junio de 2022

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Plataforma de control cooperativo centralizado para sistemas multiagente conformado por robots móviles omnidireccionales

Presentan:

García Mercado Faustino

López Sayeg Fuad Alejandro

Ugalde Sánchez Jaime Zuriel

Presidente del Jurado:

Dr. Alberto Luviano Juárez

Profesor Titular:

Dr. Leonel Germán Corona Ramírez

Asesores:

Dr. Villarreal Cervantes Miguel Gabriel

Dr. Cuervo Pinto Victor Darío

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad el que suscribe García Mercado Faustino
(se anexa copia simple de identificación oficial), manifiesto ser autor (a) y titular de los derechos morales y patrimoniales de la obra titulada Plataforma de control cooperativo centralizado para sistemas multiagente conformado por robots móviles omnidireccionales.,
en adelante "La Tesis" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo a el Instituto Politécnico Nacional, en adelante El IPN, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales, Plataforma de la Dirección de Bibliotecas del IPN y/o consulta directa en la Coordinación de Biblioteca de la UPIITA "La Tesis" por un periodo de 5 años contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a "El IPN" de su terminación.

En virtud de lo anterior, "El IPN" deberá reconocer en todo momento mi calidad de autor de "La Tesis".

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de "La Tesis", manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de "La Tesis", por lo que deslindo de toda responsabilidad a El IPN en caso de que el contenido de "La Tesis" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México, a 22 de Septiembre de 2021.

Atentamente

García Mercado Faustino

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad el que suscribe Fuad Alejandro López Sayeg (se anexa copia simple de identificación oficial), manifiesto ser autor (a) y titular de los derechos morales y patrimoniales de la obra titulada Plataforma de control cooperativo centralizado para sistemas multiagente conformado por robots móviles omnidireccionales,

en adelante "La Tesis" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo a el Instituto Politécnico Nacional, en adelante El IPN, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales, Plataforma de la Dirección de Bibliotecas del IPN y/o consulta directa en la Coordinación de Biblioteca de la UPIITA "La Tesis" por un periodo de 5 años contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a "El IPN" de su terminación.

En virtud de lo anterior, "El IPN" deberá reconocer en todo momento mi calidad de autor de "La Tesis".

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de "La Tesis", manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de "La Tesis", por lo que deslindo de toda responsabilidad a El IPN en caso de que el contenido de "La Tesis" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México., a 22 de Septiembre de 2021

Atentamente

Fuad Alejandro López Sayeg

Autorización de uso de obra

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad el que suscribe Ugalde Sánchez Jaime Zuriel (se anexa copia simple de identificación oficial), manifiesto ser autor (a) y titular de los derechos morales y patrimoniales de la obra titulada Plataforma de control cooperativo centralizado para sistemas multiagente conformado por robots móviles omnidireccionales, en adelante "La Tesis" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo a el Instituto Politécnico Nacional, en adelante El IPN, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales, Plataforma de la Dirección de Bibliotecas del IPN y/o consulta directa en la Coordinación de Biblioteca de la UPIITA "La Tesis" por un periodo de 5 *años* contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a "El IPN" de su terminación.

En virtud de lo anterior, "El IPN" deberá reconocer en todo momento mi calidad de autor de "La Tesis".

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de "La Tesis", manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto de "La Tesis", por lo que deslindo de toda responsabilidad a El IPN en caso de que el contenido de "La Tesis" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México, a 21 de Septiembre de 2021

Atentamente

Ugalde Sanchez Jaime Zuriel

Agradecimientos

Se agradece al Instituto Politécnico Nacional (IPN) y a la Secretaria de Investigación y Posgrado del IPN por el financiamiento del desarrollo tecnológico en la plataforma experimental, a través de los proyectos: SIP20200150 y SIP20210374.

A la Unidad Profesional Interdisciplinaria en Ingeniería y tecnologías Avanzadas (UPIITA) y al programa académico de ingeniería en Mecatrónica, especialmente a nuestros profesores, por su incansable tarea de transmitirnos sus conocimientos para nuestro crecimiento profesional.

A nuestro asesores por guiarnos y enriquecer con sus conocimientos y sugerencias el desarrollo del proyecto, especialmente se agradece al Dr. Miguel Gabriel Villarreal Cervantes, por su acertada orientación y la confianza que depositó en nosotros, sin duda alguna, sus enseñanzas dejaron plasmado en cada uno de nosotros un sentimiento de constancia y responsabilidad que nos hizo crecer personal y profesionalmente, definitivamente merece nuestra admiración, respeto y aprecio.

García Mercado Faustino
López Sayeg Fuad Alejandro
Ugalde Sánchez Jaime Zuriel

A mis padres, por ser mi principal fuente de motivación y porque la formación que con amor y paciencia me han otorgado es la base de la perseverancia y pasión con la que hoy proyecto mi crecimiento profesional.

A mi hermana Karina y mi hermano Fernando por todo el apoyo y amor incondicional que me brindaron, alegraron mi vida en los momentos mas difíciles.

A mis amigos y amigas, especialmente a Roberto, Jennifer y Nykté, por todos sus consejos, el tiempo que compartimos y por motivarme a ser mi mejor versión en todo momento.

García Mercado Faustino

En primer lugar a mis padres Bety y Alex, quienes han dado su máximo esfuerzo para apoyarme a lo largo de este transcurso. Los valores y formación que me aportaron desde chico son la razón por la que me encuentro terminando esta etapa de mi vida. Les estoy eternamente agradecido.

A mi abuelita Lilia, por ser tan buena compañía durante mi estancia en la CDMX. Por esas horas de desvelo en las que me esperaba a que llegara a casa después de largas horas de estudio, proyectos y prácticas.

A mis hermanas Ana Gaby y Hanne Faride, por haber sido mi soporte emocional, ellas son y siempre serán mi motivación para seguir adelante.

López Sayeg Fuad Alejandro

A mis padres, Jaime y Elizabeth por sus inagotables apoyo, esfuerzo y paciencia que me han brindado en los últimos años, sin ellos, este logro no hubiera sido posible.

A mis seres queridos, especialmente a Vero, Enrique y Marce, quienes desde pequeño me han brindado su apoyo y consejos a lo largo de este camino.

A mis asesores, el Doctor Darío y el Doctor Villarreal por todo su apoyo y sugerencias para este desarrollo del proyecto.

Ugalde Sánchez Jaime Zuriel

Resumen

En años recientes la tecnología de los sistemas robóticos multiagente ha generado importantes aportaciones en la resolución de problemas que requieren de un consumo mínimo de energía, en particular, al incrementar las tareas colaborativas entre sistemas mecatrónicos autónomos para crear en conjunto un sistema más robusto, pero manteniendo su flexibilidad. Actualmente las aplicaciones de estos sistemas van en aumento, desde el área de servicios y seguridad, hasta el sector industrial; Sin embargo, el desarrollo de esta tecnología se ve limitado por distintos factores como el consumo energético y de procesamiento de datos de cada uno de los agentes, así como la comunicación eficaz entre ellos y su nivel individual de autonomía.

A través de la implementación de un sistema de control centralizado, en un sistema de cómputo principal, se puede disponer de una mayor capacidad de procesamiento de información, que analice los datos recibidos sobre la tarea a realizar, de esta forma se puede reducir el consumo de energía de cada uno de los agentes. Por otro lado, la aplicación de un sistema de visión artificial, como receptor de la ubicación y el estado de los agentes, sustituye la necesidad de generar una red de comunicación entre agentes, situación común en sistemas descentralizados. A partir de lo expuesto anteriormente, se propone desarrollar una plataforma experimental para realizar una formación específica con robots móviles omnidireccionales, a través de un control centralizado, que por medio de una interfaz gráfica presente los datos adquiridos en cada experimento y facilite el estudio del consenso de sistemas multiagentes.

Abstract

In recent years, the technology of multi-agent robotic systems has generated important contributions in solving problems that require minimal energy consumption, in particular, by increasing collaborative tasks between autonomous mechatronic systems to jointly create a more robust system while maintaining its exibility. Currently the applications of these systems are increasing, from the area of services and security, to the industrial sector; However, the development of this technology is limited by different factors such as the energy and data processing consumption of each of the agents, as well as the effective communication between them and their individual level of autonomy.

Through the implementation of a centralized control system, in a main computer system, it is possible to have a greater capacity for information processing, which analyzes the data received on the task to be carried out, thus reducing consumption of energy of each of the agents. On the other hand, the application of an artificial vision system, as a receiver of the location and state of the agents, replaces the need to generate a communication network between agents, a common situation in decentralized systems. Based on the aforementioned, it is proposed to develop an experimental platform to carry out specific training with omnidirectional mobilerobots, through a centralized control, which through a graphical interface presents the data acquired in each experiment and facilitates the study of consensus of multi-agent systems.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

A lo largo de la historia, la colaboración entre individuos de una comunidad para lograr una meta común ha sido crucial para la supervivencia de sus integrantes, desde una manada de lobos o un cardumen hasta un grupo de personas. El éxito de una comunidad al realizar una tarea compleja se debe, principalmente, a la capacidad de segmentarla en otras más sencillas, en las que los individuos actúan de forma independiente haciendo uso de sus habilidades únicas para lograr una meta en común.

Este tipo de comportamiento ha despertado un especial interés en aplicaciones orientadas a sistemas mecatrónicos, porque, un sistema capaz de realizar una tarea compleja, compuesta de diferentes funciones debe ser lo suficientemente flexible, robusto y adaptable; y debido a que, en la mayoría de los casos no resulta ni económico, ni viable, surgieron los sistemas robóticos multiagente, dando un nuevo enfoque para la solución de problemas complejos.

Por ejemplo, la micro- y nanorrobótica, carentes de altas capacidades de cómputo, al trabajar en conjunto a través de la aplicación de estos sistemas multiagente, pueden realizar un trabajo complejo, tal como en [1] quien presenta múltiples microrrobots que adhieren bacterias a su estructura para manipularlas. De forma similar existen otras áreas de aplicación donde estos conceptos se pueden aplicar, tales como la sincronización de robots manipuladores montados sobre robots omnidireccionales, el control de formación de múltiples vehículos autónomos, mantenimiento de sistemas que representen un riesgo para la salud humana como plantas nucleares, entre otros.

Comúnmente los sistemas multiagente han enfrentado problemáticas similares a las mencionadas anteriormente, que implican encontrar el método más eficaz para lograr un consenso, formación y sincronización de sus integrantes. Existen ya múltiples avances en cuanto al control de este tipo de sistemas de forma descentralizada como se menciona en [2] sin embargo, un problema recurrente que requiere un tratamiento especial es la comunicación eficaz entre los agentes, razón por la cual se ha considerado el uso de estrategias de control centralizado que por medio de una computadora principal puedan lograr la coordinación de los integrantes del sistema, ya sea por medio de sensores internos o externos a los agentes.

1.1. Sistemas de control basados en red

Sistemas de control basados en red (NCS, por sus siglas en inglés: *Networked Control Systems*) es un término utilizado para referirse a los sistemas de control que utilizan un medio de comunicación compartido para la transferencia de datos entre el controlador y el actuador. [3] En otras palabras, se dice que el mismo medio de comunicación es empleado para establecer varias rutas de comunicación con los distintos componentes de una misma red. Por ello es de vital importancia reducir el consumo energético y lograr el máximo aprovechamiento con una red que intercambie la menor cantidad de información posible.

Hoy en día, la mayoría de los sistemas tecnológicos integran funciones, sensores, actuadores y sistemas de control, estos últimos, característicos por estar conformados de sistemas embebidos, se encargan de la interconexión de elementos dentro del sistema de forma que resulte capaz de controlarse por sí mismo. [4] La toma de datos, su procesamiento y control se realizan en tiempo discreto por medio de una plataforma o red virtual, y claramente serán afectados tanto por la naturaleza de los procesos físicos como por las capacidades o limitaciones del hardware.

Los sistemas multiagente son un claro ejemplo de este tipo de comunicación necesaria entre los miembros de

la red, de forma en que la coordinación entre todos lleve a un consenso común. Cabe aclarar la importancia que tienen los sistemas mecatrónicos embebidos dentro de éstos, ya que integran el software y hardware para obtener un resultado en los actuadores, que son los que interactúan directamente con el entorno.

En un sistema de control se identifica la planta física, el controlador, el sensor y el actuador, difícilmente se van a situar en un mismo lugar, por lo que requieren transmitir señales de uno a otro [9] como se puede apreciar en la figura 1.1.1 donde el periodo de muestreo está denotado por T. En los sistemas modernos suelen estar conectados a través de una red (típicamente canales de comunicación con limitación de banda digital), dando lugar a los ya mencionados NCS. En comparación con los sistemas de control de retroalimentación tradicionales, donde estos componentes están conectados habitualmente a través de cables, la introducción de medios de red de comunicación ofrece grandes ventajas, como bajo costo, requisitos de peso y potencia reducidos, instalación y mantenimiento simples y alta confiabilidad. [10]

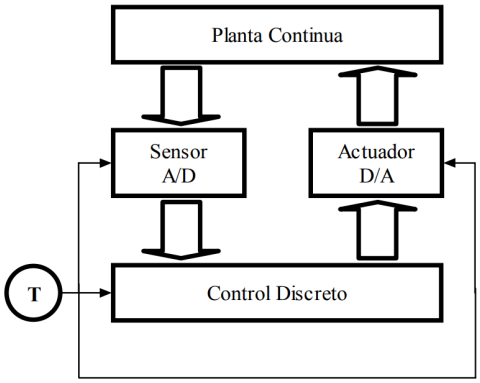


Figura 1.1.1: Diagrama de un sistema de control discreto convencional. [11]

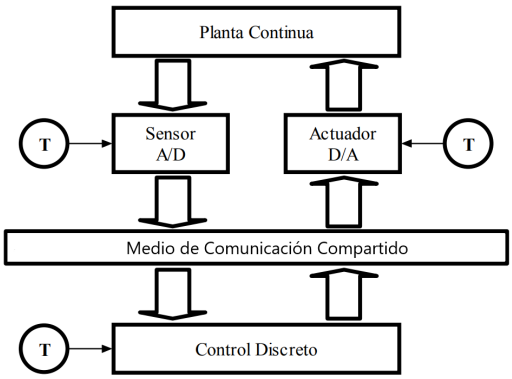


Figura 1.1.2: Diagrama de un sistema de control basado en red. [11]

Se debe tener especial atención en las trasmisiones remotas para obtener un buen intercambio de datos sin pérdida de información y que el control basado en red se pueda implementar sin inconvenientes. La comunicación entre regulador y planta queda sujeta a restricciones debidas a la existencia de un medio compartido, si se trabaja con el mismo ancho de banda existirá una pérdida de información por lo que el sistema no funciona correctamente, por consiguiente, se plantea la posibilidad de que los componentes de la red operen con diferentes tiempos de muestreo (TA, TB, TC, TS) [11] como se observa en la figura 1.1.2.

Los sistemas de control basados en red pueden clasificarse en centralizados o descentralizados [12]:

- **Centralizado:** Existe un único controlador que procesa las lecturas de los sensores y controla a los actuadores, lo cual le otorga ventajas en cuanto al diagnóstico, mantenimiento, arranque y parada de los agentes. Actualmente existe mucha investigación donde utilizan este tipo control. [13]
- **Descentralizado:** Cada componente del sistema tiene su propio controlador, este recibe información de sus vecinos, procesa la lectura de sus sensores y controla su propio sistema.

Es indispensable considerar la retroalimentación en estos sistemas de lazo cerrado, sin perder de vista el consumo energético que se busca hacer más eficiente, por lo que es necesario tener en cuenta la posibilidad de mejorar el sistema mediante la implementación de un sistema de visión artificial para el posicionamiento.

1.2. Sistemas de visión artificial para posicionamiento

El control de los SMA tiene gran cantidad de aplicaciones industriales sobre todo en los procesos de producción automatizados donde se requiere la manipulación de objetos en un espacio controlado, tal como los brazos robóticos en las plantas armadoras de automóviles. Para controlar los mecanismos se utilizan sensores de posición y modelados matemáticos [24] que dependen de la configuración geométrica del agente y algoritmos de planificación de trayectorias de acuerdo a los desplazamientos que se quieran realizar.

De lo expuesto anteriormente se concluye que es posible obtener la información de la posición de cada agente a través de un sistema de visión artificial para que cada agente alcance la posición deseada utilizando una cámara que capture imágenes continuamente mientras cada robot es controlado.

> *"La visión artificial abarca todas las aplicaciones industriales y no industriales en las que una combinación de hardware y software brinda un guiado operativo a los dispositivos en la ejecución de sus funciones de acuerdo con la captación y procesamiento de imágenes. Las tecnologías de visión integradas en robótica facilitan la justificación de los costos de la integración del sistema, brindando ganancias significativas particularmente para aplicaciones de selección y colocación, control de calidad, inspección y ensamblaje". [25]*

La visión artificial se desarrolla con base en el análisis de imágenes capturadas por medio de cámaras, en donde cada pixel es analizado para obtener características relevantes que ayuden a resolver el problema propuesto. [27] Al eliminar el contacto físico entre el sistema y las piezas que van a verificarse, la visión artificial evita daños en las piezas y elimina el tiempo y los costes de mantenimiento asociados al desgaste de los componentes mecánicos. [28]

Los principales componentes de un sistema de visión artificial (figura 1.2.1) son: la cámara, la iluminación, el centro de procesamiento y las comunicaciones.

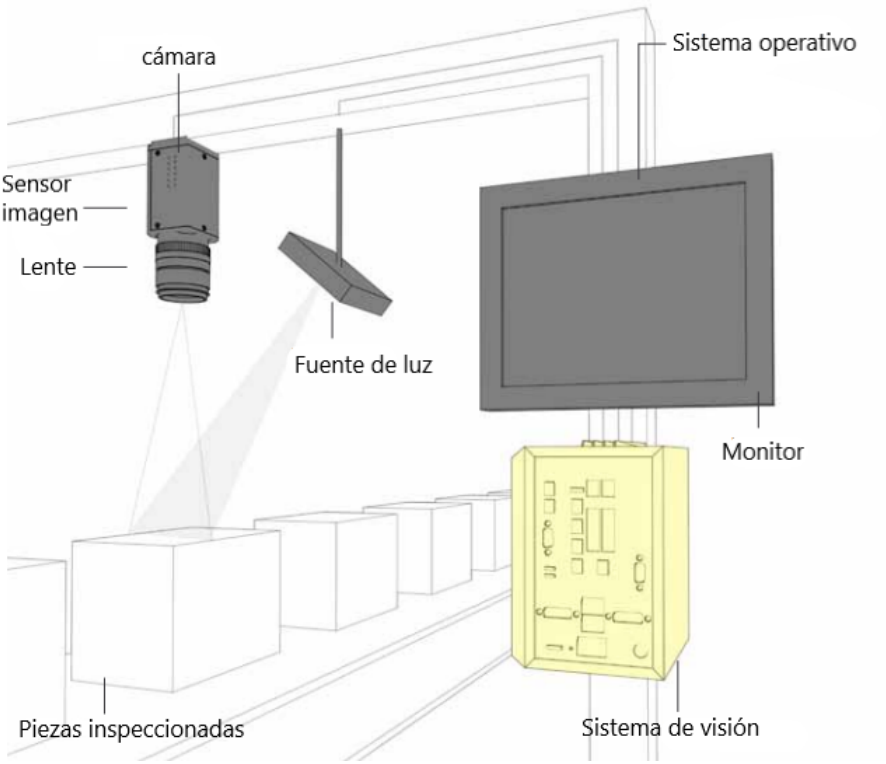


Figura 1.2.1: Principales componentes de un sistema de visión artificial

La iluminación es un aspecto clave de estos sistemas ya que proporciona la luz específica que necesitan los objetos analizados para que destaquen sus características en la imagen y esta pueda ser procesada de forma adecuada. Esta imagen digital se procesa mediante algoritmos que extraen la información precisa (posición para el caso del presente trabajo), para posteriormente ser aplicada al sistema de control y se manden las decisiones por la red de comunicación.

1.3. Importancia de la robótica móvil

En el presente siglo el desarrollo de robots ha estado fuertemente influenciado por el progreso tecnológico. Se ha pasado de máquinas que tenían como objetivo exclusivo el incremento de la potencia muscular de las personas, hasta instrumentos que son capaces de procesar información, complementando o incluso sustituyendo al hombre en algunas actividades intelectuales. [29]

A partir de las diferentes aplicaciones de la robótica en áreas como la milicia, la medicina, los servicios, la industria, etc. se fue ampliando el interés común en el estudio de nuevas soluciones que implicaran el uso de

robots. En la actualidad existen grandes avances en el área de la robótica que hacen que sea cada vez más común interactuar con diferentes tipos de robots, los cuales se clasifican según diversos criterios como:

- Generación
- Morfología
- Aplicación

Dentro de los robots industriales, uno de los sectores con mayor éxito es la industria manufacturera, de la cuál destaca el uso de robots manipuladores debido a su gran destreza y precisión para manipular objetos. En cambio, los robots móviles han mostrado interesantes aplicaciones debido a su capacidad de desplazarse en distintos terrenos y a su compatibilidad con robots manipuladores (a esta unión se le conoce como robots híbridos) dejando así una amplia gama de posibilidades al complementar la manipulación de objetos con la habilidad de trasladarse dentro de un área de trabajo.

En [30] se mencionan diferentes aplicaciones de los robots móviles en áreas como: la exploración minera, exploración planetaria, misiones militares, manejo de desechos tóxicos, automatización de procesos, vigilancia, reconocimiento de terrenos, entre otras.

Los robots móviles terrestres se pueden clasificar dependiendo de su locomoción en tres categorías: robots con patas, con orugas y con ruedas, siendo los robots con ruedas los más utilizados. Dentro de los robots con ruedas la clasificación comúnmente aceptada es la que los divide en cinco categorías de acuerdo al número y tipo de grados de libertad, los cuales a su vez se subdividen en grados de movilidad (δ_m) los cuales son todos los grados de libertad que un robot móvil puede manipular de forma inmediata por medio de cambios en la velocidad de sus ruedas, y el grado de direccionalidad (δ_s) que está relacionado con el número de llantas direccionales que puedan ser orientadas independientemente, con el propósito de dirigir al robot móvil. [2] Tomando en cuenta la notación (δ_m, δ_s) los robots móviles propulsados por ruedas se clasifican en:

- Tipo (3,0): cuenta con tres ruedas omnidireccionales (ya sean suecas o estándar) que le proporcionan tres grados de movilidad y cero grados de direccionalidad, así como total movilidad en el plano, es decir tiene la capacidad de moverse en cualquier dirección con cualquier orientación.
- Tipo (2,0): posee dos o más ruedas fijas sobre el mismo eje y ninguna rueda de direccionalidad, también se le conoce como robot diferencial y depende del cambio de velocidad en cada una de las ruedas para girar.
- Tipo (2,1): posee dos ruedas traseras para la tracción y una o más ruedas delanteras para la dirección, su aplicación más común es en los automóviles.
- Tipo (1,1): con una o más ruedas sobre el mismo eje y al menos una rueda orientable fuera del eje de las ruedas. Las ruedas traseras de este tipo de configuración no están dotadas de tracción, en este caso la rueda delantera es la que tiene acoplado un motor para la tracción y otro para la direccionalidad del robot, un ejemplo de este tipo de movimiento convencional es una motocicleta.
- Tipo (1,2): con al menos dos ruedas orientables y ninguna rueda fija, donde el grado de movilidad es la velocidad lineal y los dos grados de direccionalidad están asociados a la orientación de la rueda.

Robots móviles omnidireccionales

Entre los diferentes tipos de robots móviles existentes uno que destaca por sus ventajas en cuanto a movimiento en espacios reducidos es el tipo (3,0) también conocido como omnidireccional u holonómico, esto se debe a la capacidad que posee de desplazarse en cualquier dirección sin la necesidad de alcanzar previamente una orientación específica, lo cual le permite realizar movimientos en cualquier componente del plano así como la habilidad de girar sobre su propio eje y la facilidad con la que se desplaza en superficies planas o con cierto desnivel, los cuales son aspectos cada vez más necesarios en el ámbito industrial en el cual son usados para transportar objetos de un lugar a otro, tal como se observa en la figura 1.3.1 y 1.3.2.

Figura 1.3.1: Manipulador móvil omnidireccional comercial. [31]

Figura 1.3.2: Plataforma de carga móvil omnidireccional de uso industrial. [32]

1.4. Importancia de la arquitectura de control abierta

La arquitectura de control abierta es un tipo de estructura en la cual cada uno de sus aspectos de hardware y software pueden ser modificados sin dificultad por el usuario, tales como, sensores, interfaz gráfica de usuario, estrategias de control, entre otras, lo cual permite adaptar un sistema con mucho mayor facilidad en comparación con un sistema cerrado o de propiedad en el cual la estructura de control se encuentra oculta para el usuario y es muy difícil o imposible integrar hardware o hacer modificaciones en el sistema de control.

Actualmente, la demanda de sistemas mecatrónicos eficientes en cuanto a productividad y precio ha despertado el interés de investigadores y alumnos en mejorar el desempeño de los robots mediante el desarrollo de estrategias de control avanzadas, tales como el control de fuerza, el control adaptable y predictivo e incluso control basado en redes neuronales. Sin embargo, los avances en esta área se han quedado en muchas ocasiones como prototipos de investigación debido a que los robots comerciales presentan arquitecturas cerradas o híbridas lo cual dificulta su implementación industrial.

A pesar de que hoy en día ya existen arquitecturas de control abierta como en [33], normalmente tienen un precio elevado que dificulta su adquisición, por consiguiente, es importante el desarrollo de arquitecturas de control abiertas de bajo costo que permitan tanto a estudiantes como a investigadores implementar con facilidad diferentes estrategias de control que ayuden a la mejora de los sistemas mecatrónicos actuales.

1.5. Planteamiento del problema

Los sistemas robóticos multiagente se han propuesto recientemente como una alternativa al desarrollo de sistemas robustos, adaptables y flexibles debido a la relativa facilidad con la cual pueden adaptarse para realizar tareas más complejas de forma simultánea y coordinada. Con este tipo de tecnología se presentan distintos retos, como el diseño, desarrollo e implementación eficiente de nuevas estrategias de control cooperativo que posibiliten a los SMA (Sistemas multiagente) realizar una serie de tareas predeterminadas de forma coordinada de manera que exista un consenso.

Entre las estrategias de control cooperativo existentes, el control descentralizado ha sido estudiado en múltiples ocasiones debido a que permite a cada robot actuar como un vehículo completo que interactúa con sus vecinos, lo cual agrega flexibilidad y adaptabilidad al SMA. Sin embargo, este tipo de sistemas presenta un inconveniente en cuanto al consumo energético de cada uno de los robots, así como la velocidad y eficiencia de la comunicación entre sus agentes. En la actualidad esta problemática se ha abordado de diversas formas, una de ellas que es importante destacar es la implementación de una estrategia de control disparado por eventos para la optimización del consumo energético en sistemas multiagente de control distribuido. [2] Este enfoque dio como resultado una mejora significativa en el consumo energético. Sin embargo, deja aún un área de oportunidad en cuanto a la administración de un programa de control único (ventaja que ofrece un sistema centralizado).

Con base en lo mencionado al momento, en el presente trabajo de investigación se plantea desarrollar una plataforma de control cooperativo centralizado de arquitectura abierta que por medio de un sistema de visión artificial identifique la posición de cada uno de los agentes y determine mediante la estrategia de control la velocidad y dirección de cada uno de los robots con el objetivo de lograr una formación predeterminada.

1.6. Objetivos

Para desarrollar los objetivos, se implementó el método de árbol de objetivos de la metodología de Cross [34], dicha metodología organiza jerárquicamente los objetivos primarios, secundarios y los medios para alcanzarlos mediante un diagrama. Se realizan conjuntos de objetivos clasificándolos en niveles, donde un nivel inferior se considera el medio para alcanzar el objetivo de mayor nivel.

1.6.1. Objetivo general

Desarrollar una plataforma experimental para realizar la formación de robots móviles omnidireccionales a través de un control centralizado.

1.6.2. Objetivos específicos

1. Elaborar la plataforma de pruebas con estructura desmontable y condiciones de iluminación controladas para el sistema de visión artificial.
   - *a*) Diseñar una plataforma de pruebas desmontable.
   - *b*) Realizar el análisis estructural de la estructura.
   - *c*) Seleccionar y adaptar un sistema de iluminación para la plataforma de pruebas.
   - *d*) Validar el correcto funcionamiento de la estructura diseñada.
2. Adecuación del sistema de visión artificial.
   - *a*) Seleccionar la cámara y lente adecuadas con base en los requerimientos.
   - *b*) Instalar la cámara.
   - *c*) Crear un algoritmo de programación para detectar la posición de los agentes y su ángulo de rotación.
3. Diseñar una interfaz capaz de recibir, interpretar y presentar la información de los agentes.
   - *a*) Interpretar los datos recibidos del sistema de visión.
   - *b*) Presentar la información recibida en pantalla.
4. Incorporar un modelo físico que simule las características de los robots móviles omnidireccionales para su integración al sistema de captura de imágenes.
5. Seleccionar y acondicionar un sistema de comunicación entre la computadora central y los modelos físicos que simulan los RMO.
   - *a*) Realizar pruebas para validar la comunicación.
6. Monitorear el estado de los modelos que representan a los agentes en la interfaz.
   - *a*) Trazar la trayectoria de los agentes en la interfaz al final de su recorrido.
   - *b*) Comparar la posición mostrada en la interfaz con la obtenida mediante mediciones físicas.
7. Programar el sistema de control funcional centralizado en una computadora.
8. Implementar en simulación numérica el algoritmo de consenso para la aplicación del control de formación del SMA conformado por robots móviles omnidireccionales.
9. Realizar y validar las pruebas en simulación numérica.

1.7. Justificación

El consenso de sistemas robóticos multiagente en la actualidad tiene un amplio espectro de aplicaciones en el sector productivo y didáctico, desde usos militares, civiles, industriales y de seguridad, hasta en el área del entretenimiento por mencionar algunas. [2] La segmentación de labores en varios agentes permite obtener el mismo resultado de un sistema mono-agente robusto, con la ventaja de no depender únicamente del correcto funcionamiento de un solo sistema, además, agrega una capacidad de adaptación a nuevas tareas y la habilidad de realizar múltiples labores al mismo tiempo.

La mejora continua en estrategias robustas de control cooperativo y el desarrollo en cuanto a sistemas colaborativos permite el ingenio de nuevos conocimientos que resuelvan problemáticas recurrentes en esta área de estudio, tales como la eficiencia en el uso de recursos computacionales y energéticos, así como las limitaciones en cuanto a los canales de comunicación. El proyecto que será desarrollado está pensado para el uso de estudiantes de ingeniería y posgrado que, al estudiar este tipo de sistemas puedan interactuar con los agentes enviando y recibiendo información sobre su posición y estado en cada momento a través de una interfaz programable y de fácil uso, de esta forma la implementación y validación de la teoría respecto a estrategias de control colaborativo se facilita y permite el futuro desarrollo de esta área.

El uso de módulos inalámbricos en los agentes es indispensable para beneficiar el libre desplazamiento durante la experimentación, ya que si se trabajara de forma alámbrica podría existir algún enredo entre los cables que podrían afectar el desplazamiento de los robots o la longitud del cable limitaría el alcance de los agentes. Por ello, se propone utilizar módulos de comunicación inalámbricos que se comuniquen con facilidad con el sistema central, esta propuesta presenta un reto debido a que, en cuanto a velocidad y eficiencia de transmisión de datos la comunicación inalámbrica se encuentra en desventaja ante la comunicación alámbrica. Una estrategia propuesta para afrontar dicho reto es la implementación de un sistema de visión artificial para la localización de los agentes, de esta forma se estaría evitando depender de la velocidad de transmisión de la información de los robots hacia la computadora central.

La mecatrónica es la integración sinérgica de la ingeniería mecánica, electrónica y de control inteligente en el diseño y manufactura de productos y procesos. [35] En este proyecto se integrarán un sistema de visión artificial en conjunto con una interfaz programable en la cual se podrán observar la posición de cada uno de los agentes provenientes del sistema de odometría y visión, a su vez, se podrá visualizar la trayectoria y el tiempo efectuados en la tarea asignada a cada robot. Un sistema de control cooperativo será utilizado para lograr el consenso de los agentes en la tarea designada, se aplicará mecánica en el diseño de la estructura desmontable del área de pruebas de tal forma que soporte los esfuerzos generados por el sistema.

La integración de estas disciplinas con la finalidad de realizar un avance tecnológico para la mejora de procesos y solución de necesidades del entorno social corresponde con el perfil de egreso del Ingeniero en Mecatrónica de la UPIITA-IPN. [35]

1.8. Estado del arte

La revolución robótica está detonando un importante avance científico y tecnológico en diversas áreas de la mecánica, control, electrónica y computación; pero además ayuda a resolver problemas sociales, de salud y seguridad. [36][37] El desarrollo de la robótica móvil específicamente en el área de los sistemas multiagente ha tenido gran impacto en los últimos años y numerosos estudios han tenido como tema central el consenso de los mismos [38], dichos estudios plantean que es posible que un grupo de agentes se reúnan en un lugar o converjan en una ubicación utilizando solo la entrada de control de difusión; es decir, enviar la misma secuencia de instrucciones simples y exactamente idénticas a cada uno de los agentes. La implementación de los sistemas multiagente permite una amplia variedad de funciones como realizar tareas que requieran la sincronización de diferentes actividades, hacer una tarea en menor tiempo, o capturar múltiples muestras de forma simultánea. [39]

Un gran problema que se presenta en los sistemas multiagente es el consumo energético. En [2] se explica que una ley de control considerada de tiempo continuo supone que las señales de control se computaricen, actualicen y apliquen al sistema en un muestreo de intervalo suficientemente pequeño y constante como para no generar inestabilidad. Esta actualización constante de la señal de control implica un mayor consumo de energía y de recursos computacionales. Por lo que múltiples autores [2][40] han propuesto que la solución es un control centralizado activado por eventos basado en el intercambio de información asíncrono entre sensores, actuadores y el controlador.

Otros proyectos [41][42][44][37] se han desarrollado con un control descentralizado que se basa en la transmisión de información entre agentes vecinos de una misma red. El intercambio de comunicación se puede reducir ya que no se requiere comunicación continua y la transmisión de información solo ocurre en los instantes del evento. Una de las soluciones para que los agentes logren el seguimiento de la formación [41][45], es desarrollar un algoritmo para obtener matrices de ganancia donde se proporcionen restricciones para cumplir la tarea en el

tiempo variable esperado.

En [46] se desarrolla una plataforma experimental con un sistema de control de arquitectura abierta implementando un sistema de posicionamiento relativo basado en odometría que hace que la etapa de experimentación sea más rápida, eficiente y de menor costo comparada con otros sistemas de arquitectura de control. En [47] se realiza una evaluación de rendimiento para un robot móvil utilizando un sistema de localización absoluta basada en el mismo principio y demuestra con sus resultados que se puede tener gran fiabilidad en este tipo de sistemas.

En [30] se propone un control periódico activado por eventos o PETC (por sus siglas en inglés: *Periodic Event-Triggered Control*) para el consenso promedio y el problema de formación de una red no dirigida multivehículo que evalúa periódicamente para indicar si el control debe actualizarse o no para el instante de muestreo posterior. Para esto, se implementa el control con un robot móvil real y otros tres virtuales. De forma similar al presente trabajo, utiliza un sistema centralizado.

La innovación que se espera realizar será la implementación de un sistema de captura de imágenes para obtener la posición de los agentes que será visualizada en una plataforma de arquitectura abierta. Análogamente, esto permitirá un monitoreo en tiempo continuo para el consenso de los robots móviles. El control interno de cada agente será de menor complejidad al de los proyectos antes mencionados ya que no existirá comunicación entre los agentes, reduciendo costos en hardware por robot. A pesar de que existen plataformas como las Vicon [51] y Optitrack [52], se requiere que en México se desarrolle este tipo de plataformas de bajo costo para la investigación.

1.9. Organización del documento

El presente documento está organizado de la siguiente manera: En el capítulo 2 se abordan los temas preliminares necesarios para lograr el consenso del sistema multiagente. En el capítulo 3 se aborda el proceso diseño de los diversos sistemas que conforman a la plataforma experimental así como sus respectivos métodos de validación, posteriormente en el capítulo 4 se realiza el análisis e interpretación de los resultados obtenidos a través de los métodos de validación y finalmente, en el capítulo 5 se escriben las conclusiones y perspectivas del proyecto.

Capítulo 2

Control cooperativo centralizado para el consenso de robots móviles omnidireccionales

Con el objetivo de establecer las bases y los procedimientos apropiados para realizar el análisis teórico que permita el correcto diseño de una estrategia de control cooperativo centralizado para el consenso de robots móviles omnidireccionales, en este capítulo se abordan temas como: la teoría de grafos para distinguir la topología de comunicación de los sistemas de control basados en red y los algoritmos de consenso en el control cooperativo.

2.1. Teoría de grafos

La topología de comunicación de un sistema multiagente puede ser modelada por la teoría de grafos en la cual los agentes se representan por nodos o vértices y la comunicación entre ellos mediante bordes o aristas que pueden tener dirección. [53]

Definiciones básicas de los grafos

Definición 2.1.1. *Un grafo G, es un par de conjuntos G = (V,E) donde V es un conjunto no vacío de vértices* $(v_1, v_2, v_3, ..., v_n)$ *también conocidos como nodos, y el conjunto de aristas finitas E, también conocidos como bordes* (e_{ij})*, que unen pares de vértices y se les pueden asignar pesos* (w_{ij})*. Si E es un conjunto de pares ordenados de los elementos de V , diremos que G es un grafo dirigido o digrafo y si dos de sus vértices cualquiera pueden ser unidos por una trayectoria diremos que el digrafo está fuertemente conectado.* [2],[14],[15]

Unas de las principales características de un grafo son su orden y el grado de sus vértices. El orden de un grafo está definido por el número de vértices que posea mientras que el grado de un vértice es el número de aristas o bordes que están conectadas a dicho nodo indicado como gra(v). En caso que el grafo sea dirigido el grado de entrada de un vértice $gra^{-}(v)$ se define como el número de aristas que entran al vértice, por otro lado el grado de salida $gra^{+}(v)$ depende del número de aristas que el vértice posea como cola (aquellos que salen del vértice).

En la figura 2.1.1 se representa un grafo dirigido de orden dos, con vértices v_i y v_j, si se toma a v_i como cola y a v_j como cabeza la arista que se forma es: $e_{ij} = (v_i, vj)$ en sentido contrario $e_{ji} = (v_j, vi)$, aquí se aprecia como cada vértice tiene grado uno de salida y grado uno de entrada, al tener el mismo grado de entrada que de salida en todo el grafo se considera este un grafo balanceado.

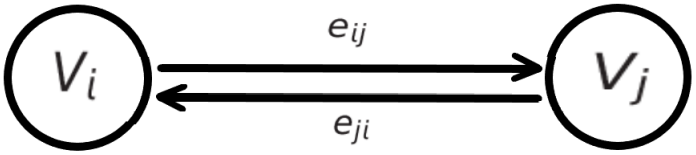


Figura 2.1.1: Características de un grafo

Definición 2.1.2. *Dado un grafo G, se dice que una arista forma un lazo cuando está asociada a dos vértices idénticos, es decir, cuando sus extremos coinciden. Dos aristas son paralelas cuando son incidentes con los mismos vértices. Cuando un grafo no posee ni aristas paralelas ni lazos se dice que se trata de un grafo simple* . [14]

Definición 2.1.3. *Un grafo no dirigido es balanceado por naturaleza ya que las relaciones entre los pares de vértices es simétrica. Un grafo regular es aquel en el que cada elemento en el grafo tiene el mismo número de vecinos.* [15]

Caminos y conceptos relacionados

Existe una necesidad de definir conceptos sobre grafos relacionados con la forma de recorrerlos, partiendo de un origen y llegando a un destino.

Definición 2.1.4. *Dado un grafo G, se denomina camino en G de* V *a* W *a una secuencia alternada de vértices* v_i *y aristas* e_i *de G.*

$$v_1, e_1, v_2, e_2, ..., v_n, e_{n+1} \tag{2.1}$$

con $v_1 = V$, $v_{n+1} = W$, *que contienen n aristas.*

$$e_i = \{v_i, v_{i+1}\},\ 1 \leq i \leq n. \tag{2.2}$$

Un camino es cerrado si $V = W$ *y* $n > 1$.

Definición 2.1.5. *Dado un grafo G, recibe el nombre de camino simple todo camino sin aristas repetidas, es decir, un camino que no pasa dos veces por la misma arista. Llamaremos camino elemental a todo camino simple sin vértices repetidos, es decir, un camino simple que no pasa dos veces por el mismo vértice.* [15]

Definición 2.1.6. *Dado un grafo G, un circuito es un camino simple cerrado. Se denominará ciclo a todo camino elemental cerrado.*

Los grafos permiten modelar y describir la interacción entre los agentes mediante distintas herramientas algebraicas computacionales como la teoría espectral del grafo, la cual consiste en asociar un grafo a su matriz de adyacencia, luego computar y estimar los valores propios de esta matriz, con los valores propios ya establecidos, se relaciona la estructura y las propiedades del grafo. Para ello es necesario representar la conectividad del grafo de forma matricial a través de la matriz de adyacencia (A), la matriz de grado (D) y la matriz Laplaciana (L).

Representación matricial de grafos

Definición 2.1.7. *Sea G = (V,E) un grafo o digrafo con* $V = n$ *y* $V = \{v_1, v_2, ..., v_n\}$. *La matriz de adyacencia A de G, respecto a los vértices anteriores, es una matriz booleana* $n \times n$, $A = [a_{ij}]$, *donde* a_{ij} *vale 1 cuando* v_i *es adyacente (unido por una arista) a* v_j *y 0 cuando no lo es.* [14]

$$a_{ij} = \begin{cases} 1 & \text{si } \{v_i, v_j\} \text{ es una arista de } G \\ 0 & \text{en caso contrario} \end{cases} \tag{2.3}$$

Definición 2.1.8. *La matriz de grado D consta de elementos D = diagonal* $[d_{out}(v_i)]$*, donde* $d_{out}(v_i) = \sum_{j=1}^{n} a_{ij}$ *(donde:* $n = |V|$*) es el grado de salida del nodo* v_i *y está dado por la suma de los elementos fila de la matriz de adyacencia.* [2]

$$d_{ij} = diagonal\{d_{out}(v_i)\} = \begin{cases} \mathrm{d}_{ij} = 0 & \text{si i} \neq j \\ \mathrm{d}_{ii} = d_{out}(v_i) & \text{si i = j} \end{cases} \tag{2.4}$$

Definición 2.1.9. *La matriz Laplaciana se denota por L(G) de tamaño* $n \times n$ *cuyas entradas están dadas por el valor de* d_{ij} *si* $i = j$ *o* $-a_{ij}$ *si* $i \neq j$ *.* [15]

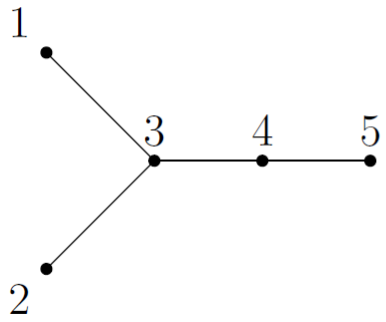
$$L = D - A := [l_{ij}] = \begin{cases} \sum_{k=1,k\neq i}^{n} a_{ik} & \text{si i = j} \\ \text{-a}_{ij} & \text{si i} \neq j \end{cases} \tag{2.5}$$

La matriz Laplaciana es una forma compacta para expresar la interconexión de comunicación entre agentes que sean miembros del SMA debido a que agrupa las características principales del grafo, proporciona información acerca del grado de cada uno de los vértices y muestra también su adyacencia. Las propiedades de la matriz Laplaciana son:

- La suma de los elementos de cada una de las filas es igual a cero.
- Sus valores propios tiene parte real no negativa.
- Cero es un valor propio y el valor propio asociado es 1^T; por lo tanto, su rango es:

$$\text{rango}(L) =\leq n - 1 \tag{2.6}$$

Un ejemplo de la representación de un grafo mediante una matriz Laplaciana se observa en la figura 2.1.2, en dicha figura se muestra un grafo simple de orden cinco no dirigido. La diagonal principal de la matriz nos dice los grados de cada uno de los vértices, al mismo tiempo se puede conocer la adyacencia entre los nodos.



$$\begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Figura 2.1.2: Ejemplo de la matriz Laplaciana de un grafo

La principal utilidad de la matriz Laplaciana en este trabajo es para evaluar algebraicamente, si un grafo está conectado o no. Esta propiedad es importante para garantizar convergencia de algoritmos de consenso.

2.2. Modelo cinemático del robot móvil omnidireccional

Considerando la posición y la orientación datos indispensables para el algoritmo de consenso, se procede a describir la forma en la que estos dos se representan matemáticamente basándose en los modelos de [66] y en los trabajos [2], [46],[47]. La representación esquemática del robot móvil (3,0) se muestra en la figura 2.2.1. Se plantea un sistema relativo de movimiento llamado $\{m_f\}$ ubicado de manera fija en el centro de masa del robot móvil. y un sistema de coordenadas absoluto fijo sobre el área de trabajo denominado $\{w_f\}$.

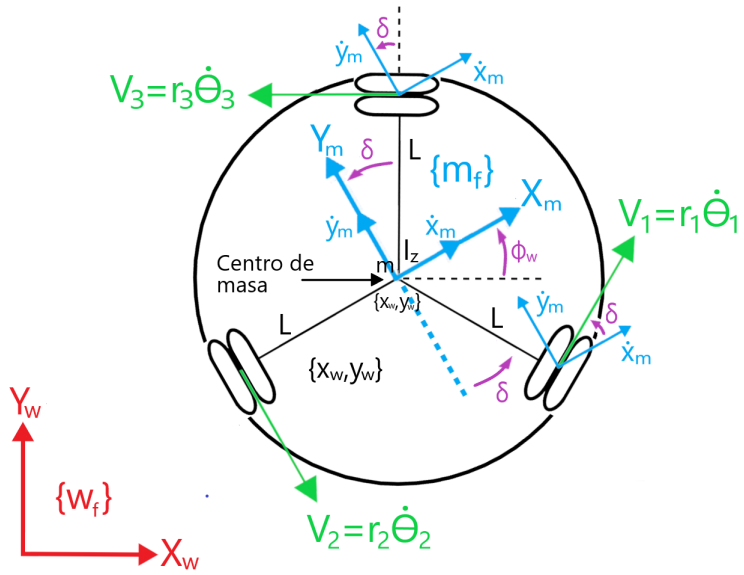


Figura 2.2.1: Diagrama del robot móvil 3.0.

Sean $\dot{\eta}_m = \left[\dot{x}_m, \dot{y}_m, \dot{\phi}_m\right]^T$ las velocidades lineales y angulares en el sistema de coordenadas de movimiento y $\dot{\eta}_w = \left[\dot{x}_w, \dot{y}_w, \dot{\phi}_w\right]^T$ las velocidades lineales y angulares en el sistema de coordenadas absoluto. El ángulo resultante (que mide la inclinación) entre un plano y el otro ha sido denominado ϕ_w. Así como se menciona en [2], se asume que no existen elementos flexibles en la estructura del robot (incluidas las ruedas), las ruedas no se deforman, no deslizan y todas tienen únicamente un punto de contacto contra el suelo, el cuál se posiciona de manera horizontal. Tomando en cuenta las consideraciones anteriores y que el ángulo descrito entre el eje Y_m y el eje axial de la rueda es de $\delta = \frac{\pi}{6}$ podemos representar el modelo cinemático de la siguiente forma:

$$\dot{x}_w = \dot{x}_m \cos(\phi_w) - \dot{y}_m \sin(\phi_w) \tag{2.7}$$
$$\dot{y}_w = \dot{x}_m \sin(\phi_w) - \dot{y}_m \cos(\phi_w) \tag{2.8}$$
$$\dot{\phi}_w = \dot{\phi}_m \tag{2.9}$$

Esta ecuación se puede representar como:

$$\dot{\eta}_w = {}^w_m R \dot{\eta}_m \tag{2.10}$$

Donde ${}^w_m R$ es la matriz de rotación.

$${}^w_m R = \begin{bmatrix} \cos(\phi_w) & -\sin(\phi_w) & 0 \\ \sin(\phi_w) & \cos(\phi_w) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Resultando:

$$\dot{\eta}_w = \begin{bmatrix} \cos(\phi_w) & -\sin(\phi_w) & 0 \\ \sin(\phi_w) & \cos(\phi_w) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\phi}_m \end{bmatrix} \tag{2.12}$$

2.2.1. Análisis cinemático del robot móvil

Podemos representar el mapeo entre la velocidad lineal de las llantas y la velocidad angular y lineal en el sistema coordenado del robot móvil (Fig. 2.2.2), para ello, primero se debe conocer la ecuación de movimiento de dos puntos en un cuerpo rígido que se representa de la siguiente manera:

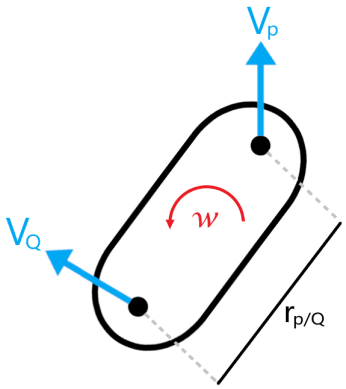


Figura 2.2.2: Representación esquemática del movimiento de dos puntos en un cuerpo rígido.

$$V_p = V_Q + w \times r_{p/Q} \tag{2.13}$$

Donde V_p, V_q son las velocidades de p y Q respectivamente, w es la velocidad angular del cuerpo rígido y $r_{p/Q}$ es la distancia de p a Q. Para obtener la relación entre las velocidades lineales de las ruedas con la velocidad lineal, se debe tomar en cuenta que $\delta = \frac{\pi}{6} = 30$, $\sin(\delta) = \frac{1}{2}, \cos(\delta) = \frac{\sqrt{3}}{2}$, resultando:

$$\dot{\theta}_1 r = \frac{1}{2}\sqrt{3}\dot{x}_m + \frac{1}{2}\dot{y}_m + L\dot{\phi}_m \tag{2.14}$$

$$\dot{\theta}_2 r = -\dot{y}_m + L\dot{\phi}_m \tag{2.15}$$

$$\dot{\theta}_3 r = -\frac{1}{2}\sqrt{3}\dot{x}_m + \frac{1}{2}\dot{y}_m + L\dot{\phi}_m \tag{2.16}$$

La ecuación resulta de la forma:

$$\dot{\theta}\bar{R} = \overline{A}^T \dot{\eta}_m \tag{2.17}$$

De tal forma que $\dot{\theta} = \left[\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3\right]^T$ es la velocidad angular de cada rueda y que $\overline{R} = [r, r, r]^T$ el radio de cada rueda siendo r = 0.0508 m y L = 0.1847 m la distancia del centro geométrico del robot móvil al centro de la rueda.

$$A^{-T} = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & L \\ 0 & -1 & L \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & L \end{bmatrix} \tag{2.18}$$

Si se desea expresar con respecto al sistema de coordenadas inercial (transformación del modelo cinemático en las coordenadas fijas), se debe considerar $\dot{\eta}_w =^w_m R^T \dot{\eta}_m$, obteniendo:

$$\overline{R}\dot{\theta} = \overline{A}^T {}^w_m R^T \dot{\eta}_m \tag{2.19}$$

2.2.2. Análisis de fuerzas y momentos

Las reacciones, resultado de las fuerzas ejercidas sobre un mismo cuerpo dependen del punto en el que estas se les apliquen, dando lugar no solo a movimientos de traslación, si no, también de rotación. A continuación, en la Fig. 2.2.3 se representan las fuerzas que puede ejercer cada llanta e interactúan en el desplazamiento que tendrá el robot.

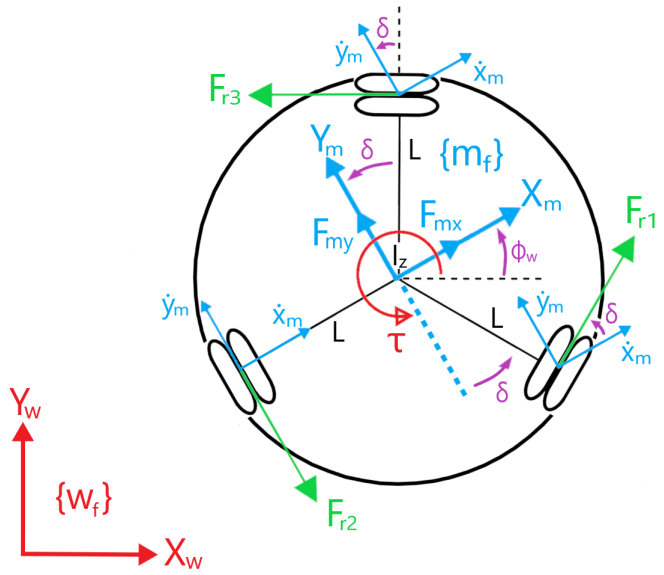


Figura 2.2.3: Diagrama de fuerzas del robot móvil.

Del diagrama de cuerpo libre se realiza el análisis de fuerzas:

$$\begin{aligned} \sum F_x &= F_{mx} \\ F_{r1}\cos(\delta) - F_{r3}\cos(\delta) &= F_{mx} \qquad (2.20) \\ \sum F_y &= F_{my} \\ F_{r1}\sin(\delta) - F_{r2}L + F_{r3}\sin(\delta) &= F_{my} \qquad (2.21) \\ \sum M_{CM} &= \tau_m \\ F_{r1}L + F_{r2}L + F_{r3}L &= \tau_m \qquad (2.22) \end{aligned}$$

Recordando que $\delta = \frac{\pi}{6} = 30° \longrightarrow \sin(\delta) = \frac{1}{2}, \cos(\delta) = \frac{\sqrt{3}}{2}$, las ecuaciones anteriores (2.20, 2.21 y 2.22), se expresan de la siguiente manera:

$$F_m = \overline{A} F_r \tag{2.23}$$

donde $F_m = [F_{mx}, F_{my}, \tau_m]^T$, $F_r = [F_{r1}, F_{r2}, F_{r3}]^T$.

y

$$A^{-T} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \\ L & L & L \end{bmatrix} \tag{2.24}$$

2.2.3. Matriz Jacobiana

La matriz Jacobiana obtiene la relación entre las fuerzas ejercidas por las ruedas del robot móvil, lo que se expresa de la siguiente manera:

$$F_r = ({}^{w}_{m}R\overline{A})^{-1}F_w \tag{2.25}$$
$$F_r = J^T F_w \tag{2.26}$$

Donde:

$$J^T = ({}^{w}_{m}R\bar{A})^{-1} = \begin{bmatrix} \frac{\sqrt{3}}{3}\cos(\phi_w) - \frac{1}{3}\sin(\phi_w) & \frac{1}{3}\cos(\phi_w) + \frac{\sqrt{3}}{3}\sin(\phi_w) & \frac{1}{3L} \\ \frac{2}{3}\sin(\phi_w) & -\frac{2}{3}\cos(\phi_w) & \frac{1}{3L} \\ -\frac{1}{3}\sin(\phi_w) - \frac{\sqrt{3}}{3}\cos(\phi_w) & \frac{1}{3}\cos(\phi_w) - \frac{\sqrt{3}}{3}\sin(\phi_w) & \frac{1}{3L} \end{bmatrix} \tag{2.27}$$

Considerando que $\tau_i = \bar{R}F_w$, la ecuación resulta:

$$\tau = J^T \bar{R} F_w \tag{2.28}$$

donde τ_i es el par ejercido por la rueda y $\bar{R}$ es una matriz diagonal que contiene el radio de las ruedas:

$$\overline{R} = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \tag{2.29}$$

2.3. Los sistemas multiagente

Un agente es un sistema computacional que intenta cumplir una serie de objetivos en un entorno dinámico y complejo. Puede detectar el entorno a través de sus sensores y actuar sobre el utilizando sus actuadores. Dependiendo de los tipos de entorno que habita, un agente puede adoptar muchas formas diferentes. Los agentes que habitan el mundo físico suelen ser robots.

Según [8] un agente presenta las siguientes propiedades:

- Autonomía
- Reactividad
- Proactividad

Un sistema multiagente entonces se considera como el conjunto de un número finito de sistemas autónomos (agentes) que de forma organizada cooperan para realizar un una actividad específica. En los últimos años, se ha mostrado un interés cada vez mayor en sistemas compuestos por varios agentes autónomos que interactúan en lugar de un solo agente. En [2] se proponen al menos tres razones para este interés en los sistemas multiagente:

1. Distinguir tareas complejas que puedan ser resultas por la naturalidad y pertinencia de involucrar de forma colaborativa a sistemas mecatrónicos del mismo tipo (agentes) para lograr un beneficio de rendimiento.
2. Establecer congruencia del SMA para realizar una tarea específica, en términos de su construcción y funcionamiento por su sencillez, costos de desarrollo, funcionalidad, flexibilidad, y tolerancia a fallo, que tener un sólo sistema mecatrónico poderoso para realizar tareas diversificadas.
3. Incorporar técnicas de comportamiento colectivo en los SMA, derivado del conocimiento de las ciencias sociales (teoría de la organización y la economía), las ciencias de la vida (la biología y la etología) y la ciencia cognitiva (la psicología, el aprendizaje, la inteligencia artificial).

Según [5][6] algunos de los rasgos más importantes que caracterizan la arquitectura grupal de un SMA son: el tipo de control, la diferenciación de los agentes y la estructura de comunicación. Por otra parte, existen dos esquemas generales para definir la colaboración de un SMA [7]: Los que se basan en el comportamiento de las especies animales y los esquemas basados en modelos que se fundamentan principalmente en conceptos de física y teoría de control. Al estudio de las condiciones que permiten que ciertas variables de un grupo de sistemas lleguen a un valor común mediante leyes de control locales se le conoce como el problema de consenso de los SMA.

2.4. Modelado de un sistema multiagente basado en robots móviles omnidireccionales

El sistema multiagente propuesto se conforma de tres (N = 3 agentes) robots móviles omnidireccionales (3,0) virtuales, donde $\eta_w = [X_w, Y_w, \phi_w]^T$ representa sus grados de libertad. Por otro lado, en la figura 2.2.1 $\dot{\theta}$ representa la velocidad angular de cada rueda del RMO. Algunas constantes de interés para definir el SMA son: $r_r = 0.0625m$ que representa el radio de la rueda y $L = 0.287m$ que es el radio del agente desde su centro de masa.

Para definir el modelo cinemático del SMA se asume que la configuración de las ruedas de cada robot móvil es rígida, que las ruedas no se pueden deslizar y que los agentes se mueven en un plano horizontal.

El vector de estados para el i-ésimo agente y su señal de control serán: $x^i = [x_1^i, x_2^i, x_3^i]^T = [x_w^i, y_w^i, \phi_w^i]^T$, $u_i = [u_1^i, u_2^i, u_3^i]^T$ respectivamente. El modelo cinemático del i-ésimo robot considerando la velocidad de las llantas está dado por las ecuaciones (2.30-2.32).

$$\dot{x}_1^i = \frac{\sqrt{3}r_r}{3}(u_1^i - u_3^i)cos(x_3^i) - \frac{r_r}{3}(u_1^i - 2u_2^i + u_3^i)sin(x_3^i) \tag{2.30}$$

$$\dot{x}_2^i = \frac{\sqrt{3}r_r}{3}(u_1^i - u_3^i)sin(x_3^i) + \frac{r_r}{3}(u_1^i - 2u_2^i + u_3^i)cos(x_3^i) \tag{2.31}$$

$$\dot{x}_3^i = \frac{r_r}{3L}(u_1^i + u_2^i + u_3^i) \tag{2.32}$$

Considerando que en el SMA propuesto existen tres agentes (N = 1,2,3 e i = 1,2,3), es necesario definir un modelo cinemático para todo el SMA independiente del número de agentes que componen el sistema. En [49] y [30] se propone que una forma compacta de representar el modelo cinemático de un SMA es:

$$\dot{X} = F(X)U \tag{2.33}$$

Donde la función suave, no lineal de elementos $F(x) \in \mathbb{R}^{3Nx3N}$ está definida por:

$$F_{i,i} = \frac{\sqrt{3}r_r}{3}cos(x_3^i) - \frac{r_r}{3}sin(x_3^i) \tag{2.34}$$

$$F_{i,N+i} = \frac{2r_r}{3}sin(x_3^i) \tag{2.35}$$

$$F_{i,2N+i} = -\frac{\sqrt{3}r_r}{3}cos(x_3^i) - \frac{r_r}{3}sin(x_3^i) \tag{2.36}$$

$$F_{N+i,i} = \frac{\sqrt{3}r_r}{3}sin(x_3^i) + \frac{r_r}{3}cos(x_3^i) \tag{2.37}$$

$$F_{N+i,N+i} = -\frac{2r_r}{3}cos(x_3^i) \tag{2.38}$$

$$F_{N+i,2N+i} = -\frac{\sqrt{3}r_r}{3}sin(x_3^i) + \frac{r_r}{3}cos(x_3^i) \tag{2.39}$$

$$F_{2N+i,i} = F_{2N+i,N+i} = F_{2N+i,2N+i} = \frac{r_r}{3L} \tag{2.40}$$

Mientras que $X = [X_1^T, X_2^T, X_3^T]^T \in \mathbb{R}^{3N}$ es un vector de estados que contiene la información de los agentes y $U = [U_1^T, U_1^T, U_1^T]^T \in \mathbb{R}^{3N}$ es la entrada de control.

2.5. Algoritmo de consenso en el control cooperativo de robots móviles omnidireccionales

En el ámbito del control cooperativo de sistemas multiagente, consenso significa que un equipo de agentes llegó a un acuerdo sobre un valor común mediante interacciones a través de una red de de comunicación compartida. Para lograr esto, deben existir reglas de interacción local entre cada agente y sus vecinos. A estas reglas que determinan el intercambio de información se les conoce como algoritmos o protocolos de consenso. Así, mientras la información se actualice constantemente, los agentes modifican sus variables de estado de acuerdo a los datos que obtengan de sus agentes vecinos.

Definición 2.5.1. *Consenso significa buscar un acuerdo respecto al valor de una variable de interés, la cual depende de los estados de todos los agentes que se encuentren en el sistema.*

Como afirma [54], para sistemas multiagente, las topologías de comunicación entre los integrantes del sistema son representadas con grafos dirigidos, debido a que el flujo de información no siempre suele ser bidireccional ya sea por las limitaciones de hardware o simplicidad del sistema. Dentro de las limitaciones de hardware, puede ser que algunos agentes únicamente estén equipados para la recepción de datos y no para su transmisión, cabe mencionar que al evitar el intercambio de información entre cada agente con los del resto del sistema se reduce el tiempo en el cual una acción de control puede ser ejecutada por todo el SMA.

Un algoritmo de consenso puede ser dinámico, permitiendo que la transmisión de información sea variable, e incluso intermitente. Lo que da una gran ventaja al implementarlo en aplicaciones reales, ya que las conexiones inalámbricas tienden a presentar ruido, ser intermitentes y poco fiables en procesos que requieran precisión y velocidad.

En la actualidad existen diversos algoritmos, que surgen como respuesta a diferentes problemáticas de consenso, dentro de los que se han distinguido por sus aplicaciones a sistemas multiagente están:

- *Rendez vous*: Se considera equivalente a un consenso de posición de un número de agentes con una topología donde la posición es inducida.
- *Flocking/swarming*: Se inspira en movimientos observados en aves, cardumenes y rebaños, por lo que la topología del sistema es dinámica y el algoritmo de control debe cumplir como regla la prevención de colisiones, relación de velocidad y centro de formación.
- Estabilización de formación: Los agentes deben mantener una forma geométrica preestablecida.
- Consenso promedio: Los estados de los agentes convergen al promedio de las Condiciones Iniciales (CI).

Específicamente, durante el desarrollo de este proyecto estaremos abordando el problema del consenso promedio de las condiciones iniciales de los RMO para posteriormente lograr la estabilización de su formación. Sin embargo, se pretende que la plataforma experimental desarrollada sirva de base para la implementación de diversos

algoritmos de control cooperativo.

2.5.1. Estrategia de control para el consenso del SMA

La estrategia de control para el consenso del sistema multiagente que se utilizó es la reportada en [49], la cual toma como base lo desarrollado en [30]. El objetivo principal al diseñar una estrategia de control para el consenso del SMA es obtener una $U = \overline{\varphi}(X)$ y una función de Lyapunov V, tales que se cumpla condición de estabilidad (2.41) para alcanzar el consenso.

$$\lim_{t\to\infty} x_j^i \;=\; x_j^k \quad y \quad \dot{V} \leq 0 \tag{2.41}$$

Realizando el cambio de variable $\Delta = X - \overline{A} \in \mathbb{R}^{3N}$ donde X representa el vector de estados que será monitoreado periódicamente y $\overline{A}$ es un vector con el promedio de las condiciones iniciales de los agentes $\overline{A} = [1\overline{a}_1, 1\overline{a}_2, 1\overline{a}_1]^T \in \mathbb{R}^{3N}$, $1 \in \mathbb{R}^N$ con $\overline{a}_j = \frac{1}{N}\sum_{i\in\nu} x_j^i(0)$ y tomando en cuenta la siguiente función V:

$$V(\Delta) \;=\; \frac{1}{2}\sum_{j=1}^{m=3}\sum_{i=1}^{N}\left(\delta_j^i\right)^2 \tag{2.42}$$

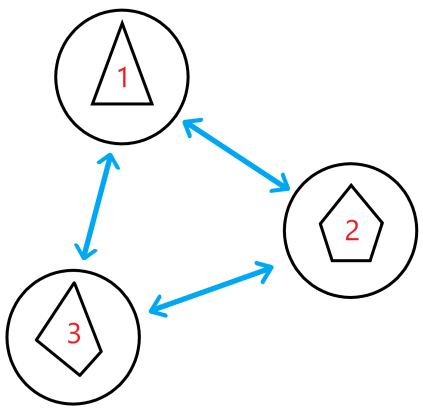
Donde $\Delta = X - \overline{A} = [\delta_1^1, \delta_1^2, \delta_1^3, \delta_2^1, \delta_2^2, \delta_2^3, \delta_3^1, \delta_3^2, \delta_3^3]^T$ (para el caso de N = 3), Entonces V es una función de Lyapunov para el consenso del SMA con el control estabilizante $U = \overline{\varphi}(\Delta) = [\overline{\varphi}_1^1, \overline{\varphi}_1^2, \overline{\varphi}_1^3, \overline{\varphi}_2^1, \overline{\varphi}_2^2, \overline{\varphi}_2^3, \overline{\varphi}_3^1, \overline{\varphi}_3^2, \overline{\varphi}_3^3]^T$ definido por:

$$\overline{\varphi}_1{}^i \;=\; \left(\frac{r_2^i + \sqrt{3}r_1^i}{2r_r}\right)cos(\delta_3^i + \overline{a}_j) + \left(\frac{\sqrt{3}r_2^i - r_1^i}{2r_r}\right)sin(\delta_3^i + \overline{a}_j) + \frac{Lr_3^i}{r_r} \tag{2.43}$$

$$\overline{\varphi}_2{}^i \;=\; \frac{Lr_3^i}{r_r} - \frac{r_2^i}{r_r}cos(\delta_3^i + \overline{a}_j) + \frac{r_1^i}{r_r}sin(\delta_3^i + \overline{a}_j) \tag{2.44}$$

$$\overline{\varphi}_3{}^i \;=\; -\left(\frac{r_1^i + \sqrt{3}r_2^i}{2r_r}\right)sin(\delta_3^i + \overline{a}_j) - \left(\frac{\sqrt{3}r_1^i - r_2^i}{2r_r}\right)cos(\delta_3^i + \overline{a}_j) + \frac{Lr_3^i}{r_r} \tag{2.45}$$

Donde los elementos del vector $r_j = [r_j^1, r_j^2, r_j^3]^T$ son expresados como: $r_j^i = -\sum_{k=1}^{N} L_{ik}\ \delta_j^k$. Cabe mencionar que aquí L_{ik} hace referencia al Laplaciano del grafo del SMA definido como se muestra en la figura 2.5.1.



$$L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Figura 2.5.1: Grafo del SMA, Laplaciano y matriz de adyacencia.

Por lo tanto, (2.43 - 2.45) estabilizan al i-ésimo agente para su j-ésimo grado de libertad en su consenso promedio:

$$\delta_j^i(t) \longrightarrow \frac{1}{N}\sum_{i\in\nu}\delta_j^i(0), \quad mientras \quad t \longrightarrow \infty \tag{2.46}$$

(2.46) Significa que el promedio del i-ésimo agente relacionado a su j-ésimo grado de libertad permanece constante a lo largo del tiempo para toda $i \in V$.

Para demostrar la estrategia de control para el consenso del SMA se consideró la función de Lyapunov V como suave, definida positiva y apropiada. Así, la derivada de V(X) resulta:

$$\dot{V}(\Delta) = \sum_{j=1}^{m=3}\sum_{i=1}^{N}\delta_j^i\dot{\delta}_j^i \tag{2.47}$$

$$\dot{V}(\Delta) = \sum_{j=1}^{m=3}\sum_{i=1}^{N}\sum_{l=1}^{N}\delta_j^i F_{i+N(j-1),l+N(j-1)}(\Delta)\overline{\varphi}_j^l(\Delta) \tag{2.48}$$

Incluyendo la ley de control (2.43 - 2.45) en (2.48):

$$\dot{V}(\Delta) = -\sum_{j=1}^{m=3}\sum_{i=1}^{N}\sum_{l=1}^{N}\delta_j^i\delta_j^l L_{ik} \tag{2.49}$$

$$\dot{V}(\Delta) = -\sum_{j=1}^{m=3}\sum_{(i,l)\,\in\,\varepsilon}(\delta_j^i - \delta_j^l)^2 \tag{2.50}$$

Como es bien conocido, para $\dot{V}(\Delta) < 0$ la energía del sistema decrece cuando $t \longrightarrow \infty$ por lo que el SMA logra el consenso para cada RMO.

2.5.2. Estrategia de control para la formación del SMA

Tomando en cuenta que la estrategia de control para el consenso hace que el j-ésimo grado de libertad del i-ésimo agente converja a el promedio cuando $X = \overline{A}$, se puede proponer una formación si es requerida a través de un cambio de variable tal como en [49] $[\Delta - \overline{\xi}]^T \in R^{3N}$ donde: $\overline{\xi} = [\overline{\xi}_1^1, ..., \overline{\xi}_1^N, \overline{\xi}_2^1, ..., \overline{\xi}_2^N, ..., \overline{\xi}_3^1, ..., \overline{\xi}_3^N]^T \in R^{3N}$ es el vector de posición relativa desde el consenso promedio.

La posición relativa se propone a partir del consenso a través de la variable acotada $L < r_n < X_{max} - \overline{A}$ que representa la distancia radial desde el consenso a la formación y de ϕ_{nf}, el ángulo de rotación respecto al eje de las abscisas en sentido anti horario. Donde $L = 0.287m$ es el radio del agente desde su centro de masa, X_{max} es el valor máximo posible los primeros dos grados de libertad dentro del área de pruebas (3x3 m).

Capítulo 3

Plataforma experimental

3.1. Sistema de posicionamiento global de los agentes con visión artificial

En éste capítulo se abordan los temas correspondientes al sistema de posicionamiento global mediante visión artificial implementado para cumplir los objetivos propuestos, así como un análisis del proceso desarrollado por el sistema para llegar a una solución.

Para todo sistema de control que incluya agentes móviles, es necesario el conocimiento de la posición y ángulo de rotación de cada uno de ellos; así como los límites del área de trabajo para llegar a un consenso. Una vez que se tiene un objetivo, es necesario el trazado de una trayectoria por lo que se requieren considerar objetos y obstáculos en caso de existir dentro del entorno de trabajo.

Con el propósito de cumplir los objetivos planteados para el sistema de visión de forma satisfactoria se implementó un algoritmo en el lenguaje de programación "Python", debido a la facilidad que posee dicho lenguaje para realizar cambios o mejoras de forma sencilla. "Python", es un lenguaje de programación que cuenta con librerías de visión artificial y proporciona una variedad de aplicaciones en esta área. [26] A continuación, se abordará el proceso realizado (figura 3.1.1) para realizar el seguimiento de los agentes en el área de pruebas.

3.1.1. Adquisición de la imagen

La base de todo sistema de visión es la obtención de la imagen a trabajar [27]; esto implica considerar algunos factores importantes que influyen en la captura satisfactoria de la imagen como: la frecuencia de captura, la resolución y la apertura de la cámara seleccionada y la iluminación del área de trabajo (tema que se aborda más adelante). Los principales componentes que determinan la calidad de la imagen adquirida son el sensor de imagen y la lente seleccionados. Por ello, se llevó a cabo un proceso de selección basado en los requerimientos esperados del sistema los cuales se muestran en la Tabla 3.1.1.

Aspectos de gran importancia para la selección de la lente y la cámara son la altura de trabajo y las condiciones a las cuales estarán expuestos dichos componentes, principalmente, debido a que la altura esta directamente relacionada con el ángulo de apertura de la lente necesaria para abarcar la totalidad del área de trabajo. Por otro lado, las condiciones de iluminación afectan la apertura del diafragma necesaria en la lente para que el sensor de imagen reciba la cantidad adecuada de luz.

Finalmente, se dio prioridad a ciertas características de la cámara sobre otras debido a las necesidades específicas del proyecto. Por ejemplo, la frecuencia de captura es más relevante que la resolución debido a que es más fácil compensar una imagen con baja resolución a tratar de solucionar un sistema de control basado en red con una retroalimentación demasiado lenta.

La imagen del área de trabajo está delimitada por la plataforma del área de pruebas, la cual por demanda mide 3m $\times$ 3m. Tomando en consideración los requerimientos presentados en la Tabla 3.1.1, se seleccionó una cámara de la marca Basler modelo acA800 - 510uc debido a que cumple con la mayoría de los requerimientos presentados, las prestaciones de dicho modelo se muestran dentro de la misma tabla a un costado de su respectivo requerimiento.

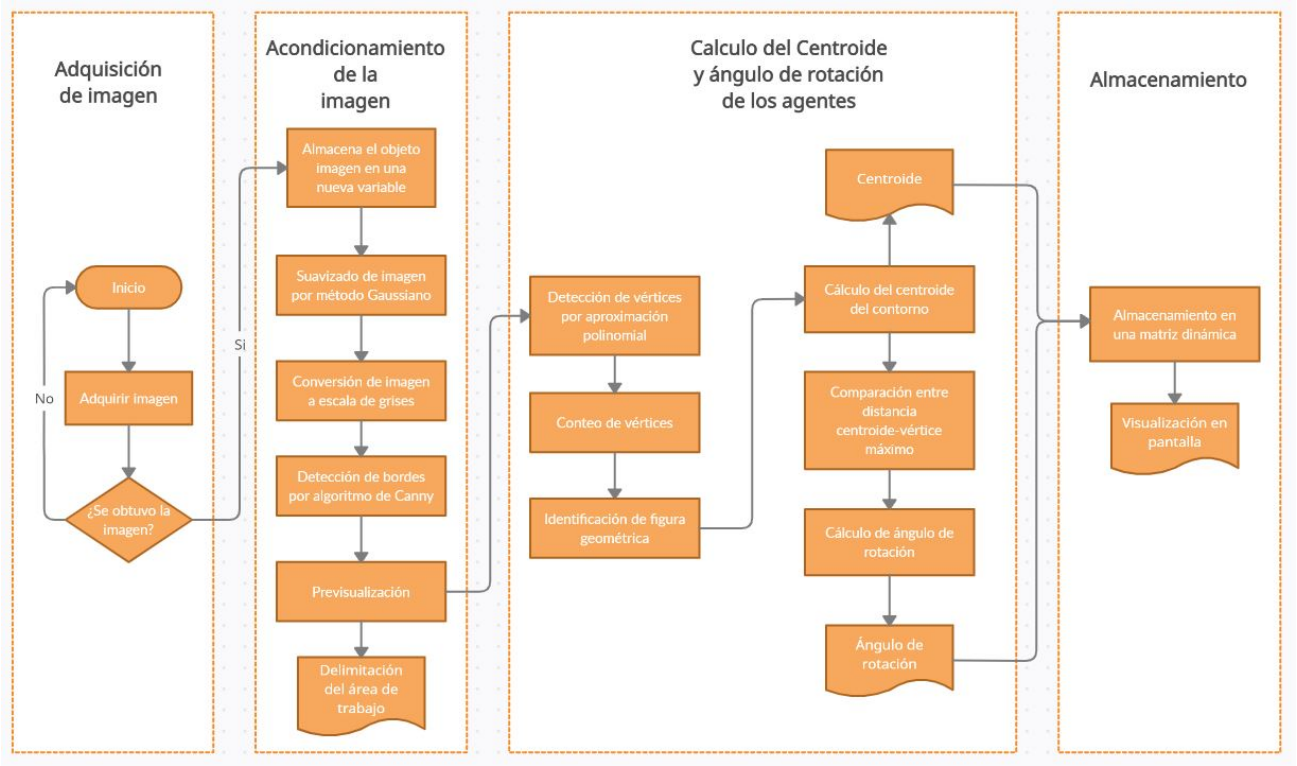


Figura 3.1.1: Diagrama del funcionamiento del sistema de visión artificial

Criterios de solicitados para la cámara vs prestaciones de la Basler acA800		
1	Debe contar con una frecuencia de captura de al menos 60 FPS. (cuadros por segundo	511 FPS
2	La resolución debe estar en concordancia con el área de trabajo (3m x 3m)	0.48 megapíxeles (600 x 800 px)
3	Utilización de un sensor con tecnología CMOS	Sensor con tecnología CMOS.
4	La cámara debe contar con una interfaz de comunicación USB 3.0 o GigE	Comunicación USB 3.0
5	Debe haber compatibilidad entre software e interfaces de operación	Interfaz amigable compatible con Windows
6	El costo total de la cámara debe ser inferior a 1000 USD	$509.45 USD

Tabla 3.1.1: Criterios vs prestaciones de la cámara

Cabe mencionar que la resolución de la cámara está compuesta por dos valores, ancho y largo. Para calcular la resolución por píxel esperada; es necesario tomar en cuanta el valor más pequeño de estos dos componentes. Recordando que el área de trabajo es de 3m $\times$ 3m, se divide la distancia del ancho de la imagen (3m aprox.) entre la resolución del ancho (600 px), dando como resultado que la resolución esperada es de 5 $\frac{\text{mm}}{\text{px}}$ aproximadamente.

Una vez que se ha seleccionado la cámara, es necesario realizar diversos cálculos para elegir correctamente una lente adecuada a las necesidades requeridas que se muestran en la Tabla 3.1.2.

La lente seleccionada (Basler C125 0418) fue aquella que cumplió con los requerimientos presentados en la Tabla 3.1.2, misma cuyas prestaciones se pueden observar del lado derecho de la Tabla. La cámara y lente seleccionada se pueden observar en la figura 3.1.2

Criterios de solicitados para la lente vs prestaciones de la lente Basler C125 0418		
1	Montura Tipo C	Montura tipo C
2	Ángulo de apertura de 80º con ±15 de libertad.	f1.8 - f22
3	Distancia focal de 3.7mm ±0.5	0 - 4 mm
4	no exceder los 1000 USD contando el costo de la cámara	$174 USD

Tabla 3.1.2: Criterios vs prestaciones del lente

La integración de la cámara al sistema de visión se realiza mediante la librería gratuita para el lenguaje de programación “Python"llamada “Pypylon", la cual permite crear un objeto de vídeo compatible con dicho lenguaje, algunos procesos de adquisición de imagen relacionado al diagrama de flujo presentado en la figura 3.1.1 se explican a continuación.

· **Inicio:** Es el inicio del programa.

· **Adquirir imagen:** Se obtiene la imagen después de haber creado un objeto *camera* y se hace una conversión de formato a BGR (del inglés, Blue Green Red) para que sea compatible con las funciones que se usarán a continuación [55].

· **¿Se obtuvo la imagen?:** En esta parte, se comprueba si se adquirió exitosamente la imagen, en caso de existir algún error se vuelve a intentar el proceso de adquisición.

Figura 3.1.2: Cámara Basler acA800 con lente Basler C125 0418

3.1.2. Acondicionamiento de la imagen

Una vez que se ha recibido la imagen con éxito, es necesario acondicionarla a través de operaciones morfológicas para volverla más manipulable y evitar errores ocasionados por ruido en las operaciones matemáticas que se van a realizar posteriormente. En otras palabras, facilitar el análisis para el proceso de detección.

Debido a que el proceso por el cual se pretende reconocer a los agentes se realiza mediante la detección de bordes, la etapa de acondicionamiento está direccionada para facilitar ésta tarea. Por ello, es importante reducir el contraste, suavizar los bordes de la imagen y eliminar pequeños detalles antes de la segmentación del objeto de interés.

El filtro Gaussiano ayuda a suavizar la imagen, es decir, a reducir la cantidad de variación de intensidad entre píxeles vecinos y conseguir que las intensidades de los objetos pequeños se mezclen con el fondo con el objetivo de detectar los objetos de mayor tamaño para limpiar la imagen de objetos no deseados o falsos bordes. Sí bien el filtro de media también es una opción más simple de implementar se prefirió el Gaussiano debido a produce un suavizado más uniforme y es separable, es decir, se puede implementar una máscara vertical y otra horizontal.

El filtro Gaussiano se comporta siguiendo la expresión:

$$G_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

A continuación se explica el proceso llevado a cabo para el acondicionamiento de la imagen de forma general tal como se aprecia en el diagrama de bloques del algoritmo 3.1.1.

- **Almacena el objeto imagen en una nueva variable:** Después de haber obtenido la imagen, se procede a almacenarla en la memoria de la computadora para su posterior uso al presentar la información en la interfaz.

- **Suavizado de imagen por método Gaussiano:** El filtro Gaussiano es un filtro digital que suaviza la imagen facilitando la detección de bordes [56]. En la figura 3.1.3 se muestra una imagen ejemplo a la que se le ha aplicado el filtro Gaussiano.

Figura 3.1.3: Muestra de una imagen a la cual se le ha aplicado un filtro Gaussiano

- **Conversión de imagen a escala de grises:** Se convierte la imagen de formato BGR a escala de grises [57] para facilitar la conversión a imagen binaria previo a la detección de bordes por el algoritmo de Canny.

- **Detección de bordes por algoritmo de Canny:** La detección de bordes por algoritmo de Canny fue desarrollada por John F. Canny en 1986 [58] como un algoritmo multi-etapas que se basa en cuatro procesos o etapas principales :

1. Reducción de ruido: Debido a que la detección de bordes es susceptible al ruido en la imagen, el primer paso es quitar dicho filtro con un filtro Gaussiano de quinto orden (previamente realizado).

2. Encontrar el gradiente de intensidad de la imagen: Después del suavizado de la imagen, se procede a filtrar con un filtro Sobel, después se realiza la primera derivada en la dirección horizontal G_x y en dirección vertical G_y para encontrar el gradiente del borde y la dirección de cada píxel.

$$\text{Gradiente\_Borde(G)} = \sqrt{G_x^2 + G_y^2} \tag{3.2}$$

$$\text{Angulo}(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{3.3}$$

3. Supresión no-máxima: Después de obtener la magnitud y dirección del gradiente, se realiza un escaneo completo de la imagen para remover píxeles innecesarios que no son parte propiamente del borde. Para lograr esto, cada píxel es revisado y si este es un máximo local en la vecindad de la dirección del gradiente, véase la figura 3.1.4.

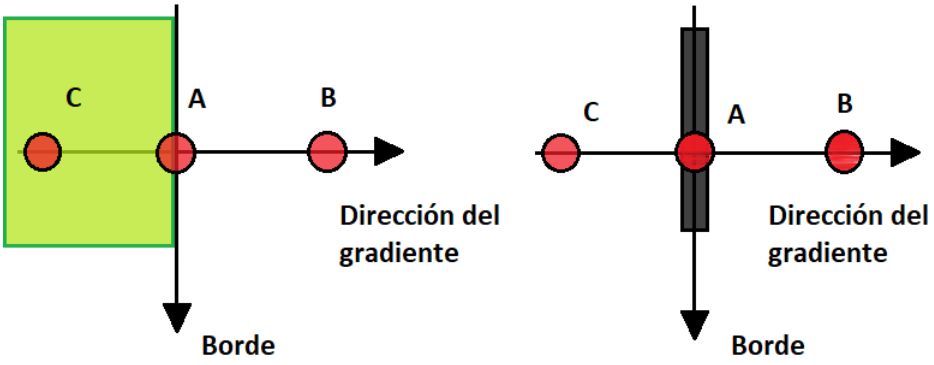


Figura 3.1.4: Supresión no-máxima en el proceso de detección de bordes por Canny

Donde el punto **A** está en el borde (en la dirección vertical), la dirección del gradiente es normal al borde y los puntos **B** y **C** están en la dirección del gradiente. Así entonces el punto **A** se analiza junto con **B** y **C** para comprobar si forma un máximo local. De ser así, es candidato para la siguiente etapa, de otra forma se elimina. En términos más sencillos, el resultado es una imagen binaria con bordes delgados.

4. Umbral de histéresis: Esta etapa hace una selección definitiva de los bordes, para esto, se proponen dos valores de histéresis $valor_{max}$ y $valor_{min}$. Cualquier borde cuya intensidad de gradiente mayor a $valor_{max}$ se considera borde definitivo y cualquier borde debajo de $valor_{min}$ se descarta. Aquellos bordes que yacen entre los dos umbrales se clasifican en bordes o candidatos a bordes dependiendo de su conectividad. Si están enlazados a píxeles con bordes definitivos, también se consideran bordes definitivos, de otra forma, se descartan. Véase la figura 3.1.5.

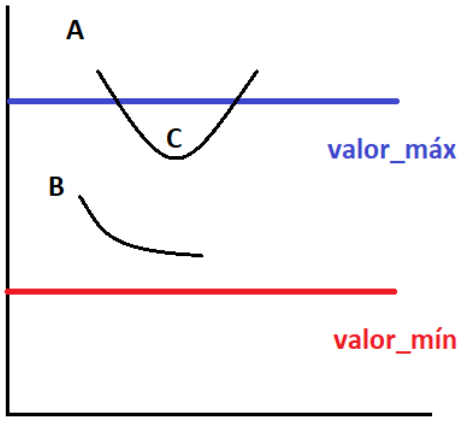


Figura 3.1.5: Umbrales de histéresis

Nótese que el borde **A** es superior a $valor_{max}$ entonces se consiera un borde definitivo. Aunque el borde **C** está debajo de $valor_{min}$, éste se encuentra conectado al borde **A**, así que se considera un borde válido, de la misma forma, aunque el borde **B** sea inferior a $valor_{min}$ y esté en la misma región que **C**, no está conectado a ningún borde definitivo, entonces se descarta.

En la figura 3.1.6, en la parte derecha, se muestra la imagen original con algunas adecuaciones necesarias para calcular algunas variables de interés y en la parte izquierda se muestra la imagen después de haber aplicado el filtro Gaussiano y el algoritmo de Canny. En este caso se han tratado los cuatro vértices del área de trabajo y los tres marcadores que corresponden a los robots omnidireccionales.

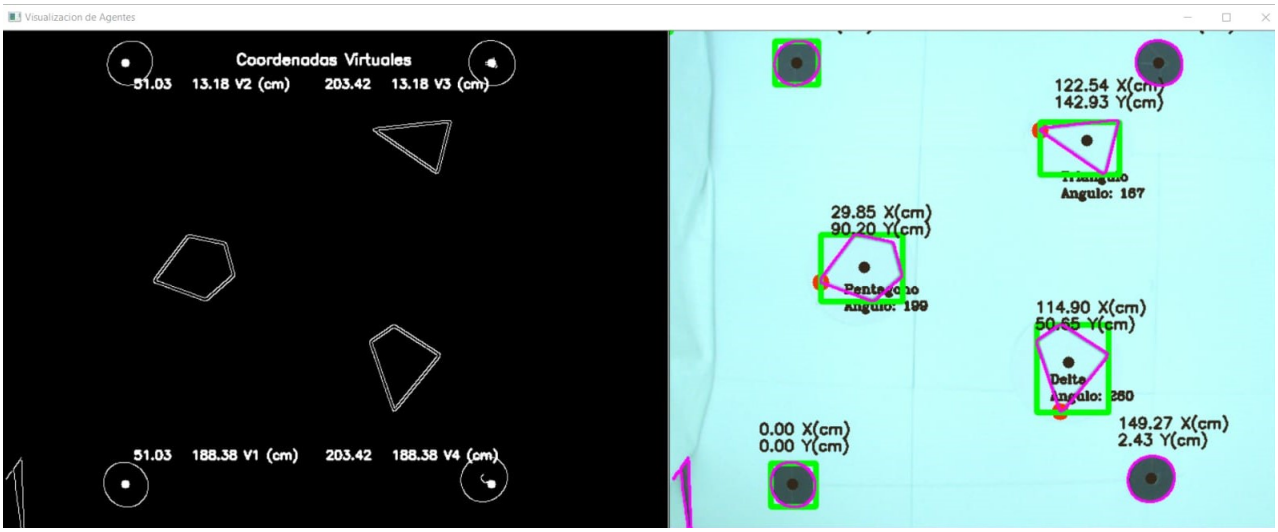


Figura 3.1.6: Interfaz del sistema de visión mostrando el proceso de reconocimiento de agentes

3.1.3. Cálculo de los centroides de los agentes

La teoría de los momentos proporciona una útil alternativa para la representación de formas de objetos. Si tenemos un objeto en una región que viene dado por los puntos en los que $f(x, y) > 0$, se define el momento de orden p,q como: [59]

$$m_{pq} = \int \int x^p y^q f(x, y)\ dxdy \qquad \text{para} \quad p, q = 0, 1, 2, ... \tag{3.4}$$

El teorema de representación de los momentos nos dice que el conjunto infinito de momentos mp, q, p, q = 0, 1,... determinan unívocamente f(x,y) y viceversa. Tendremos una imagen digital definida por la función f(x,y), donde (x,y) son las coordenadas de un punto y f(x,y), el valor de ese punto; en el caso desarrollado, este valor será 0 si el punto es distinto de negro y 1 si es negro; si se hubieran tenido en cuenta los colores en una imagen este valor dependería del color del punto. Pero al tratarse de imágenes digitales, el momento de orden (p+q) se define como:

$$M(p, q) = \sum_x \sum_y x^p y^q f(x, y) \tag{3.5}$$

Momentos de orden 0

Momentos Simples de Orden 0, M(0,0): Suma todos los píxeles cuyo valor es uno, es decir los que son distinto de blanco, por lo tanto calcula el área. El momento simple de orden 0 representa la superficie de la figura en imágenes binarias y a su vez, la superficie de imágenes en escala de grises. Es la suma de los valores de todos los píxeles. Para ello nos basamos en la fórmula de los momentos simples.

$$M(0, 0) = \sum_x \sum_y f(x, y) \tag{3.6}$$

Momentos de orden 1

Momentos Simples de Orden 1, M(1,0), M(0,1): Como hemos comentado anteriormente se usa principalmente para hallar el centro de masa de una figura, como se puede apreciar en la ecuación (6).

$$M(1,0) = \sum_x \sum_y x f(x,y) \qquad M(1,0) = \sum_x \sum_y y f(x,y) \tag{3.7}$$

Momentos Centrales de Orden 1, MC(1,0), MC(0,1): Estos momentos son 0 por definición. A grandes rasgos,se usará este fundamento matemático en la implementación de nuestro proyecto.

$$U(1,0) = \sum_x \sum_y (x-\overline{x})^1 (y-\overline{y})^0 f(x,y) = M(1,0) - \frac{M(1,0)}{M(0,0)} M(0,0) \tag{3.8}$$

$$U(0,1) = \sum_x \sum_y (x-\overline{x})^0 (y-\overline{y})^1 f(x,y) = M(0,1) - \frac{M(0,1)}{M(0,0)} M(0,0) \tag{3.9}$$

3.1.4. Cálculo de los ángulos de rotación de los agentes

Una vez implementado el algoritmo para la detección de contornos, se almacenan las coordenadas del vértice uno de referencia (x_0, y_0), las coordenadas de los centroides de cada agente x_i, y_i para $i = 1, 2, 3$ y las coordenadas que corresponden a la distancia máxima formada entre los vértices y centroide de cada agente en cuestión (x_{m_i}, y_{m_i}) para $i = 1, 2, 3$ como se puede apreciar en la figura 3.1.7

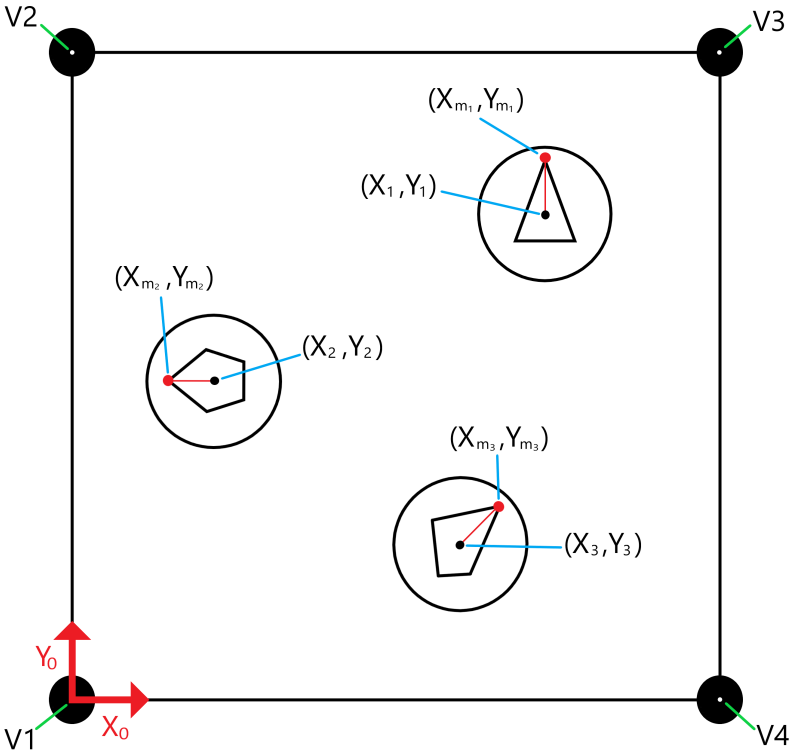


Figura 3.1.7: Disposición virtual de los agentes en un plano bidimensional

El algoritmo es capaz de detectar las coordenadas que corresponden a la distancia euclidiana máxima formada entre los vértices y centroide de cada indicador utilizando el modelo 3.10.

$$D_{\text{máxima}} = \sqrt{(\Delta y)^2 + (\Delta x)^2} \tag{3.10}$$

Para ejemplificar el algoritmo de cálculo del ángulo de rotación propuesto, a continuación se detalla un ejemplo práctico, en este caso el agente con un triángulo como identificador, como se puede apreciar en la figura 3.1.8 y 3.1.9, en este caso en particular, la distancia máxima corresponde a cinco unidades, de esta forma las coordenadas x_{m_1}, y_{m_1} (previamente explicadas) son calculadas exitosamente.

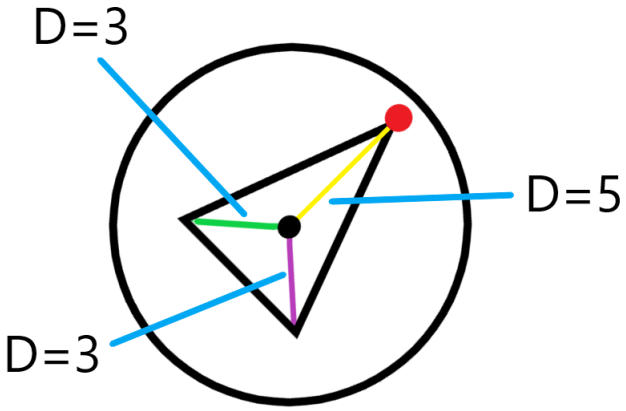


Figura 3.1.8: Caso práctico para ejemplificar el algoritmo de cálculo de ángulos de rotación

Figura 3.1.9: Modelo 3D de un agente con el indicador propuesto

Una vez obtenidas las coordenadas (x_{m_1}, y_{m1}) y (x_1, y_1) se procede a realizar el cálculo del ángulo de rotación del agente mediante la función **atan2()** [50] la cuál regresa un valor numérico entre $-\pi$ y π representando el ángulo θ de un punto (x, y) tomando como referencia un eje x positivo. El ángulo θ de rotación del agente es entonces

$$\theta = \text{atan2}\left(\frac{y_{m_1} - y_1}{x_{m_1} - x_1}\right) \tag{3.11}$$

3.1.5. Almacenamiento

Para poder utilizar la información obtenida por el sistema de visión artificial es necesario almacenar la información respecto a la posición y ángulo de rotación ϕ_m de forma conveniente para su posterior integración con el sistema de control.

Una vez que se identifica el número de vértices que posee cada indicador colocado sobre los agentes se clasifica la información obtenida de cada uno de ellos y se almacena en matrices, siendo la primera la matriz de posición, en la cual se guardan las coordenadas en las cuales se encuentra el centroide de la figura en cada periodo de tiempo y la segunda la matriz de rotación de dicho agente, en la cual se almacena el ángulo de rotación en cada periodo de tiempo tal como se muestra en la siguiente expresión.

$${}^{0}_{n}P = \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad {}^{0}_{n}R_z = \begin{bmatrix} Cos(\phi_{m_n}) & -Sin(\phi_{m_n}) & 0 \\ Sin(\phi_{m_n}) & Cos(\phi_{m_n}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

3.1.6. Marcos de referencia

Debido a que al obtener la captura el origen del sistema de coordenadas se encuentra en la esquina inferior izquierda de la imagen, es necesario establecer un sistema de coordenadas inercial, que no sea propenso a variaciones debido a movimientos de la cámara o la estructura.

Sistema de coordenadas inercial

Para establecer nuestro propio origen se implementó un sistema de indicadores en los extremos del área de trabajo para delimitar la zona deseada, el proceso consiste en una pre-visualización que se ejecuta al iniciar el programa en la cual el usuario tiene la posibilidad de ajustar los valores de los umbrales correspondientes a la detección de bordes con el propósito de lograr una calibración más precisa del sistema de detección de bordes.

Una vez reconocidos los cuatro indicadores de los extremos del área de trabajo el usuario deberá presionar una tecla preestablecida para capturar las coordenadas de los indicadores y visualizarlas en la pantalla. Dichas coordenadas tomarán el nombre de vértices y al vértice que se encuentre más cerca del origen de imagen (esquina superior izquierda de la imagen) se le asignará como el vértice dos tomando como nuevo origen del sistema de coordenadas inercial la esquina inferior izquierda de la imagen desde el cual se calculará la posición y ángulo de rotación de los agentes.

En la figura 3.1.11 se indican rojo el sistema de coordenadas absoluto, mientras que en la figura 3.1.10 se presenta un ejemplo del funcionamiento de la etapa de pre-visualización que almacena los vértices para el sistema de coordenadas inercial.

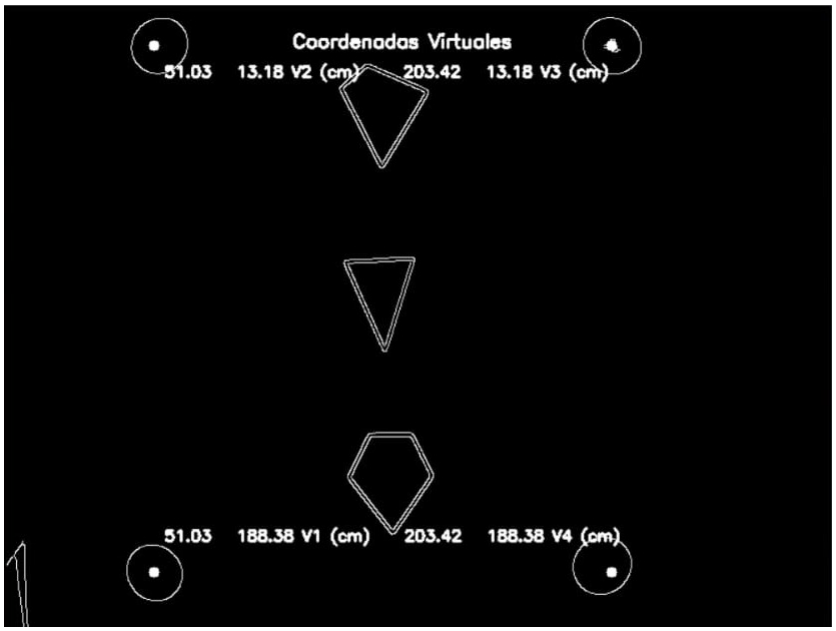


Figura 3.1.10: Etapa de pre visualización del programa

3.1.7. Método de validación y resultados del sistema

Para comprobar que el algoritmo implementado para la detección de la posición y ángulo de rotación de los agentes funciona adecuadamente se propuso realizar una serie de pruebas con representaciones de los agentes (debido a la situación en la cual se encuentra el país en la actualidad no ha sido posible contar con el espacio previsto en el CIDETEC para realizar las pruebas con los agentes reales por el momento) en las cuales se realizan diez formaciones distintas con variaciones en la posición de cada uno de los agentes y su ángulo de rotación tal como se observa en el esquema de la figura 3.1.11.

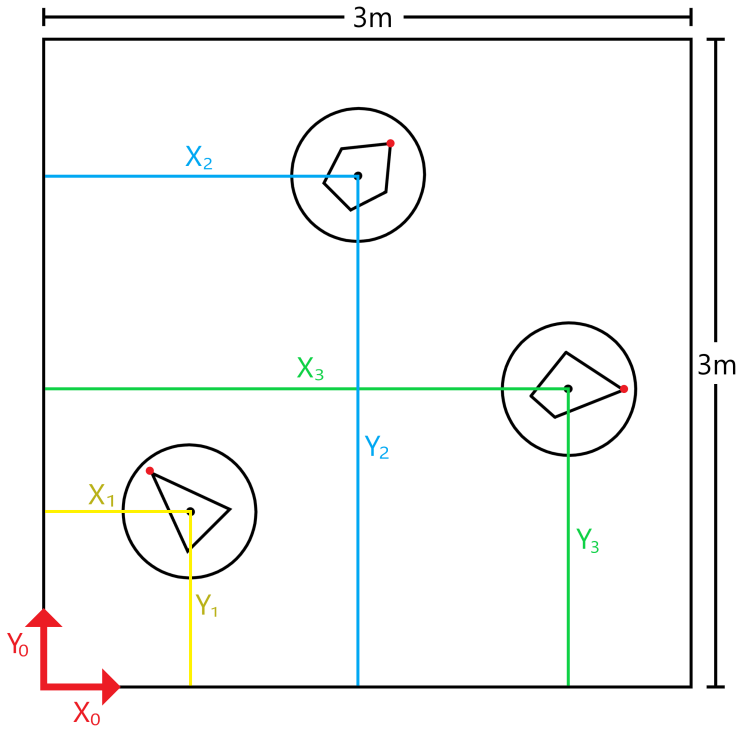


Figura 3.1.11: Esquema propuesto para las pruebas de validación del sistema de localización de agentes

En cada formación realizada el algoritmo implementado obtendrá las posiciones de cada uno de los centroides de los indicadores, así como las coordenadas de los vértices máximos (indicadores del ángulo de rotación ϕ_m con los cuales calculará y presentará en pantalla tanto la ubicación en cm de cada agente con respecto al vértice 1 (origen del sistema de coordenadas inercial) como su ángulo de rotación en radianes.

El cálculo del ángulo de rotación real se obtendrá a partir de las coordenadas medidas del vértice máximo (el que se encuentra a mayor distancia del centroide de la figura) y del origen mediante el uso de la función atan2 tal como se ve en la siguiente expresión:

$$\phi_{m_n} = \text{atan2}\left(\frac{Y_m - Y_0}{X_m - X_0}\right) \tag{3.13}$$

Donde (X_m, Y_m) son las coordenadas del vértice máximo con respecto al origen y (X_0, Y_0) son las coordenadas del centroide de la figura indicador.

Por otro lado, se tomará la distancia medida con un flexómetro desde el eje de las ordenadas y abscisas hacia el centroíde del indicador como la referencia real para la posición de los agentes, análogamente se calculará el valor máximo, mínimo y promedio para conocer así el error presente en el posicionamiento de los agentes.

Adicionalmente se propuso una segunda prueba para el sistema de visión, en la cual se realizarán dos trayectorias (tal como se aprecia en la figura 3.1.12) haciendo uso de dos de las representaciones de los agentes para probar el sistema de posicionamiento en condiciones donde los objetivos no permanecen estáticos. Para lograr este objetivo se propone el desarrollo una interfaz Gráfica de Usuario (GUI o *Graphic User Interface*, del inglés), en

la cual se pueda visualizar la trayectoria de cada uno de los agentes que haya sido detectada por el sistema de visión.

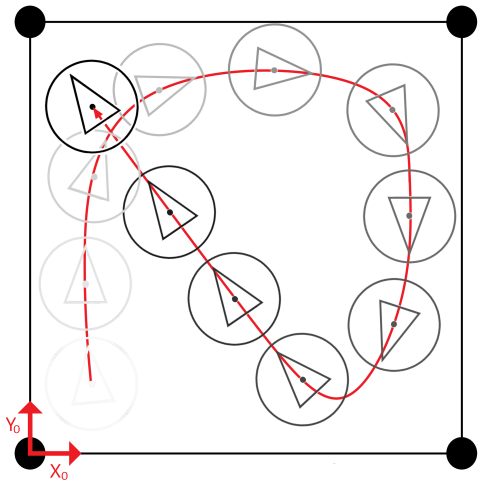


Figura 3.1.12: Diagrama de la trayectoria propuesta para la prueba dinámica del sistema de visión

3.2. Sistema de soporte estructural de la cámara

En este capítulo se abordan los temas correspondientes al diseño de la estructura de soporte estructural de la cámara, desde los requerimientos y necesidades hasta la selección del diseño final y la validación del mismo.

La metodología de diseño implementada para el desarrollo de la estructura de soporte fue la metodología de Cross [62], en la cual se toman como objetivos principales:

1. Establecer un área de pruebas segura de 3x3m para la formación de los agentes
2. Posicionar la cámara a la altura de trabajo calculada y con condiciones de iluminación controladas.

A partir de los atributos de rendimiento deseados para el sistema (Figura 3.2.1) se determinaron los requerimientos de diseño (Tabla 3.2.1) a considerar para la generación de alternativas.

#	Requerimientos	Grado	
		Demanda	Deseable
1	Dimensiones		
	Longitud de 3m	X	
	Ancho de 3m	X	
	Altura de trabajo de 4m	X	
2	Mantenimiento		
	Mantenimiento sencillo		X
	Piezas fácilmente intercambiables		X
3	Forma		
	Desmontable	X	
	Almacenable	X	
4	Utilización		
	Ensamble sencillo		X
	Altura de cámara ajustable		X
5	Capacidad de carga		
	Al menos 350Kg en plataforma	X	
6	Condiciones de trabajo		
	Resistente a la lluvia y el viento	X	
	Iluminación controlada	X	

Tabla 3.2.1: Requerimientos del sistema de soporte estructural de cámara

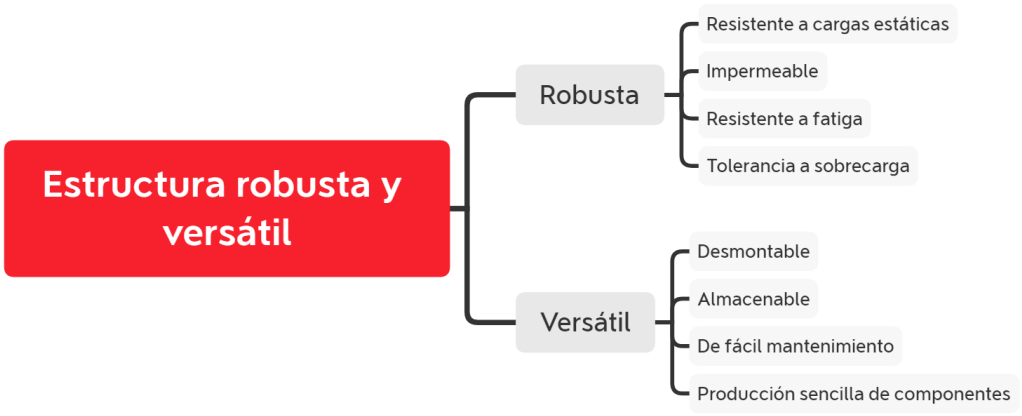


Figura 3.2.1: Atributos de rendimiento deseables en el sistema de soporte estructural

En la tabla 3.2.2 se organizan las rutas de diseño alternativas que se consideraron para la estructura, la primera ruta se identifica con el color verde mientras que la segunda posee un color azul.

Concepto	Opción 1	Opción 2	Opción 3
Establecer área de pruebas de 3x3m	Paneles modulares sobre estructura con niveladores	Panel de 3x3m sobre el suelo	Panel de 3x3 sobre estructura
Estructura desmontable	Uniones con tornillos y abrazaderas	Uniones por inserción	Uniones por inserción con ajuste por tornillo
Iluminación controlada	Iluminación colocada en techo	Colocada en postes laterales	Colocada en techo y postes laterales
Estructura resistente a lluvia y viento	Techo a dos aguas con tensores laterales anclados	Techo a dos aguas con tensores verticales	Techo a dos aguas con soporte diagonal interno
Altura de cámara ajustable	Mecanismo piñón corredera con soporte para cámara	Perfil tubular con ajuste por tornillo y soporte para cámara	Pistón y soporte para cámara

Tabla 3.2.2: Alternativas de diseño para estructura de soporte

3.2.1. Selección de diseño

A partir de las alternativas de diseño generadas que se observan en la tabla 3.2.2, se propusieron dos rutas de diseño las cuales fueron evaluadas siguiendo el método de los factores ponderados, los criterios de evaluación, así como su peso relativo se muestran en la Tabla 3.2.3.

Factores	Peso relativo %	Alternativas	
		A	B
Precio	30	9	8
Confiabilidad	35	9	9
Facilidad de mantenimiento y/o calibración	10	8	7
Facilidad de manufactura o adquisión	15	9	8
Facilidad de ensamble	10	9	8
Puntuación total		**8.9**	**8.25**

Tabla 3.2.3: Factores ponderados para la selección del tipo de estructura

La representación mediante CAD de la plataforma experimental se realizó con el programa de diseño especializado Solid Works, debido a que éste permite realizar el modelado y ensamble de los elementos mecánicos de la estructura con gran facilidad, además de permitir una visualización precisa del modelo 3D. En la figura 3.2.2 se presentan las vistas frontal e inferior de la estructura de soporte diseñada, por otro lado en la figura 3.2.3 se puede observar la vista isométrica completa.

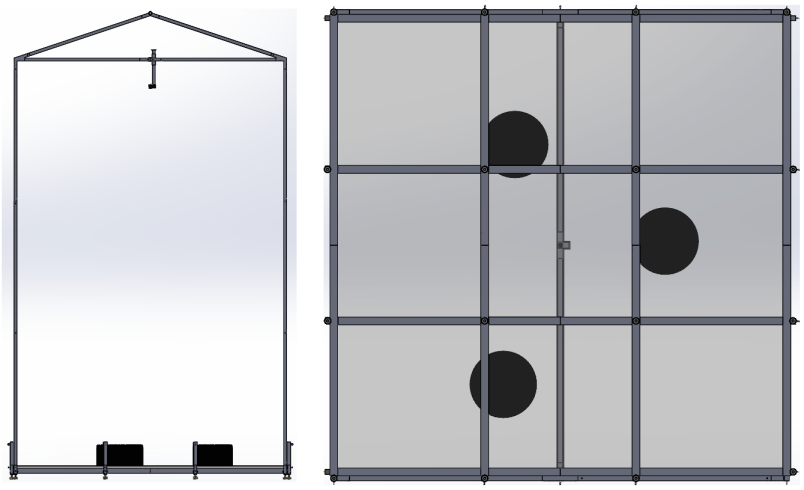


Figura 3.2.2: Vista frontal e inferior de la estructura de soporte diseñada

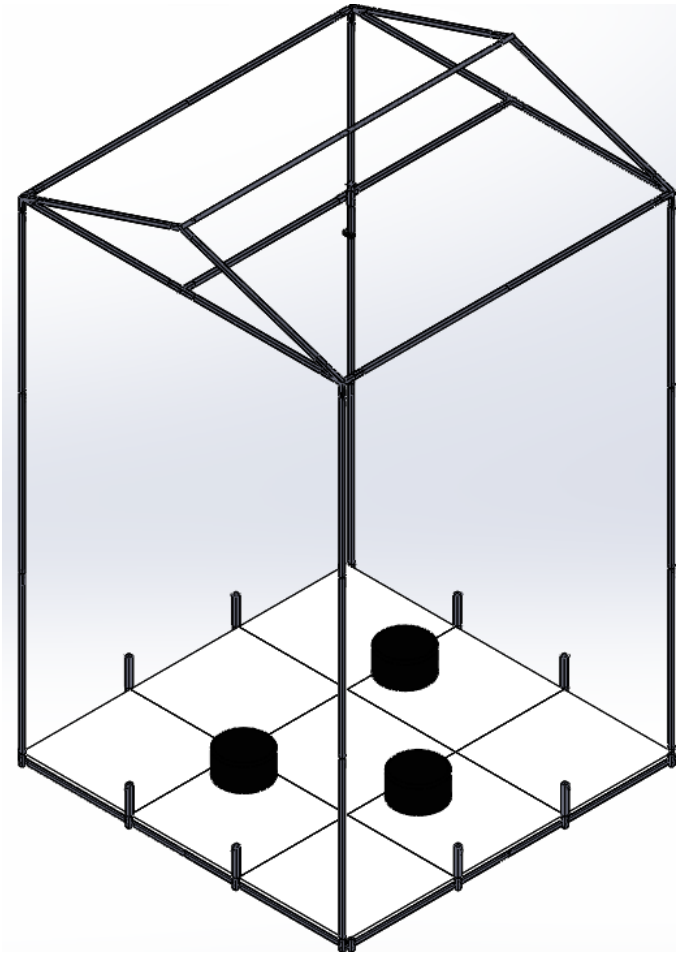


Figura 3.2.3: Vista isométrica de la estructura de soporte diseñada

3.2.2. Análisis de esfuerzos de la plataforma

Cada uno de los componentes de la estructura que se diseñaron ahora se analizarán con métodos computacionales para demostrar que efectivamente el diseño está dentro de los valores límites de esfuerzo y deformación y no se presentará ninguna falla. El software CAD utilizado fue COMSOL (con licencia estudiantil). Las cargas a las cuales será sometida la estructura se calcularon considerando el peso de los tres agentes, el material seleccionado para la estructura así como para la plancha del área de pruebas y el peso ocasional de una persona.
La estructura principal y las uniones se fabricaron con perfiles tubulares cuadrados calibre 18 de acero estructural A-36 mientras que la sección del área de pruebas se elaboró con nueve planchas de MDF (fibra de madera de media densidad) con chapa de melamina blanca mate para facilitar la detección de los agentes por el sistema de visión. El peso de toda la estructura se calculó tomando en cuenta la densidad de los materiales empleados y el volumen de cada una de las piezas (tabla 3.2.4).

Análisis de cargas					
Pieza	**Cantidad**	**Volumen** (m^3)	**Densidad** (Kgm^3)	**Masa (Kg)**	**Masa total (Kg)**
Perfil tubular cuadrado 1	12	$4.75x10^{-4}$	7860	3.73	44.82
Perfil tubular cuadrado 2	4	$1.85x10^{-4}$	7860	1.46	5.83
Perfil tubular cuadrado 3	8	$2.48x10^{-4}$	7860	1.95	15.59
Unión T	2	$1.03x10^{-4}$	7860	0.81	1.62
Unión B	4	$2.26x10^{-4}$	7860	1.77	7.10
Unión A	2	$1.61x10^{-4}$	7860	1.27	2.53
Soporte de cámara	1	$1.15x10^{-4}$	7860	0.90	0.90
Tubular de cámara	1	$1.44x10^{-4}$	7860	1.13	1.13
Sujetador 3D de cámara	1	$3.62x10^{-4}$	1240	0.04	0.04
MDF con melamina	9	$2.95x10^{-2}$	450	13.28	119.56
				Total	**199.13**

Tabla 3.2.4: Análisis de peso de la estructura de soporte

Es importante mencionar que para el análisis de esfuerzos de algunos componentes como los perfiles tubulares cuadrados que soportan el techo de la estructura, no se consideró la carga total de el resto de los componentes debido a que dichas piezas no soportaban el peso del resto de los componentes. Por ello se realizó un análisis especifico para las diferentes secciones de la estructura.

Base de soporte

El peso de las planchas de MDF y de los agentes es soportado por una estructura modular compuesta de perfiles tubulares rectangulares calibre 18 soldados para formar una superficie de contacto de dos pulgadas. Dicha estructura cuenta con 16 niveladores ubicados en las uniones para permitir colocar la plataforma en cualquier terreno y evitar un ángulo de inclinación no deseado. La carga a la que fue sometida la base para el análisis de esfuerzos fue la combinación de el peso de las planchas de MDF con los tres agentes y una persona (sobre una sola plancha de MDF para simular el escenario donde se presentaría la carga máxima DE 300Kg) así como el peso del techo repartido sobre los cuatro apoyos donde se recargan las columnas.

En la figura 3.2.4 se observa el análisis de esfuerzos por Von Mises para una carga repartida de 500 Kg (La carga esperada de 300Kg se sustituyó por una de 500Kg para trabajar con un mejor factor de seguridad) a lo largo del área de pruebas.

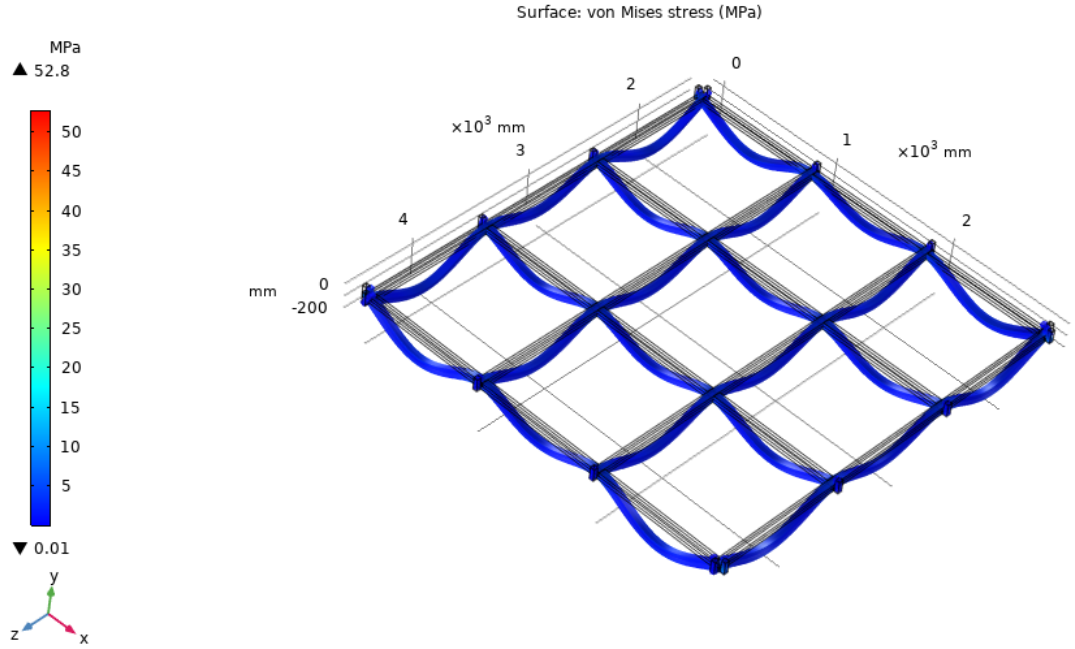


Figura 3.2.4: Análisis de esfuerzos de la base de la estructura de soporte

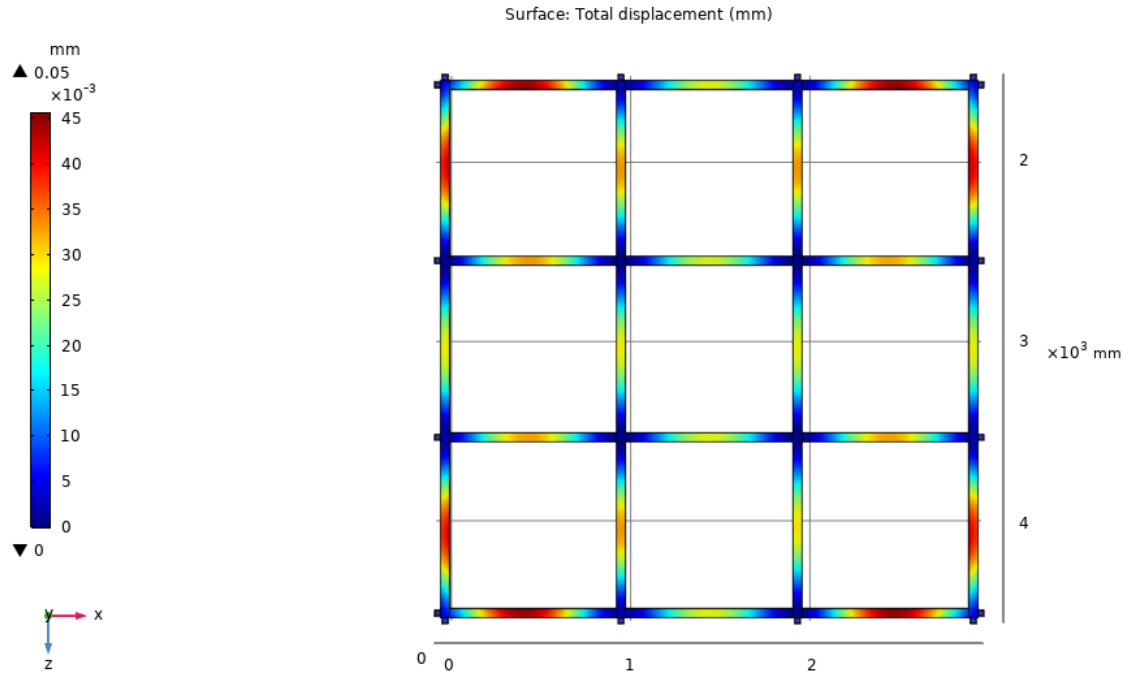


Figura 3.2.5: Deformación de la base de la estructura de soporte al ser sometida a una carga de 4905 N

Tal como se observa en la figura 3.2.4 el esfuerzo máximo presente en la base (indicado de color rojo) tiene el valor de 52.8 MPa, dicho esfuerzo se presenta principalmente en las secciones del perfil tubular más alejadas de los soportes (niveladores). Considerando los coeficientes de seguridad permitidos para cargas muertas en acero (figura 3.2.6) y tomando en cuenta que el objetivo que se busca es que la pieza no se deforme, se realizó el cálculo del factor de seguridad basándonos en la resistencia de fluencia del material.

TIPO O CLASE DE CARGA	ACERO, METALES DÚCTILES		HIERRO FUNDIDO, METALES FRÁGILES	MADERA DE CONSTRUC-CIÓN
	Basado en la resistencia máxima*	Basado en la resistencia de fluencia**	Basado en la resistencia máxima*	
Carga muerta o Carga variable bajo análisis por fatiga	3 – 4	1.5 - 2	5 – 6	7

Figura 3.2.6: Factores de seguridad mínimos recomendados. Tabla modificada de Faires [63]

El esfuerzo de fluencia para el acero estructural A36 según [64] es de 250 MPa, por lo que el coeficiente de seguridad fue:

$$\eta_s = \frac{\sigma_{Fluencia}}{\sigma_{mx}} = \frac{250\ MPa}{52.8\ MPa} = 4.73 \tag{3.14}$$

Por otro lado, la deformación máxima presente se mantiene por debajo de 0.05 mm tal como se aprecia en la figura 3.2.5, considerando que el elemento no rebasa el límite elástico para el cual fue diseñado podemos asumir que la base soportará las cargas a las cuales será sometida.

Plancha de MDF

La carga máxima esperada en una sola plancha de MDF de 1m por 1m y 30 mm de espesor se calculó de 190 Kg, considerando el peso de los tres agentes y una persona (85 Kg). Sin embargo, para asegurar minimizar la deformación presente en el elemento se hizo el análisis de esfuerzos tomando como carga una carga repartida de 500 Kg.

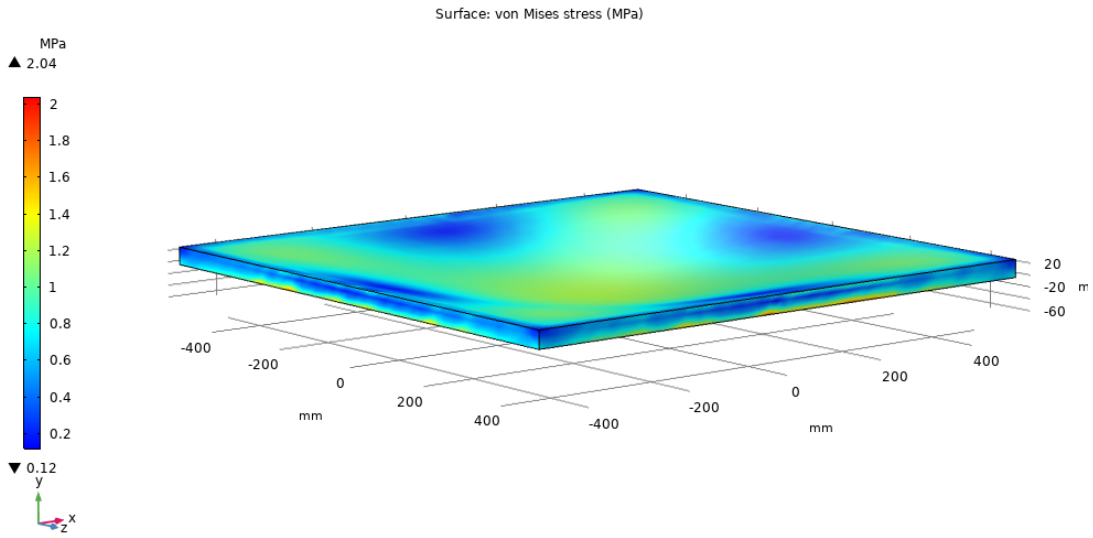


Figura 3.2.7: Análisis de esfuerzos de una placa de MDF de 30mm de espesor sometida a una carga repartida de 4905 N

El módulo de elasticidad de la melamina es de 9 MPa, tomando en cuenta que el esfuerzo máximo presente en el elemento propuesto se encuentra en las orillas donde es apoyado y tiene un valor máximo de 2.04 MPa (como se observa en la figura 3.2.7) se obtuvo un coeficiente de seguridad de:

$$\eta_s = \frac{\sigma_{Fluencia}}{\sigma_{mx}} = \frac{9 \text{ MPa}}{2.04 \text{ MPa}} = 4.41 \tag{3.15}$$

Por lo que se puede afirmar que el elemento propuesto no superará su límite elástico tal como se ve en la figura 3.2.8, donde la deformación máxima ubicada en el centro de la placa, no excede el valor de 1.09 mm.

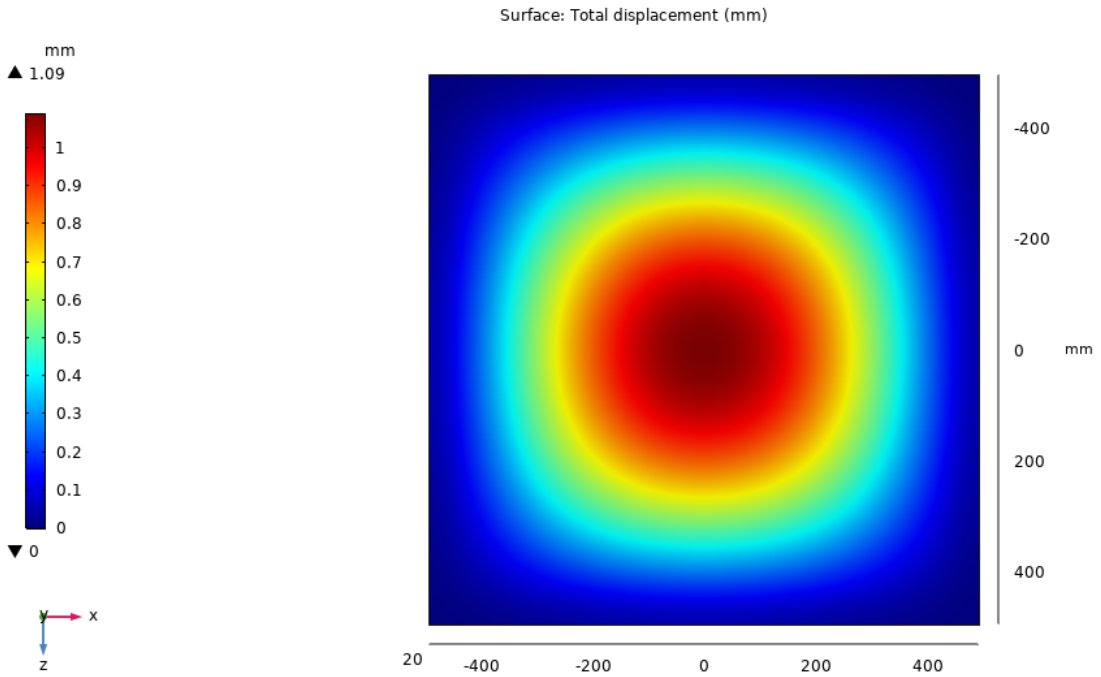


Figura 3.2.8: Deformación de una placa de MDF de 30 mm de espesor sometida a una carga repartida de 4905 N

Nivelador comercial

El nivelador utilizado se seleccionó considerando la carga esperada a la cual estaría sometido y a la facilidad para obtener un remplazo. Considerando el peso de el techo y las columnas, así como la carga repartida de 500Kg calculada, se realizó un análisis de esfuerzos asistido por computadora a un modelo 3D diseñado a partir de la pieza comercial seleccionada. Dichos niveladores poseen un perno con cuerda con un diámetro de 3/8 de pulgada de acero A-36, así como una base de neopreno para compresión.

Como se observa en la figura 3.2.9 el esfuerzo máximo se presenta en la unión del perno con la placa de sujeción (1/4 pulgada de espesor) siendo su valor 113 MPa. El coeficiente de seguridad para el nivelador es:

$$\eta_s = \frac{\sigma_{Fluencia}}{\sigma_{mx}} = \frac{250 \text{ MPa}}{113 \text{ MPa}} = 2.21 \tag{3.16}$$

Con dicho coeficiente de seguridad confirmamos que el nivelador seleccionado es capaz de soportar incluso una carga axial equivalente a 500 Kg, por lo que para cargas menores el factor de seguridad esperado será mayor.

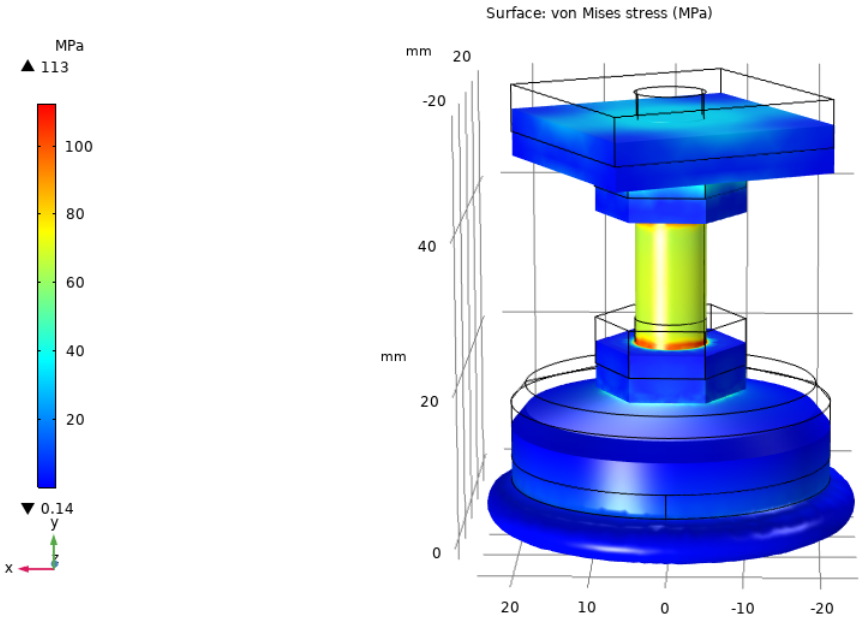


Figura 3.2.9: Análisis de esfuerzos del nivelador seleccionado bajo una carga axial equivalente a 4905 N

Columnas de apoyo

Las columnas que soportan la estructura del techo donde será colocada la cámara y la iluminación están compuestas por el mismo perfil tubular cuadrado calibre 18 de 1 1/4 pulgadas. El material, como se ha mencionado antes es acero estructural A-36 y dichos elementos estarán sometidos a una carga axial de 785 N (80 Kg de la estructura) el análisis de esfuerzos se realizó para una carga axial de 1kN. Tal como se observa en la figura 3.2.10 el esfuerzo máximo se presenta en los apoyos.

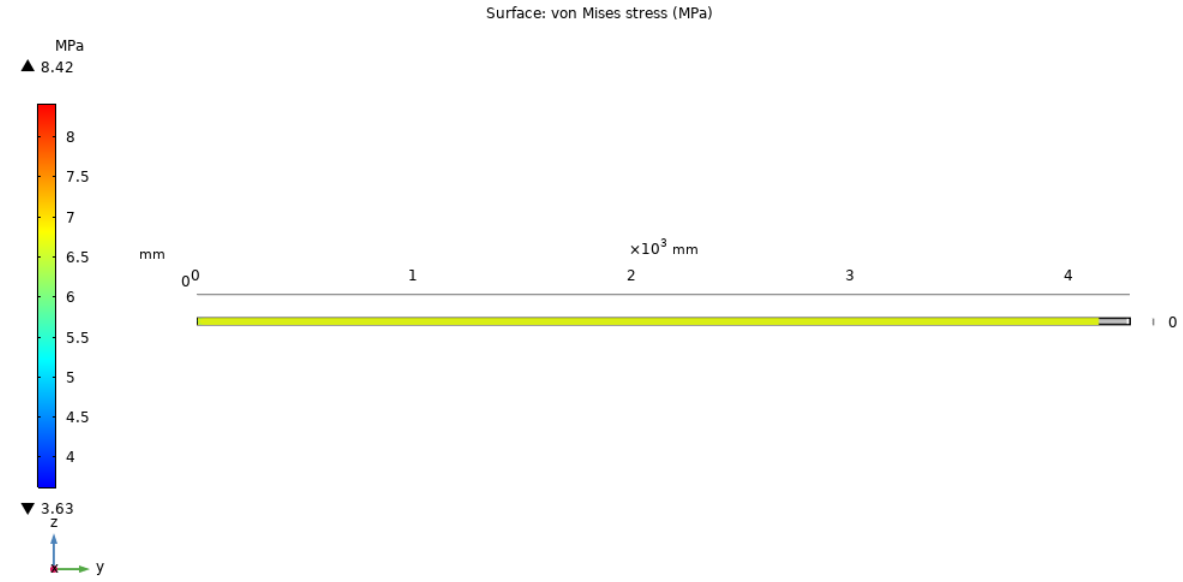


Figura 3.2.10: Análisis de esfuerzos de columna sometida a una carga axial de 1 kN

El esfuerzo máximo presente en la columna es de 8.42 MPa, para lo cual se obtiene un coeficiente de seguridad bastante alto para la aplicación.

$$\eta_s = \frac{\sigma_{Fluencia}}{\sigma_{mx}} = \frac{250 \text{ MPa}}{8.42 \text{ MPa}} = 29.69 \tag{3.17}$$

Por otro lado la carga máxima (considerando únicamente el esfuerzo axial permitido y un factor de seguridad mínimo de 3) permitida en el techo para no pasar el límite elástico del material es de 1000 Kg

$$F_{max} = (\sigma_{Fluencia})(Area) = \frac{(250\ MPa)(1.1876x10^{-4}m^2)}{3} = 9.8966 \text{ kN} \tag{3.18}$$

Por otro lado el esfuerzo combinado resultante de la carga del viento y el peso de la estructura se analizó consultando la velocidad promedio del viento en el área metropolitana (donde será colocada la estructura). Según [60] la velocidad promedio en la ciudad de México es de 7 nudos o $3.60\frac{m}{s}$, la carga debido al viento se obtuvo siguiendo la estimación:

$$F = A_{proyectada}P_{viento}Cd = \left(13.55m^2\right)\left(7.94\frac{N}{m^2}\right)(2) = 215.17 \text{ N} \tag{3.19}$$

Donde $(A_{proyectada})$ es el área proyectada de la estructura que estará en contacto con el viento, Cd es el coeficiente de arrastre el cual para superficies planas perpendiculares al flujo se considera igual a 2, mientras que la presión del viento está relacionada con su velocidad y un coeficiente obtenido por la asociación Mexicana de Ingenieros Civiles [61] a partir de la densidad típica del aire $P = 0.613V^2$.

En la figura 3.2.11 se puede observar el análisis de esfuerzos realizado, la carga de 215.17 N será soportada por las cuatro columnas de la estructura. Sin embargo, se aplico dicha carga a una columna para observar los efectos del viento sobre el elemento en casos extremos. La fuerza máxima fue equivalente a 189 MPa, por lo que se consideró el uso de tensores fijados al suelo para aumentar la rigidez de la estructura ante ráfagas de viento. El coeficiente de seguridad obtenido fue de 1.32.

$$\eta_s = \frac{\sigma_{Fluencia}}{\sigma_{mx}} = \frac{250\ MPa}{189\ MPa} = 1.32 \tag{3.20}$$

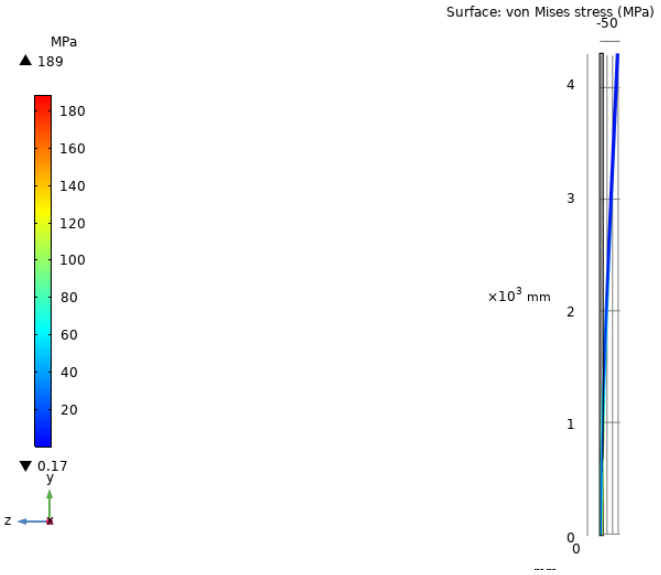


Figura 3.2.11: Análisis de esfuerzos de columna sometida a una ráfaga de viento de 3.60 $\frac{\text{m}}{\text{s}}$

3.3. Sistema de comunicación agentes - control central

Con respecto al subsistema de comunicación con el usuario, se optó por el envío y recepción de los valores de las variables de control de los agentes de forma inalámbrica, en este caso, el mecanismo de comunicación seleccionado y utilizado es la especificación industrial Bluetooth debido a su facilidad de implementación, cabe destacar que se se utilizó un módulo HC-05, mismo que se puede apreciar en la figura 3.3.1. Para ejemplificar la topología de comunicación se elaboró el grafo y su respectiva matriz Laplaciana, ambas se aprecian en la figura 3.3.2.

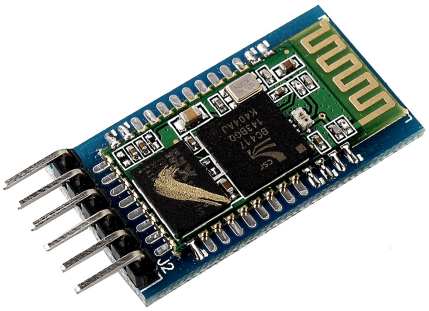


Figura 3.3.1: Módulo Bluetooth HC-05

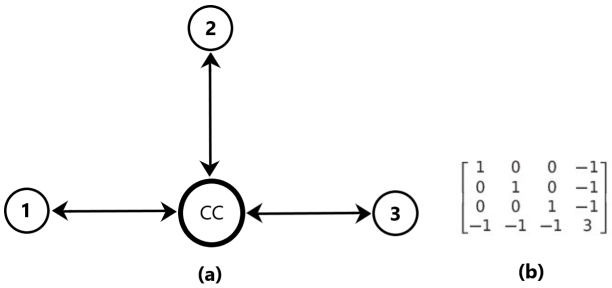


Figura 3.3.2: (a) Grafo de topología de comunicación y (b) su matriz Laplaciana

Diseño de aplicación para teléfono

Para validar el subsistema de comunicación entre los agentes y el control central se desarrolló una aplicación para dispositivos móviles Android capaz de realizar el promedio del tiempo de recepción de los datos, (el proceso se detallará en breve, en un diagrama de flujo) para realizar el proceso de medición se recorrió el área de trabajo en puntos uniformemente distribuidos, como se aprecia en la figura 3.3.3. Cabe destacar que se tomaron un promedio de veinte mediciones en cada punto, dando un total de ciento ochenta mediciones de tiempo, las coordenadas de los puntos se pueden apreciar en la tabla 3.3.1.

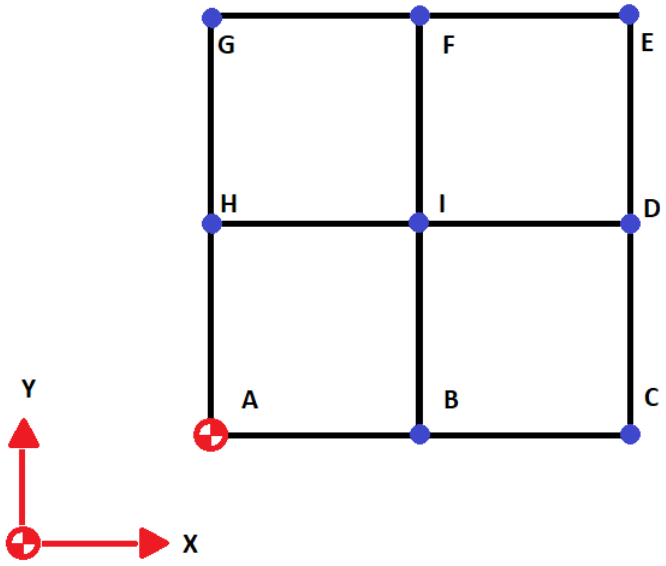


Figura 3.3.3: Disposición del área de trabajo para realizar la prueba de validación

Punto	X[m]	Y[m]
A	0	0
B	1.5	0
C	3	0
D	3	1.5
E	3	3
F	1.5	3
G	0	3
H	0	1.5
I	1.5	1.5

Tabla 3.3.1: Puntos evaluados en la prueba de comunicación

A continuación en la figura 3.3.4 se muestra el diagrama de flujo que corresponde al funcionamiento de la aplicación desarrollada.

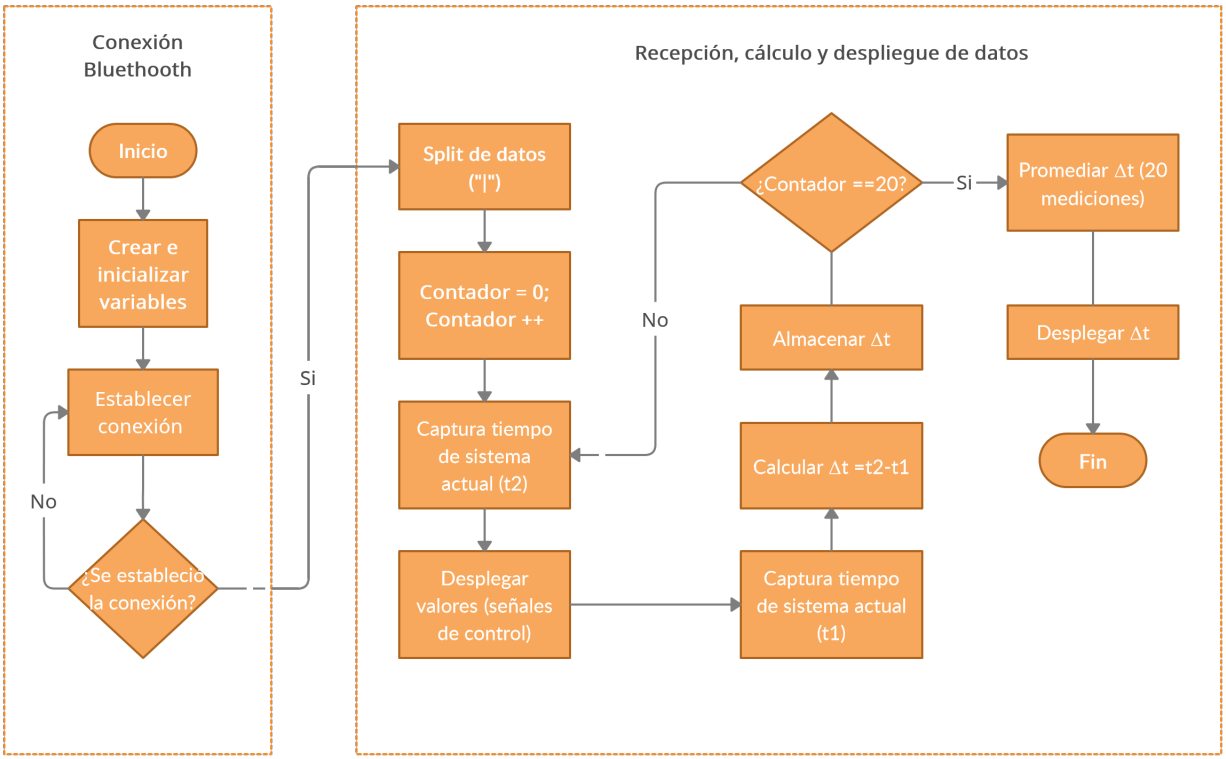


Figura 3.3.4: Diagrama de flujo de la aplicación

Visualización y operación de la aplicación

Cabe destacar que el diagrama de bloques propio de dicha aplicación se presenta en el apéndice C, en este caso el desarrollo de la aplicación se programó en la plataforma MIT AppInventor. A continuación se describe el proceso del manejo de la aplicación en pasos.

1. En primera instancia se solicita la conexión al módulo presionando el ícono *Bluetooth* como se puede apreciar en la figura 3.3.5, en este momento el estado de la aplicación se debería visualizar en color rojo y con la leyenda : **Status: Desconectado**
2. Se verifica que la conexión ha sido exitosa ya que el estado de aplicación debería cambiar a color verde y con la leyenda: **Status : Conectado**, como se puede apreciar en la figura 3.3.5, además, se deberán visualizar los valores de las señales de control que corresponden a cada RMO.

A continuación, en las figuras 3.3.6 y 3.3.5 se muestra gráficamente la aplicación desarrollada por los integrantes del equipo.

Figura 3.3.5: Aplicación antes de realizar la conexión

Figura 3.3.6: Aplicación tras haber realizado exitosamente la conexión

3.4. Sistema interfaz de usuario

Como se comentó anteriormente, se desarrolló una Interfaz Gráfica de usuario para poder visualizar gráficamente la trayectoria de los RMO, para lograr dicho desarrollo, se utilizó el paquete *de facto* Tkinter, integrado en Python para creación de GUIs. En la GUI, en primera instancia, se han almacenado los datos que corresponden a la trayectoria y ángulo de rotación de cada agente; sin embargo, para fines interactivos, se han colocado tres botones con la leyenda : *Graficar* debajo de cada una de las gráficas, como se puede apreciar en la figura 3.4.1.

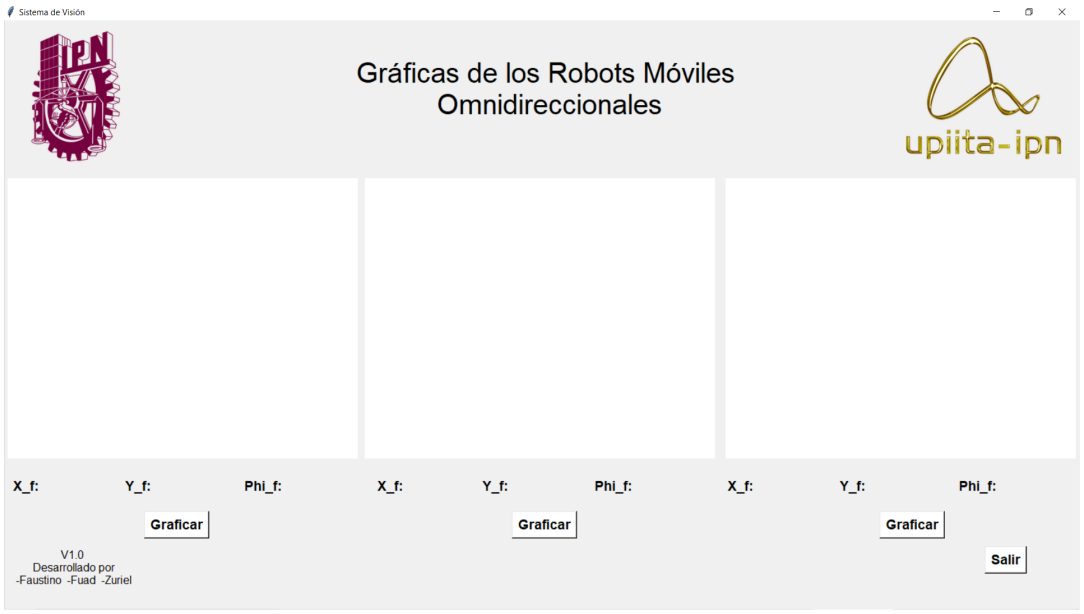


Figura 3.4.1: Interfaz Gráfica de Usuario desarrollada por el equipo antes de desplegar los datos

Una vez presionados los tres botones con la leyenda : *Graficar*, se muestran las gráficas de los tres agentes así como sus posiciones finales en x, y y su ángulo de rotación Φ final, como se puede apreciar en la figura 3.4.2

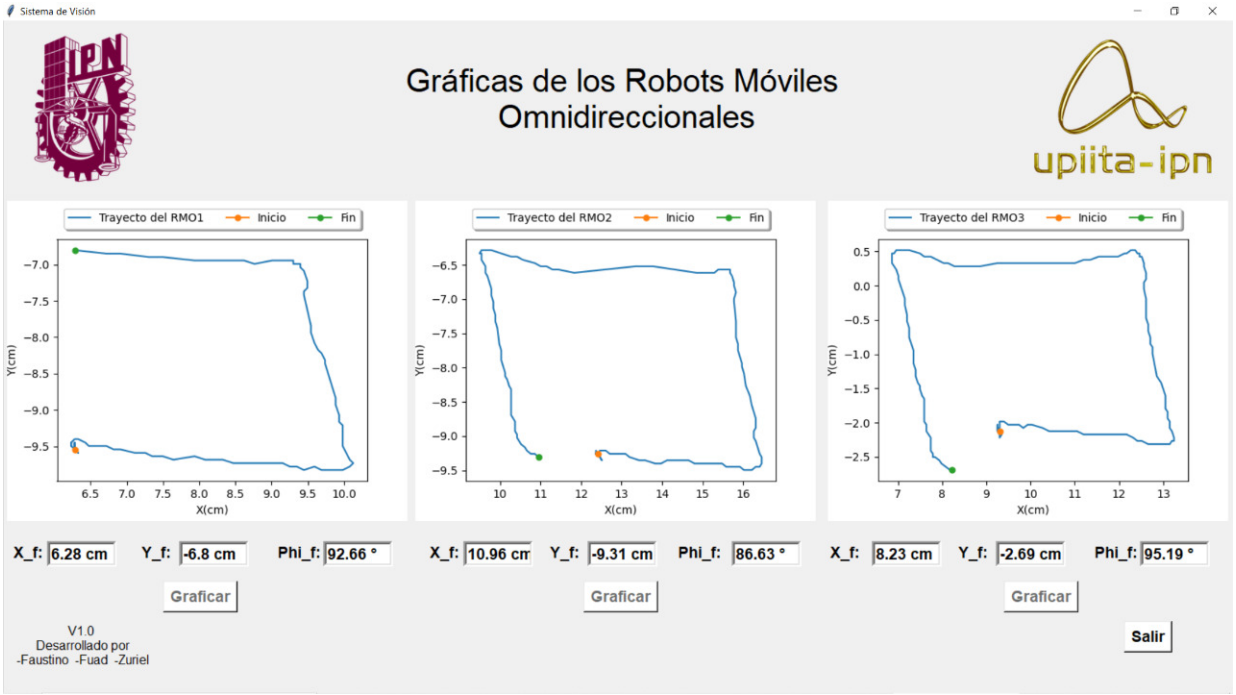


Figura 3.4.2: Interfaz Gráfica de Usuario desarrollada por el equipo después de desplegar los datos

Capítulo 4

Resultados

En el presente capítulo se muestran y analizan los resultados obtenidos de cada uno de los experimentos (definidos en el capítulo tres) y simulaciones realizadas para la validación de los distintos sistemas que permiten el control cooperativo centralizado de los agentes, así como la evidencia fotográfica del ensamble de la plataforma experimental.

Las simulaciones del consenso y la formación de los agentes fueron realizadas con la herramienta matemática MATLAB junto con el paquete Simulink, mientras que las pruebas experimentales se realizaron en un espacio de trabajo controlado con las condiciones y características mencionadas en el capítulo 3.

4.1. Simulación numérica del control cooperativo centralizado para el consenso

La simulación numérica integra herramientas matemáticas que permiten modelar o predecir el comportamiento de algún sistema con el propósito de comprender su funcionamiento o visualizar su respuesta específica ante cierto estímulo. En esta sección se analiza la simulación en tiempo continuo que tiene como fin modelar el comportamiento de la estrategia de control cooperativo implementada para el consenso de los agentes.

El objetivo principal del experimento realizado es demostrar mediante una simulación numérica, el consenso de los tres grados de libertad de cada uno de los agentes con el promedio de sus condiciones iniciales. Para ello, se plantea colocar a los tres agentes en posiciones iniciales distintas escogidas de forma aleatoria alrededor del área de trabajo (la cual es de 3 metros de largo por 3 m de ancho), posteriormente el algoritmo de control implementado deberá calcular el promedio de las condiciones iniciales de cada agente y registrarlas como referencia para el consenso promedio, tras ejecutar la estrategia de control para el consenso del SMA propuesta en el capítulo dos, los agentes deberán converger en un mismo punto del área de pruebas (en este caso el consenso promedio).

En el esquema mostrado en la figura 4.1.1 se representa la estrategia general implementada para el consenso.

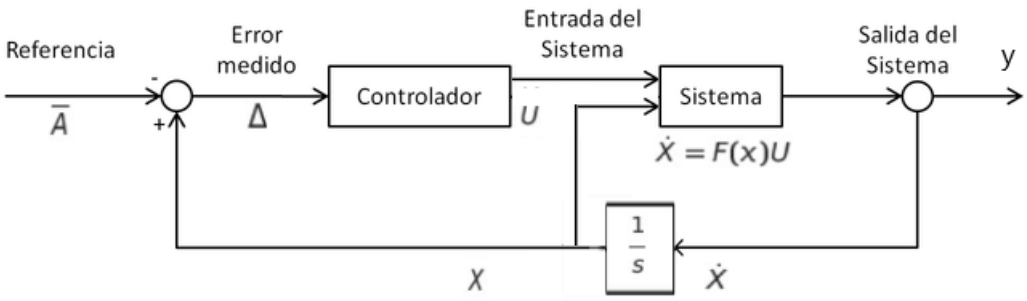


Figura 4.1.1: Esquema general del control para el consenso

Para implementar la estrategia propuesta en el esquema de la figura 4.1.1 se utilizó el programa MATLAB-Simulink debido al entorno de programación visual que emplea, el cual permite trabajar con bloques para la simulación de sistemas, lo cual facilita el entendimiento del comportamiento general de la estrategia de control. El diagrama de control presentado en la figura 4.1.2 se compone de dos bloques principales, control estabilizante y modelo cinemático del sistema multiagente (Dichas estrategias se revisan a detalle en el capítulo 2). Cabe mencionar que la referencia deseada es proporcionada por el sistema de visión y corresponde al promedio de las condiciones iniciales de los tres agentes para el caso del problema de consenso.

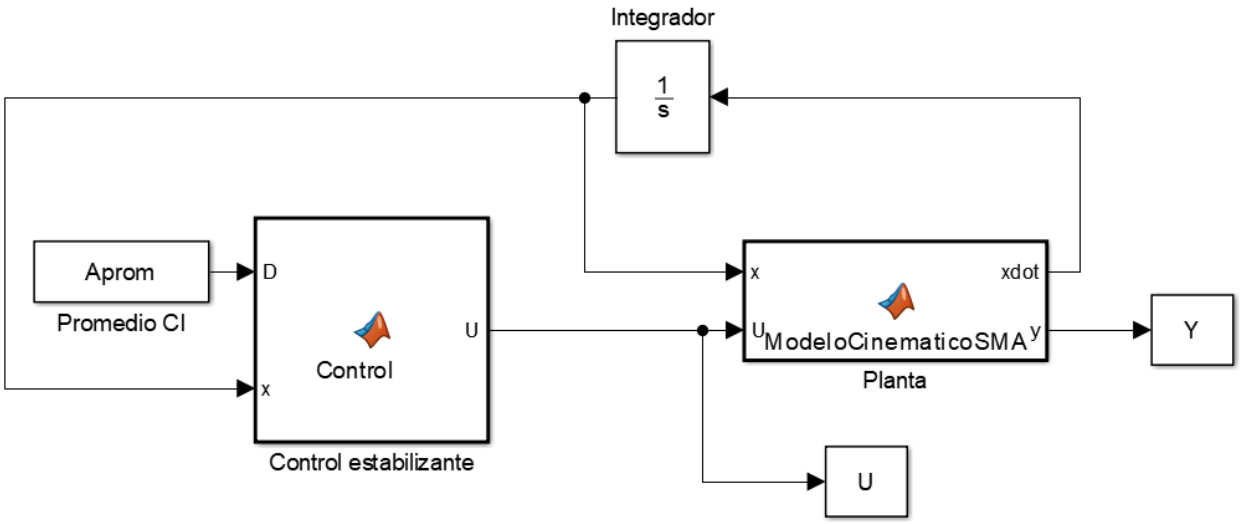


Figura 4.1.2: Diagrama en MATLAB-Simulink para la simulación del consenso

Tomando en cuenta que la validación del sistema de visión se realizó de manera independiente para asegurar la precisión de las condiciones iniciales detectadas para cada agente, se propusieron tres posiciones iniciales aleatoriamente con las cuales comprobar el funcionamiento de la estrategia de control implementada para el consenso (considerando que se pretende en trabajo futuro recibir las condiciones iniciales a través del sistema de posicionamiento). Las condiciones iniciales para las nueve variables de estado se presentan en la tabla que se muestra a continuación. El objetivo del consenso se logra cuando: $\lim_{t\to\infty} x_j^i = x_j^k \quad y \quad \dot{V} \leq 0$.

Variable		X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
Valor	Posición 1	1 m	2.5 m	0.7 m	2.5 m	1.5 m	0.7 m	0 rad	2π rad	π rad
	Posición 2	0.5 m	1 m	2.5 m	1 m	2.5 m	1.5 m	2π rad	$\frac{\pi}{4}$ rad	$\frac{\pi}{2}$ rad
	Posición 3	2 m	1.5 m	0.5 m	2 m	1 m	1 m	$\frac{\pi}{4}$ rad	0 rad	$\frac{\pi}{3}$ rad

Tabla 4.1.1: Posiciones iniciales propuestas para la simulación del consenso

Tras implementar la estrategia de control para el consenso en MATLAB-Simulink para la primera posición se obtuvieron los resultados mostrados en las figuras 4.1.3 -4.1.7. Tal como se observa en la figura 4.1.6, los agentes logran el consenso dentro del tiempo de simulación establecido (t = 5 [s]). Por otro lado, en la figura 4.1.7 se muestra como la señal del controlador decrece conforme $t \longrightarrow \infty$ y se alcanza el consenso. Este comportamiento se presenta independientemente de las condiciones iniciales de los agentes, en las figuras 4.1.8 - 4.1.12 se presentan los resultados obtenidos para la segunda formación, en la cual el consenso se logra en el mismo intervalo de tiempo. Finalmente, en la figuras 4.1.13 a 4.1.17 se observa un comportamiento similar en la tercera formación.

Cabe mencionar que el método de integración utilizado fue el método de integración numérica Runge-Kutta con un paso fijo de 0.001 [s] debido a que proporciona un pequeño margen de error con respecto a la solución real del problema y es fácilmente programable en un software para realizar las iteraciones necesarias.

Tal como se menciona en el capítulo 2, las variables de estado $x = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]$ representan el espacio de operación X_w, Y_w, ϕ_w de cada uno de los agentes con respecto al sistema de coordenadas inercial.

4.1.1. Primera posición

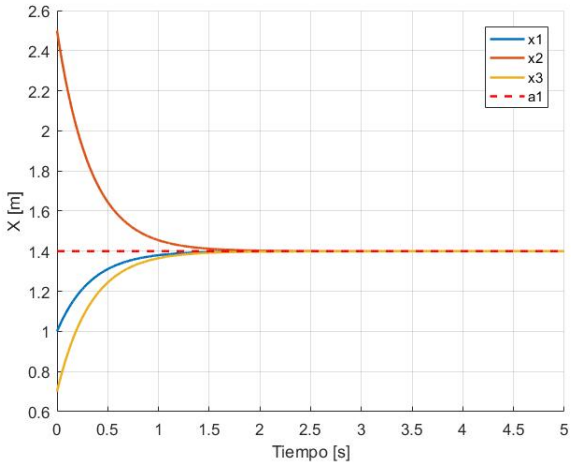


Figura 4.1.3: Consenso del primer grado de libertad de los tres agentes para la posición uno.

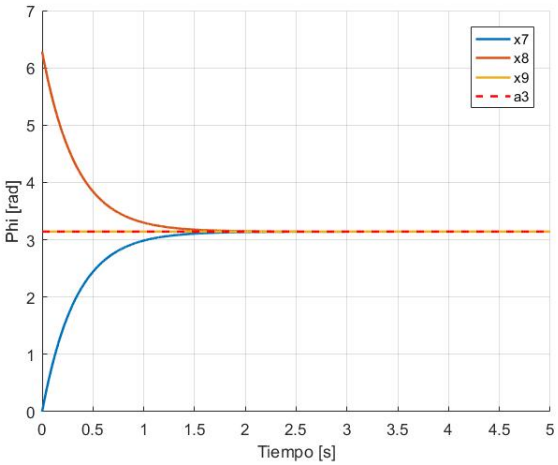


Figura 4.1.5: Consenso del tercer grado de libertad de los tres agentes para la posición uno.

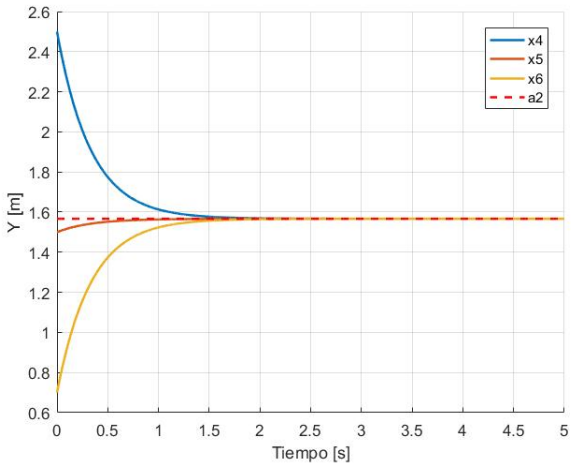


Figura 4.1.4: Consenso del segundo grado de libertad de los tres agentes para la posición uno.

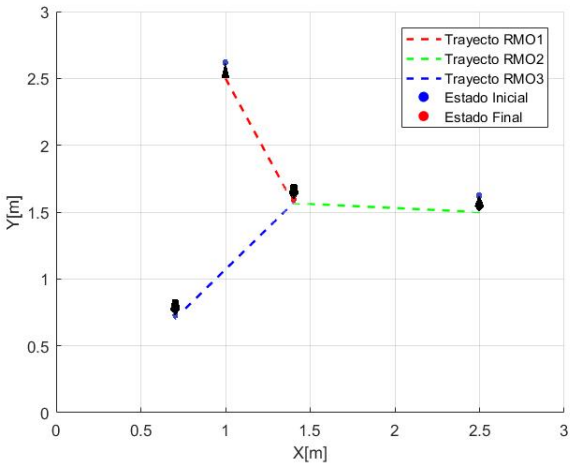


Figura 4.1.6: Trayectoria realizada por cada agente desde la primera posición al consenso.

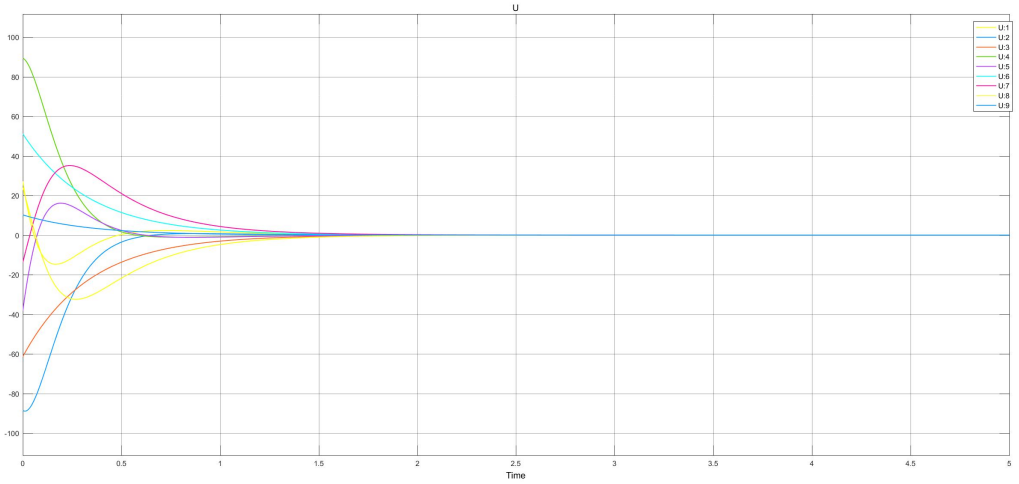


Figura 4.1.7: Señal de control de la primera posición

4.1.2. Segunda posición

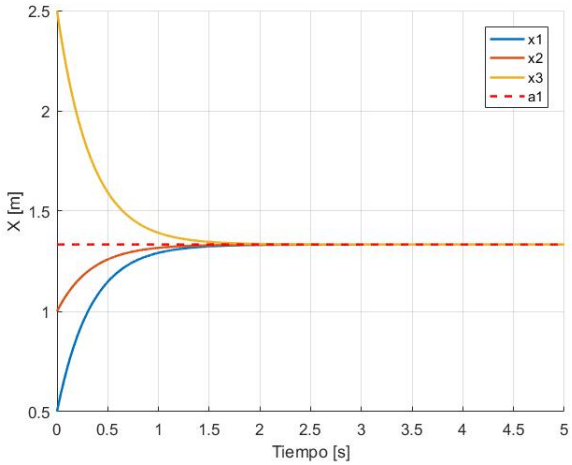


Figura 4.1.8: Consenso del primer grado de libertad de los tres agentes para la posición dos.

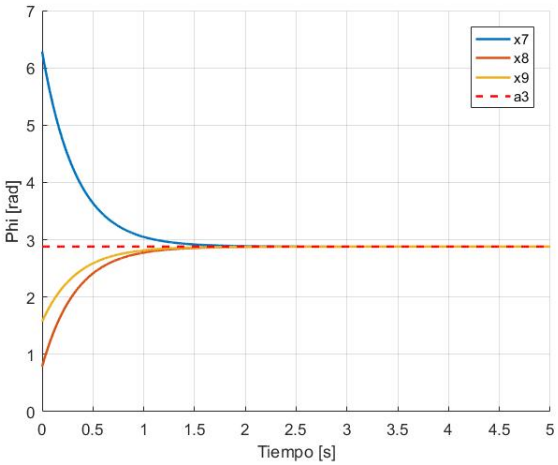


Figura 4.1.10: Consenso del tercer grado de libertad de los tres agentes para la posición dos.

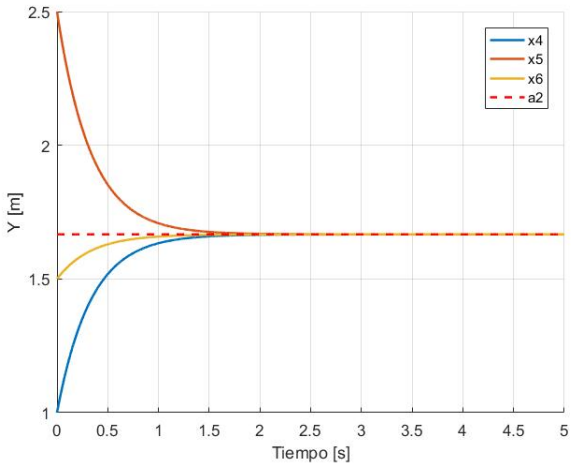


Figura 4.1.9: Consenso del segundo grado de libertad de los tres agentes para la posición dos.

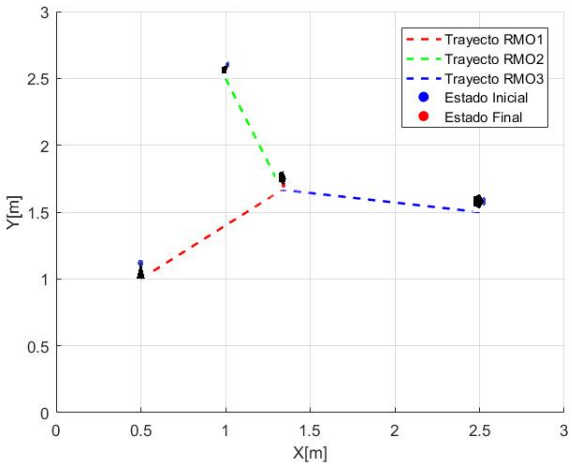


Figura 4.1.11: Trayectoria realizada por cada agente desde la segunda posición al consenso.

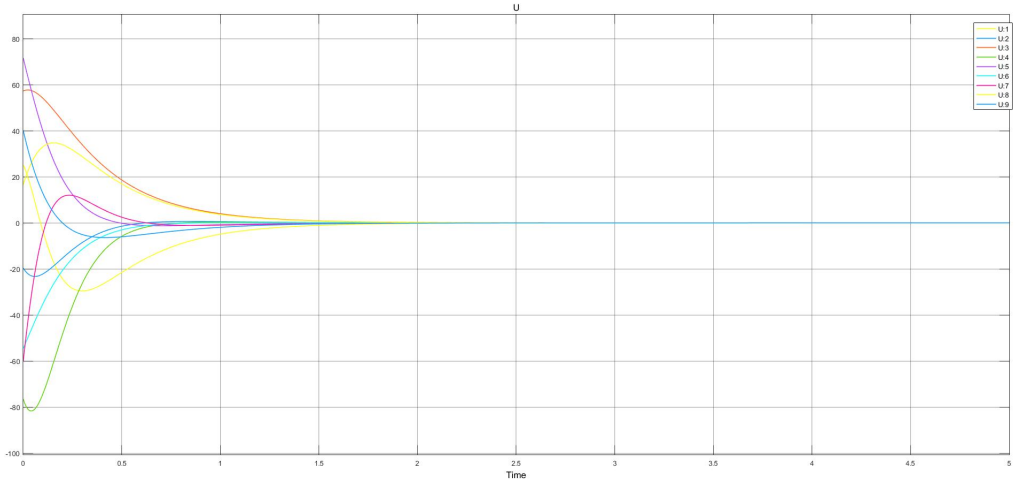


Figura 4.1.12: Señal de control de la segunda posición

4.1.3. Tercera posición

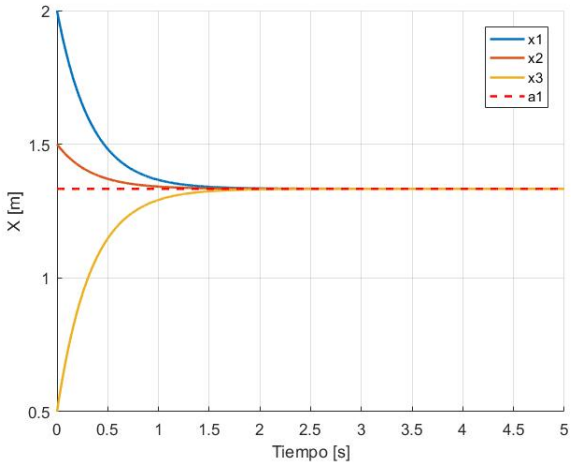


Figura 4.1.13: Consenso del primer grado de libertad de los tres agentes para la posición tres.

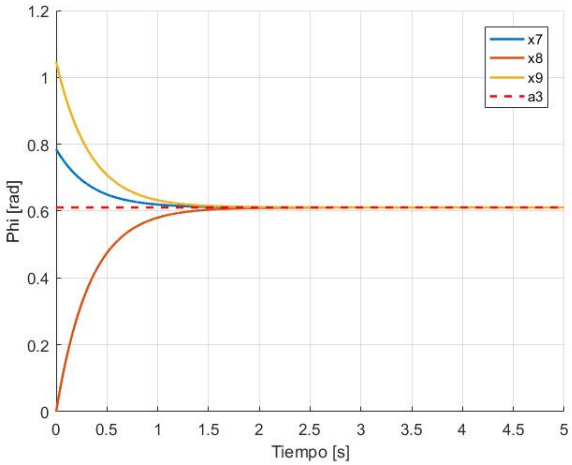


Figura 4.1.15: Consenso del tercer grado de libertad de los tres agentes para la posición tres.

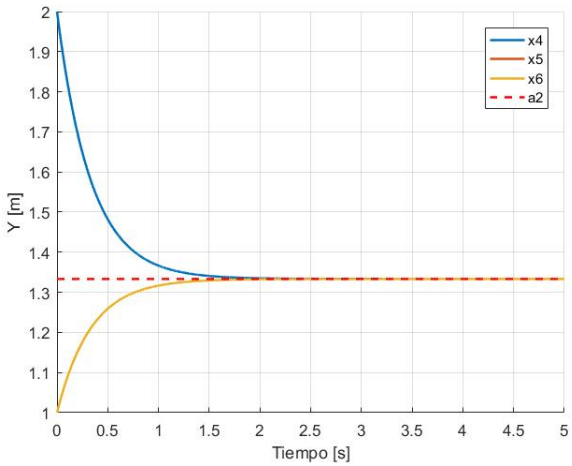


Figura 4.1.14: Consenso del segundo grado de libertad de los tres agentes para la posición tres.

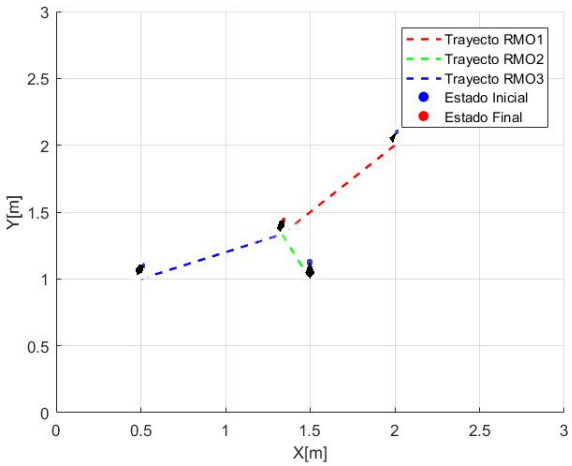


Figura 4.1.16: Trayectoria realizada por cada agente desde la tercera posición al consenso.

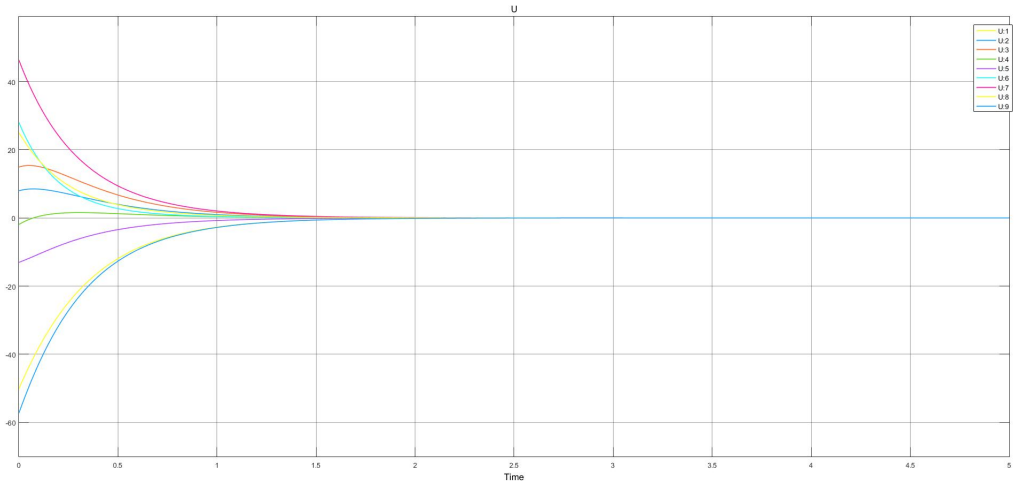


Figura 4.1.17: Señal de control de la tercera posición

4.2. Simulación numérica del control cooperativo centralizado para la formación

En esta sección se abordan los resultados de la simulación numérica de la estrategia de control centralizado para la formación de los agentes implementada en MATLAB-Simulink (propuesta en el capítulo dos). Posteriormente se analizan e interpretan los resultados numéricos para validar la solución del problema de formación.

El experimento consistió en la implementación de tres formaciones distintas a las que los agentes debían llegar, partiendo de las condiciones iniciales definidas en la tabla 4.1.1. Tal como se detalló en el capítulo dos, la formación de los agentes se logra cuando la posición y ángulo de rotación de cada robot convergen con la referencia deseada, la cual se encuentra a una distancia radial "r" y un a ángulo de rotación "β" alrededor del consenso promedio $\overline{A}$. Es decir, para cada grado de libertad de los agentes existirán de una a tres referencias las cuales debe alcanzar.

En las figuras 4.2.1 - 4.2.3 se muestran las formaciones propuestas para validar la estrategia de control.

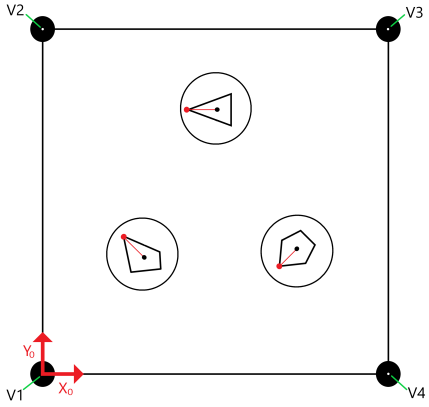


Figura 4.2.1: Primera formación propuesta para la simulación numérica.

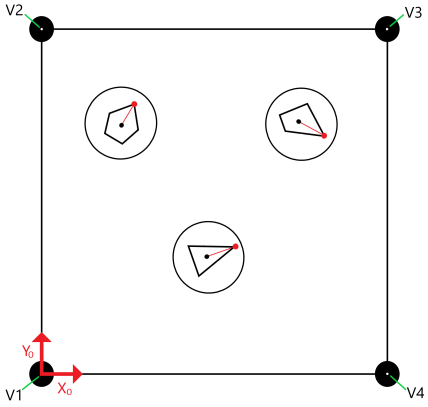


Figura 4.2.2: Segunda formación propuesta para la simulación numérica.

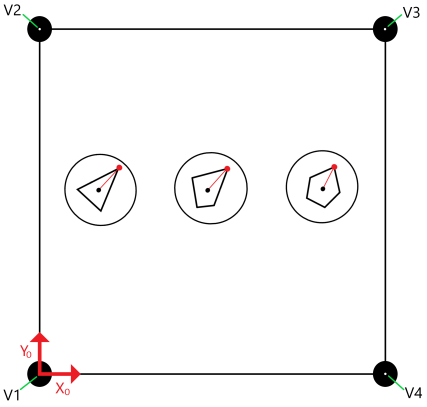


Figura 4.2.3: Tercera formación propuesta para la simulación numérica.

A continuación, en la tabla 4.2.1 se presentan los resultados numéricos obtenidos para las tres formaciones y su error, posteriormente en las figuras 4.2.4 - 4.2.18 se muestran los resultados gráficos.

Formación		X_1	X_2	X_3	X_4	X_5	x_6	X_7	X_8	X_9
1	Deseada	1.4000 m	1.0464 m	1.7536 m	2.5667 m	1.9202 m	1.9202 m	4.7124 rad	5.4978 rad	3.9270 rad
	Final	1.4000 m	1.0465 m	1.7536 m	2.5665 m	1.9200 m	1.9200 m	4.7121 rad	5.4973 rad	3.9266 rad
	Error	0.0000 m	0.0001 m	0.0000 m	0.0002 m	0.0002 m	0.0002 m	0.0003 rad	0.0005 rad	0.0004 rad
2	Deseada	1.3333 m	1.6869 m	0.9798 m	0.5000 m	1.1464 m	1.1464 m	7.5922 rad	8.3776 rad	6.8068 rad
	Final	1.3334 m	1.6869 m	0.9798 m	0.5001 m	1.1466 m	1.1466 m	7.5909 rad	8.3758 rad	6.8050 rad
	Error	0.0000 m	0.0000 m	0.0000 m	0.0001 m	0.0002 m	0.0002 m	0.0013 rad	0.0018 rad	0.0018 rad
3	Deseada	1.5000 m	2.0000 m	2.5000 m	1.3333 m	1.3333 m	1.3333 m	0.6109 rad	0.6109 rad	0.6109 rad
	Final	1.4999 m	1.9998 m	2.4998 m	1.3334 m	1.3334 m	1.3334 m	0.6109 rad	0.6109 rad	0.6109 rad
	Error	0.0001 m	0.0002 m	0.0002 m	0.0001 m	0.0001 m	0.0001 m	0.0000 rad	0.0000 rad	0.0000 rad

Tabla 4.2.1: Resultados numéricos de la formación

Tal como se aprecia en la tabla 4.2.1, la diferencia entre el valor deseado y la posición final es mínima (menor a 0.001 m), este resultado se ve representado de forma gráfica en las figuras 4.2.4 - 4.2.18. en las cuales se presenta la evolución de los tres grados de libertad de los agentes en el tiempo. En las tres formaciones se alcanza el objetivo (línea punteada) en un $t < 15$s.

Por otro lado, en las figuras 4.2.7, 4.2.12 y 4.2.17 se aprecia la trayectoria realizada por cada uno de los agentes a lo largo del área de trabajo para realizar la formación solicitada. Finalmente en las figuras 4.2.8, 4.2.13 y 4.2.18 presentan la señal de control del sistema, la cual alcanza el objetivo en $t < 3$s. Con lo cual se concluye el experimento con resultados satisfactorios.

4.2.1. Primera formación

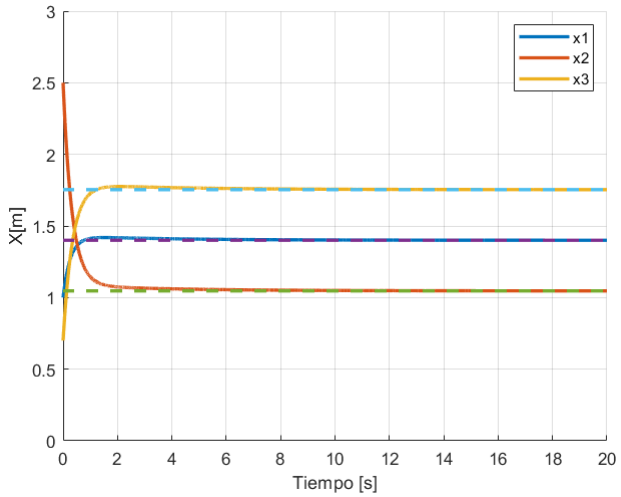


Figura 4.2.4: Formación del primer grado de libertad de los tres agentes para la primera referencia.

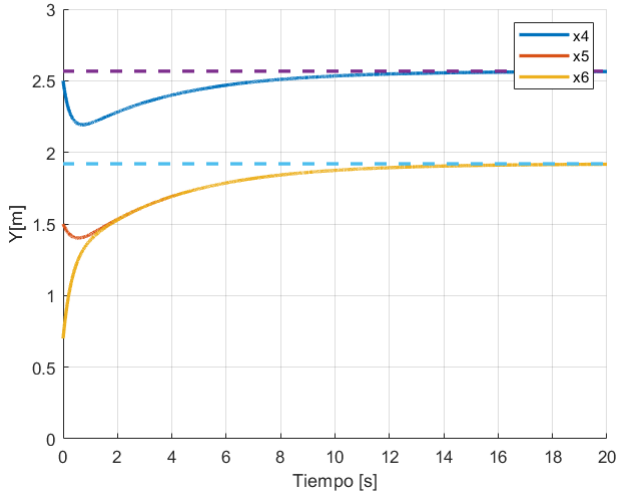


Figura 4.2.5: Formación del segundo grado de libertad de los tres agentes para la primera referencia.

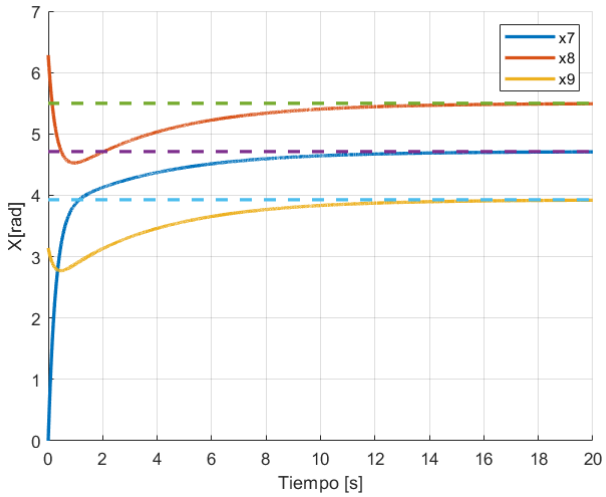


Figura 4.2.6: Formación del tercer grado de libertad de los tres agentes para la primera referencia.

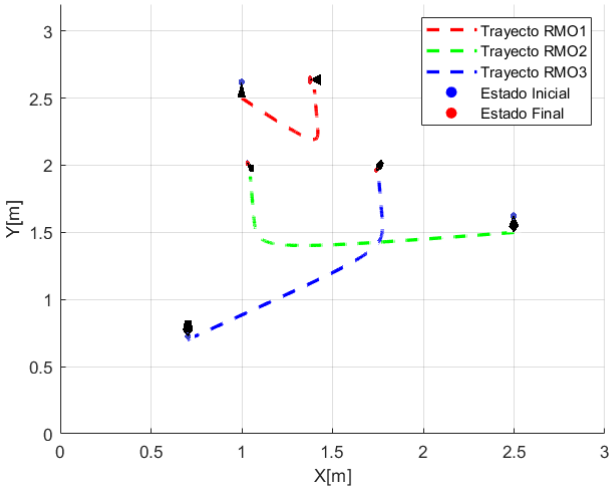


Figura 4.2.7: Trayectoria realizada por cada agente desde sus condiciones iniciales a la primera formación.

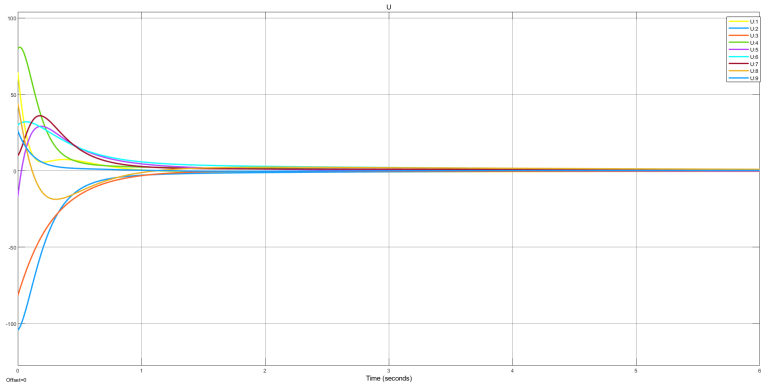


Figura 4.2.8: Señal de control de la primera formación.

4.2.2. Segunda formación

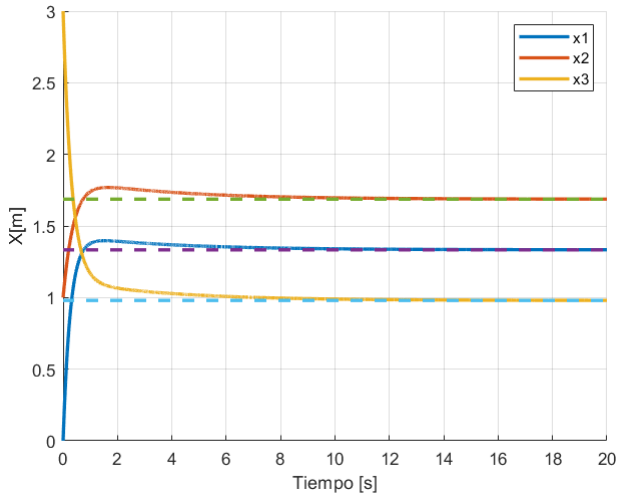


Figura 4.2.9: Formación del primer grado de libertad de los tres agentes para la segunda referencia.

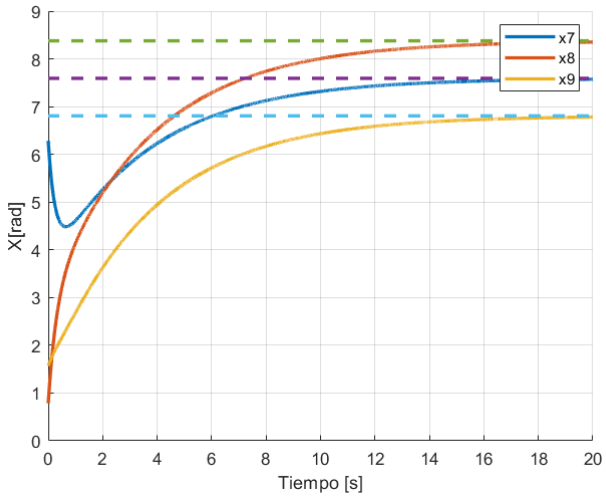


Figura 4.2.11: Formación del tercer grado de libertad de los tres agentes para la segunda referencia.

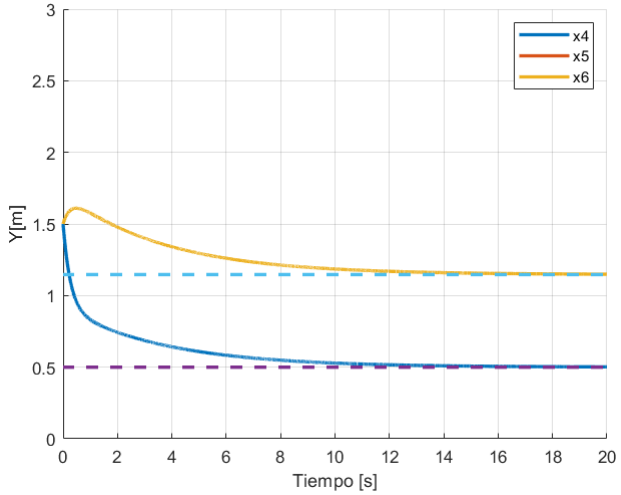


Figura 4.2.10: Formación del segundo grado de libertad de los tres agentes para la segunda referencia.

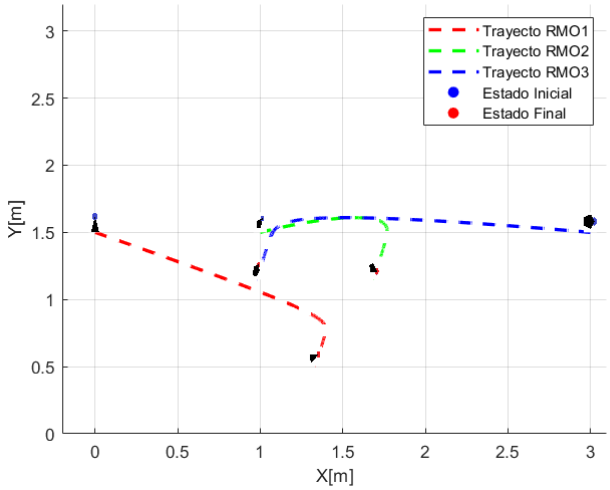


Figura 4.2.12: Trayectoria realizada por cada agente desde sus condiciones iniciales a la segunda formación.

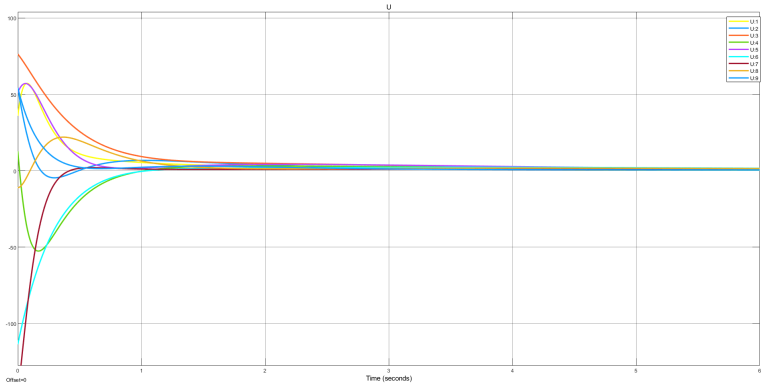


Figura 4.2.13: Señal de control de la primera formación.

4.2.3. Tercera formación

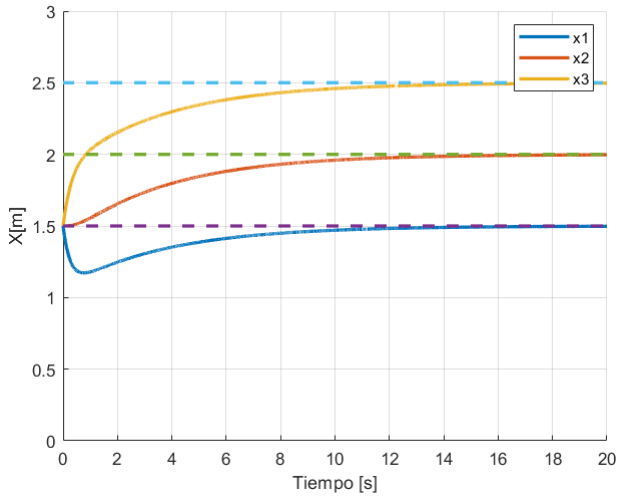


Figura 4.2.14: Formación del primer grado de libertad de los tres agentes para la tercera referencia.

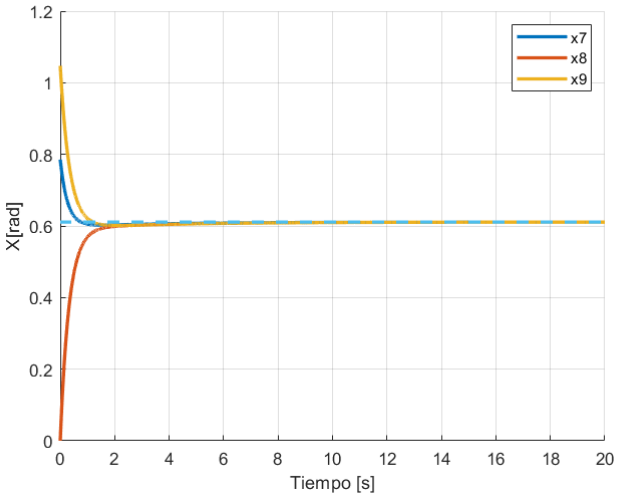


Figura 4.2.16: Formación del tercer grado de libertad de los tres agentes para la tercera referencia.

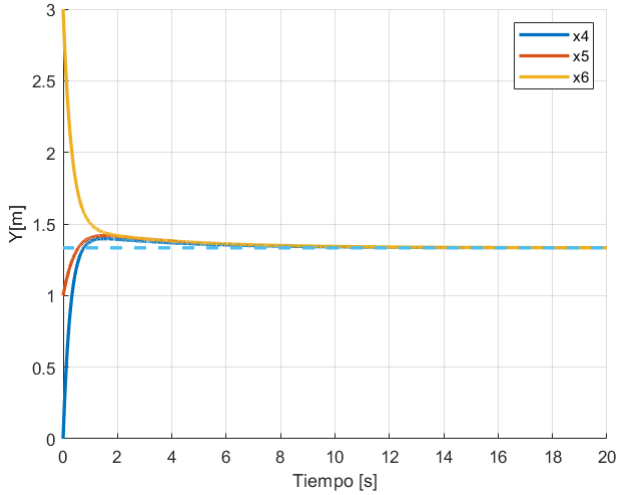


Figura 4.2.15: Formación del segundo grado de libertad de los tres agentes para la tercera referencia.

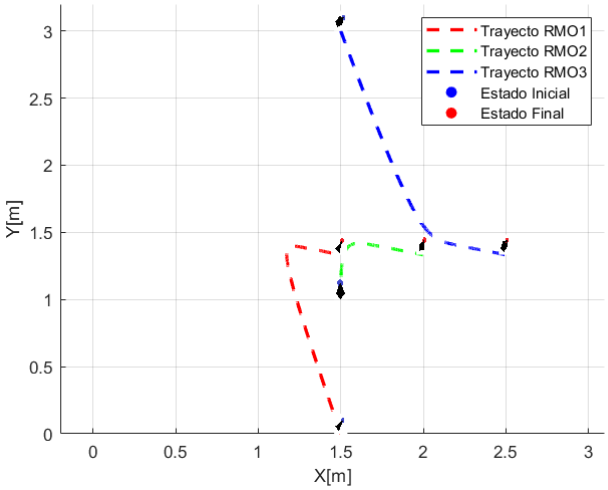


Figura 4.2.17: Trayectoria realizada por cada agente desde sus condiciones iniciales a la tercera formación.

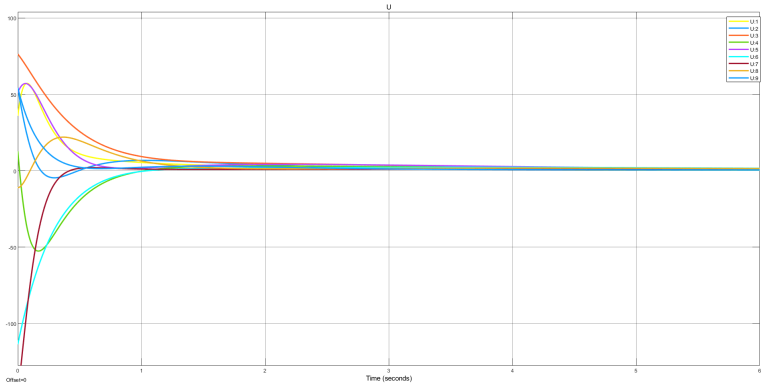


Figura 4.2.18: Señal de control de la tercera formación.

4.3. Validación experimental

A partir de los métodos de validación propuestos en el capítulo tres, en esta sección se evalúa el funcionamiento de los sistemas de posicionamiento mediante visión artificial y el sistema de comunicación que trasmite la señal de control a cada uno de los agentes, posteriormente se hace un análisis e interpretación de los resultados obtenidos.

4.3.1. Sistema de posicionamiento global de los agentes

Las pruebas del sistema de posicionamiento se realizaron sobre la plataforma experimental descrita en el capítulo 3, la cámara con la cual se obtuvieron las imágenes necesarias para la detección de los centroides mantuvo una frecuencia de captura de 511 cuadros por segundo

Para poder comparar los resultados obtenidos fue necesario obtener la posición y ángulos reales, lo cual se logró midiendo la distancia tanto en el eje X como en el Y de cada uno de los centroides de los agentes con respecto a los ejes del sistema de coordenadas inercial, así como la ubicación de los vértices máximos que indicaban su ángulo de rotación, tal como se propone en el capítulo 3.

Tras realizar las pruebas experimentales bajo las condiciones de iluminación controladas se calculó el error tanto en posición, como en ángulo de rotación. A partir de los resultados obtenidos se obtuvo el error máximo,mínimo y promedio esperado en el sistema de posicionamiento global.

Adicionalmente, en el anexo B del apéndice se muestra una tabla con la información recabada después de realizar las diez formaciones de los agentes. De dicho experimento se identificaron: un error máximo de posición de 0.0464m en la coordenada x y 0.00323 m en y, con un promedio obtenido en los diez experimentos de 0.0195 m y 0.0129 m respectivamente. En cuanto al ángulo de rotación, se identificó un error máximo de 3.46°, con un promedio en los diez experimentos de 1.96°, dicha información se puede apreciar en la Tabla 4.3.1.

	X (cm)	Y (cm)	Angulo (grados)
Error mínimo	0.01	0.03	0.065
Error máximo	4.64	3.23	3.464
Error promedio	1.95	1.29	1.960

Tabla 4.3.1: Resultados del método de validación

En las Figuras 4.3.1 se muestra la disposición de los tres agentes en la interfaz de visualización para dos de las diez formaciones propuestas. Con letras de color negro se indica el nombre de la forma geométrica (indicador de agente), relacionado con el número de bordes, la posición de su centroide y el ángulo de rotación tomando como referencia el primer cuadrante como inicio en el sentido opuesto a las manecillas del reloj.

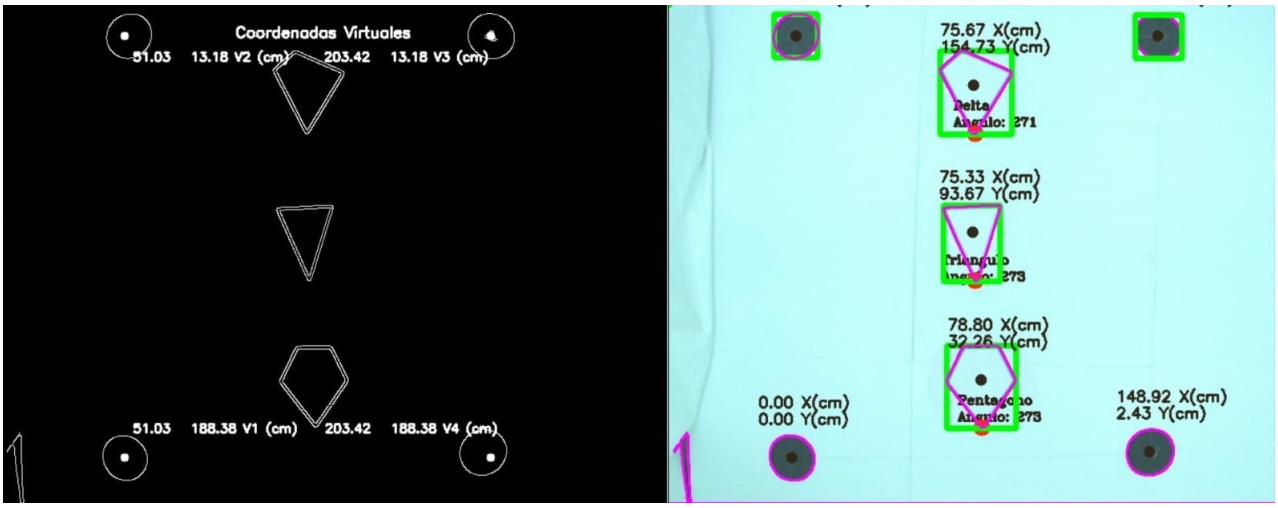


Figura 4.3.1: Primera prueba de reconocimiento del ángulo de rotación y la posición de tres agentes

Debido a las características físicas del sensor de imagen que posee la cámara Basler, el error mínimo esta limitado a 0.005 [m] esto se debe a la sensibilidad que el fabricante especifica para la cámara. Tomando en consideración el error promedio obtenido tras realizar las diez posiciones del experimento, los resultados se consideran suficientes para la función que el sistema de posicionamiento debe cumplir.

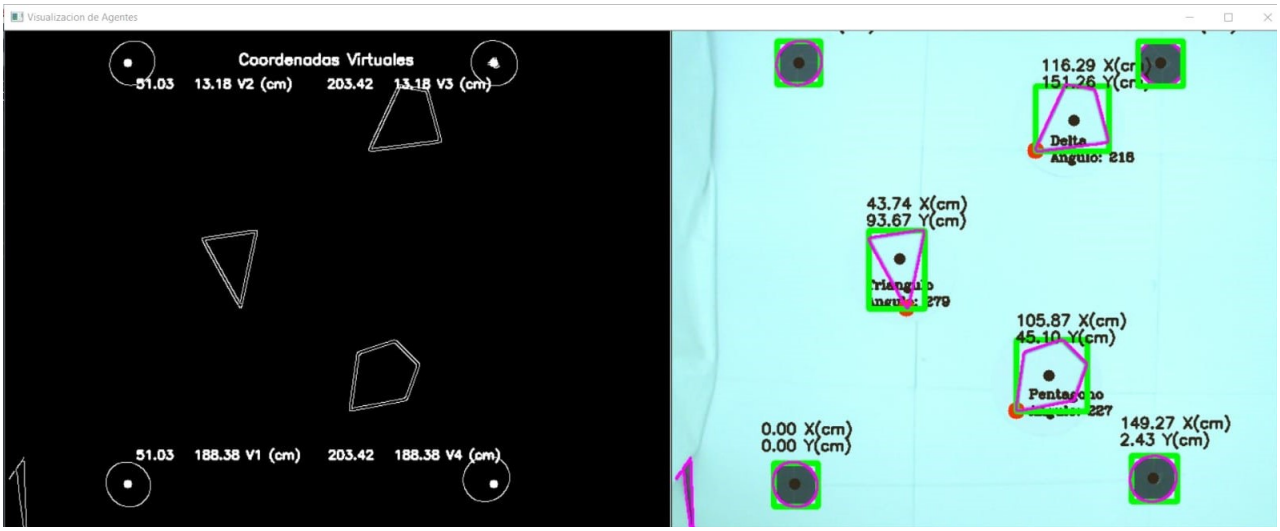


Figura 4.3.2: Segunda prueba de reconocimiento del ángulo de rotación y la posición de tres agentes

Por otro lado, las pruebas experimentales con trayectorias se realizaron sobre la misma plataforma por lo que las condiciones y marcos de referencia permanecieron idénticos. Se realizaron las trayectorias propuestas en el capítulo tres de forma intermitente para poder obtener los valores de posición reales de forma análoga a la prueba estática.

El procedimiento de pruebas consistió en colocar el agente seleccionado cerca del origen, medir su posición inicial, tanto con el sistema de visión, como con un medidor de distancia láser. Posteriormente se trasladó el agente a la siguiente ubicación (vértice 2) mientras el sistema de visión almacenaba todas las posiciones detectadas con un periodo de muestreo de 1.95 ms para luego presentar la trayectoria recorrida por el agente en la interfaz de usuario, finalmente se midió la posición final con el medidor láser y se registró el error de ambas posiciones para compararlas con las obtenidas por el sistema de visión.

En la figura 4.3.3 se observa el primer tramo de la trayectoria propuesta para el agente uno representada en la interfaz de usuario, mientras que en la figura 4.3.4 se aprecia lo correspondiente al agente dos.

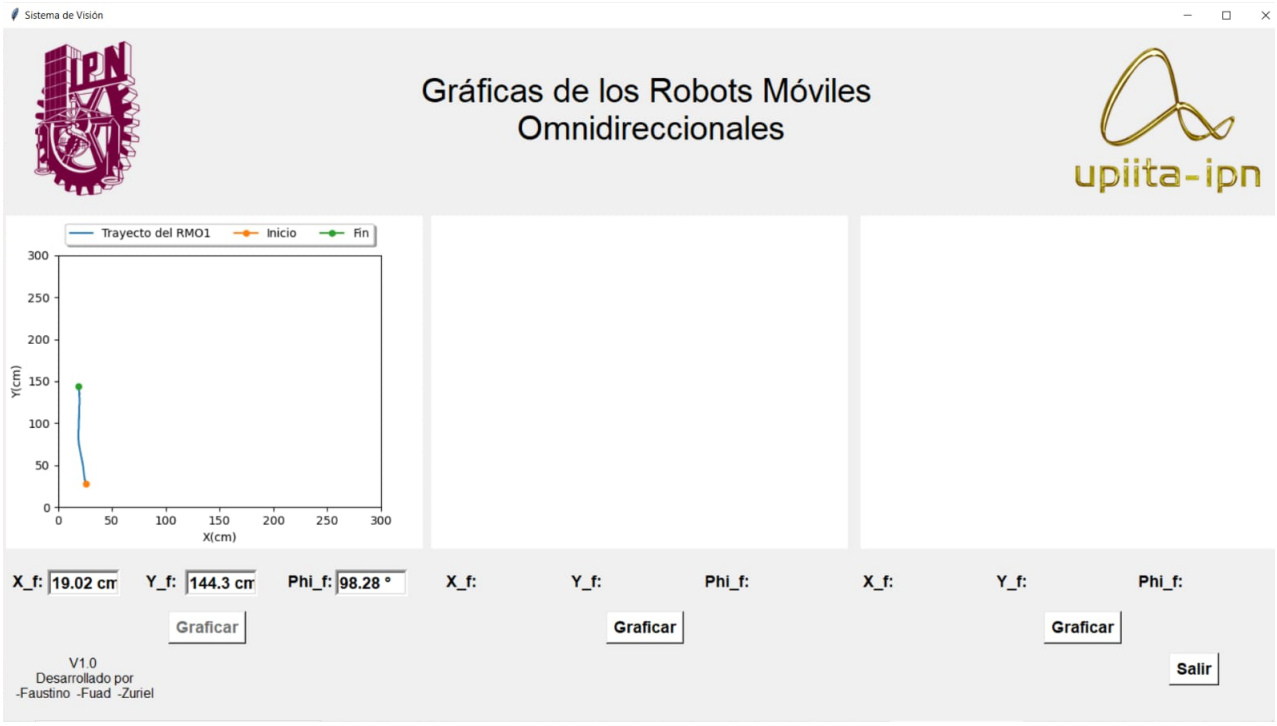


Figura 4.3.3: Primer tramo de la trayectoria propuesta para el agente uno representada en la interfaz de usuario.

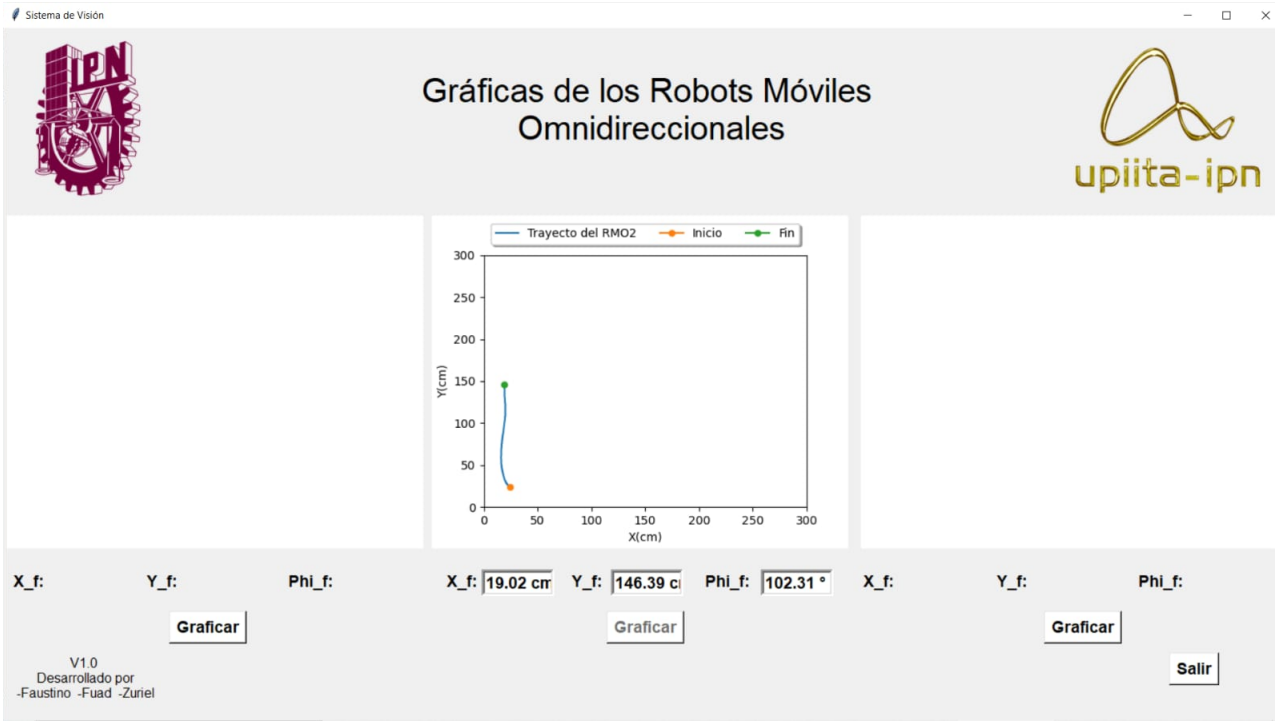


Figura 4.3.4: Primer tramo de la trayectoria propuesta para el agente dos representada en la interfaz de usuario.

Posterior a la realización del experimento dinámico propuesto, se llevo a cabo una trayectoria continua respetando el orden de ubicaciones de la prueba anterior con el propósito de visualizar la representación de un recorrido completo en la interfaz de usuario, el resultado se muestra en la figura 4.3.5.

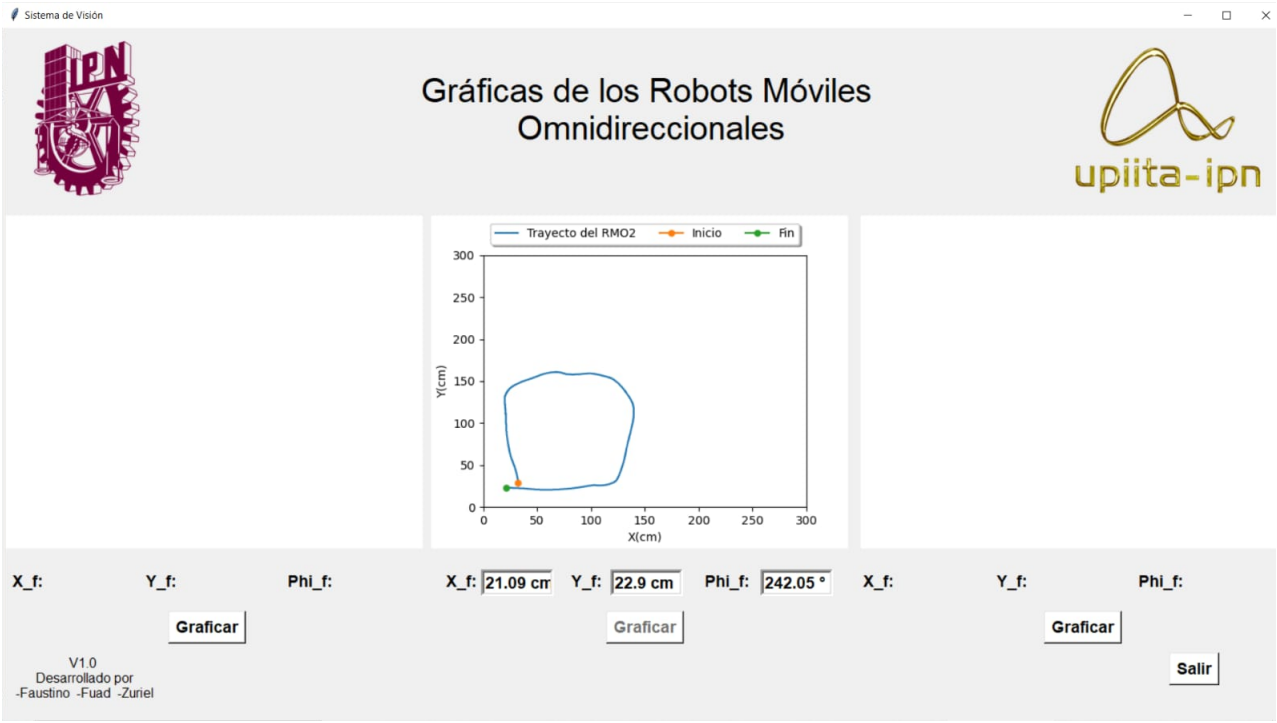


Figura 4.3.5: Trayectoria completa del segundo agente representada en la interfaz de usuario.

Al finalizar la prueba dinámica propuesta se obtuvo un error máximo de las variables de estado X_w, Y_w, ϕ_w de: (2.52 cm, 1.9 cm, 4.87°) respectivamente, mientras que el error mínimo fue de (0.59 cm, 0.23 cm, 0.10°) y el promedio de (1.92 cm, 1.10 cm, 1.77°), tal como se observa en la tabla 4.3.2.

	X (cm)	Y (cm)	Angulo (grados)
Error mínimo	0.59	0.23	0.10
Error máximo	2.52	1.9	4.87
Error promedio	1.92	1.10	1.77

Tabla 4.3.2: Resultados del método de validación dinámico

Los resultados de cada una de las trayectoria se presentan a detalle en el apéndice B.

4.3.2. Sistema de comunicación

En la tabla 4.3.3 podemos observar los resultados obtenidos al haber realizado la prueba de comunicación.

Punto	X[m]	Y[m]	Δ_t [ms]
A	0	0	9.4
B	1.5	0	9.6
C	3	0	11.75
D	3	1.5	11.95
E	3	3	13.4
F	1.5	3	13.85
G	0	3	12.7
H	0	1.5	14
I	1.5	1.5	14.2
PROMEDIO			12.31

Tabla 4.3.3: Retraso promedio en la comunicación alrededor del área de pruebas

Con base en la tabla anterior podemos concluir que el tiempo promedio de envío y recepción de datos en el área de trabajo fue de 12.31 ms.

Frecuencia de muestreo

Según el teorema de muestreo de Nyquist-Shannon [65], para poder digitalizar una señal analógica y transmitirla por un medio eléctrico a grandes distancias y poder recuperarla en el extremo distante con la máxima fidelidad posible, se requiere que la señal analógica sea muestreada al menos dos veces su frecuencia máxima. [REF]. En este caso, la frecuencia máxima estará dada por la frecuencia de respuesta del SMA ante una señal de control dada. Sabemos que la cámara trabaja con una frecuencia de captura de 511 fps (especificado anteriormente). Sea la frecuencia de muestreo f_s:

$$f_s = 511\text{fps}; \qquad T_\text{s} = \frac{1}{f_s} \tag{4.1}$$

$$T_s = 1.956\text{ms} \tag{4.2}$$

Si tomamos en cuenta que la frecuencia máxima a medir depende del tiempo de envío (el cual presenta un retraso promedio de 12.31 ms) recepción e interpretación (25 ms [46]) de la señal de control, la frecuencia máxima a medir es $f_{max} = 26.80Hz$ por lo que, según el teorema de muestreo de Nyquist la frecuencia de muestreo del sistema de posicionamiento es suficiente para garantizar la fidelidad de la señal medida.

$$511Hz > 2(26.80)Hz$$

Cabe mencionar que al determinar el tiempo de retraso en el procesamiento de imágenes realizado por la computadora central (el cual se ha calculado haciendo uso de la función time.time() como se puede apreciar en la figura 4.3.6.) se obtuvo un tiempo $t_{\text{retraso}} = 6\text{ns}$, al ser un proceso previo al envío de la información y poseer un tiempo de retraso significativamente menor, se tomó el tiempo de retraso del envío de la señal (12.31 ms).

```
In [3]: runfile('C:/Users/zurie/OneDrive - Instituto Politecnico Nacional/TT/TT1/
Programacion/prueba6.0/pruebatimeit.py', wdir='C:/Users/zurie/OneDrive - Instituto
Politecnico Nacional/TT/TT1/Programacion/prueba6.0')
5.999999999062311e-08
```

Figura 4.3.6: Tiempo de procesamiento de una imagen

4.4. Plataforma experimental

En esta sección se abordan los resultados obtenidos del diseño detallado en el capítulo tres respecto al sistema de soporte estructura y el sistema de iluminación de la cámara.

Las características del sistema estructural elaborado son:

- Estructura modular desmontable de fácil transporte y almacenamiento.
- Altura de la cámara variable de 4 a 4.5 m
- Área de trabajo de 3 x 3 m lisa color blanco mate resistente a las cargas de trabajo.
- Sistema de iluminación de fácil ensamble colocado en la parte superior de la estructura.
- Estructura resistente a lluvia y ráfagas de viento menores 215.7N
- Base con niveladores para ajuste dependiendo de la topografía del lugar.

A continuación se presenta la evidencia fotográfica del ensamble de la plataforma experimental y se describen las partes que la conforman.

Figura 4.4.1: Vista lateral de la plataforma experimental.

Figura 4.4.2: Vista del interior de la plataforma experimental.

Figura 4.4.3: Plataforma experimental sin lona.

En las figuras 4.4.1 y 4.4.3 se presenta una vista externa de la plataforma experimental con lona y sin ella respectivamente, en ellas se observa el recubrimiento anticorrosivo blanco que se le aplicó a la estructura para evitar corrosión al estar a la intemperie. Por otro lado, en la figura 4.4.2 se muestra el interior de la plataforma bajo condiciones controladas, ahí se observa la disposición de las placas de MDF blanco mate que conforman la base sobre la cual transitarán los agentes.

Finalmente, en la figura 4.4.4 se presenta un acercamiento a la parte de la estructura donde se encuentra el soporte de la cámara y el sistema de iluminación. Ahí se aprecia la distribución de las luminarias y el soporte con altura ajustable de la cámara, el cual permite un movimiento total de 50cm, dando una altura de trabajo entre 4 y 4.5m

Figura 4.4.4: Soporte estructural de la cámara y sistema de iluminación.

Capítulo 5

Conclusiones y perspectivas

En el presente trabajo de investigación, se presenta el desarrollo de una plataforma experimental de arquitectura de control abierta para la implementación de diversas estrategias de control cooperativo en sistemas multiagentes. Dicha plataforma está compuesta por un sistema de posicionamiento global a través de visión artificial, una unidad de procesamiento central, representaciones de tres robots móviles omnidireccionales, una estructura con condiciones de iluminación controladas y una interfaz de usuario que presenta las trayectorias recorridas por los agentes.

La estrategia implementada para el reconocimiento de los robots móviles omnidireccionales es mediante la detección y el conteo de contornos de los identificadores colocados sobre los agentes, dicha estrategia facilita el cálculo del ángulo de rotación puesto que se elimina la necesidad de implementar un indicador adicional como punto de referencia para obtener la matriz de rotación. La fiabilidad del sistema de posicionamiento se ve plasmada en los resultados experimentales donde se obtiene un error promedio de ($\pm$ 1.95 cm, $\pm$ 1.29 cm, 1.96° en (x_w, y_w, ϕ_w) respectivamente para un área de trabajo de 3x3m analizada desde una altura de 4.5m

El uso de un sistema centralizado permite eliminar la necesidad de establecer una red de comunicación entre los agentes simplificando el flujo de información, lo cual permite la adición de agentes adicionales sin comprometer la capacidad de procesamiento del sistema. Por otro lado, se realiza la simulación numérica de la estrategia de control propuesta para lograr el consenso de n agentes consiguiendo el objetivo en $\Delta t < 2s$ cuando $n = 3$.

Aunado a esto, el uso de una estructura con condiciones de iluminación controladas y una interfaz basada en software libre o de fácil acceso en el medio académico facilita la modificación y mejora de la plataforma y hace que la etapa de experimentación sea mas rápida, eficiente y confiable comparada con otras plataformas especializadas en sistemas multiagente del mercado que implementen un sistema de posicionamiento.

En resumen, con el trabajo realizado se logra implementar y validar en simulación numérica una estrategia de control cooperativo centralizado a través de un sistema de posicionamiento global con condiciones controladas para el consenso promedio de tres agentes, partiendo del modelo matemático del robot móvil (3,0) en su forma cinemática y definiendo su topología de comunicación mediante la teoría de grafos. Finalmente, se construye una interfaz gráfica y una estructura de control de arquitectura abierta que permite validar experimentalmente diversas estrategias de control cooperativo centralizado. Si bien, los resultados obtenidos son satisfactorios se pretenden realizar diversas mejoras al sistema de posicionamiento (con el objetivo de aumentar la fiabilidad del sistema) como la implementación de un soporte con estabilizador de tres ejes para evitar la distorsión en los resultados debido a vibraciones anormales o la adquisición de una lente con distancia focal y apertura de diafragma ajustables vía remota para una calibración de la lente mas rápida.

Bibliografía

[1] E. B. Steager, M. S. Sakar, D. H. Kim, V. Kumar, G. J. Pappas, and M. J. Kim, *Electrokinetic and optical control of bacterial microrobots*, J. Micromech. Microeng., vol. 21, no. 3, p. 035001, 2011.

[2] J. P. Sánchez Santana, "Control colaborativo disparado por eventos aplicado a sistemas multiagentes", tesis doctoral, CIDETEC. CDMX, México, 2018.

[3] D. Casanova, "Sistemas de control basados en red. Modelado y diseño de estructuras de control.", tesis doctoral, Univ. Politécnica de Valencia, 2005 [En línea]. Disponible en: http://hdl.handle.net/10251/1864 [Accedido: 02-Marzo-2020].

[4] R. Hernández, "Introducción a los sistemas de control: Conceptos, aplicaciones y simulación con Matlab", Pearson Educación, México, 2010 [En línea]. Disponible en: http://lcr.uns.edu.ar/fcr/images/Introduccion%20a%20Los%20Sistemas%20de%20Control.pdf.

[5] Y. Uny Cao, AlexS. Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7-27, 1997.

[6] H.G. Tanner and A. Kumar. Towards decentralization of multi-robot navigation functions. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4132-4137, April 2005.

[7] Hiroaki Yamaguchi. A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations. *Robotics and autonomous systems*, 43(4): 257-282, 2003.

[8] J. Liu and J. Wu, *Multi-agent robotic systems*. Boca Raton, Flor.: CRC Press, 2001.

[9] H. Gao *et al*, "*A new delay system approach to network-based control*", Automatica, vol. 44, pp. 39-52, 2008. doi: 10.1016/j.automatica.2007.04.020.

[10] H. Ishii and B. Francis, *"Limited Data Rate in Control Systems with Networks"*. Lecture Notes in Control and Information Sciences, vol 275. Springer, Berlin, Heidelberg, 2002. doi: 10.1007/3-540-45796-8.

[11] J. Salt *et al*, Sistemas de control basados en red modelado y diseño de estructuras de control", Revista Iberoamericana de automática e informática industrial, vol. 5, no. 3, pp. 5-20, 2008 [En línea]. Disponible en: https://n9.cl/gfl4.

[12] B. Moreno y J. Hernández, "Control centralizado y descentralizado de edificaciones mediante acristalamientos activos", Revista de investigación pensamiento matemático, vol. VII, no. 1, pp. 19-38, 2016 [En Línea]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=6000062.

[13] F. Rehman *et al*, *"Centralized control system design for underwater transportation using two hovering autonomous underwater vehicles (HAUVs)" IFAC Papers Online*, vol. 52, no. 11, pp. 13-18, 2019. doi: 10.1016/j.ifacol.2019.09.111.

[14] M. Rodríguez, "Teoría espectral de grafos en la formación de redes", tesis de maestría, Univ. Rosario, 2015.

[15] V. María, "Laplacianas de digrafos y topologías de Alexandrov", tesis de fin de grado, Univ. Sevilla, 2018.

[16] J. H. Godínez, M. V. Villa y J. A. Vásquez, "Problema de consenso en redes de agentes de primer orden con retardo en la comunicación", en *2014*, Congreso latinoamericano de control automático.

[17] Royal Society of Chemistry, "*Aluminium - Element information, properties and uses | Periodic Table*", Rsc.org, 2021. [En línea]. Disponible en: https://www.rsc.org/periodic-table/element/13/aluminium [Accedido: 13- Febrero - 2021].

[18] Pololu Inc., *"Polulu 5V, 2.5A Step-Down Voltage Regulator D24V22F5"*, 2021. [En línea]. Disponible en: https://www.pololu.com/product/2858 [Accedido: 13 - Febrero - 2021].

[19] LSI Computer Systems, Inc.,*" 32-Bit quadrature counter with serial interface"* 2021. [Online]. Available: https://lsicsi.com/datasheets/LS7366R.pdf. [Accessed: 14 - February - 2021].

[20] S. Campbell, *"Basics of the SPI Communication Protocol"*, Circuit Basics, 2021. [En línea]. Disponible en: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/. [Accedido: 14- Febrero- 2021].

[21] S. Campbell, *"Basics of the I2C Communication Protocol"*, Circuit Basics, 2021. [En línea]. Disponible: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accedido: 14- Febrero- 2021].

[22] Microchip Technology *"ATmega328 - 8-bit Microcontrollers"*, 2021. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega328 [Accessed: 14 - February - 2021].

[23] Maxon Group.com,*"ESCON 50/5 Servocontroladora"*, 2021. [En línea]. Disponible en: https://www.maxongroup.com/medias/sys_master/root/8834332327966/409510-ESCON-50-5-Referencia-del-Dispositivo-Es.pdf. [Accedido: 14 - Febrero - 2021].

[24] K. Fu *et al*, *"Robotics: Control, Sensing, Vision, and Intelligence"*, McGraw-Hill, USA: Purdue University, 1987. [Online]. Available: http://bit.ly/2LyHf89.

[25] Global Association for Vision information,*"Embedded vision in the robotics industry"*, 2020. [Online]. Available: https://www.visiononline.org/embedded-vision/robotics-industry.

[26] OpenCV: OpenCV-Python Tutorials", Docs.opencv.org, 2021. [Online]. Available: https://docs.opencv.org/master/d6/d00/tutorialpyroot.html. [Accessed: 07- January - 2021].

[27] M. Borja, "Sistema de posicionamiento con visión artificial para un brazo robótico articulado de seis grados mediante redes neuronales artificiales", 16th LACCEI International Multi-Conference for Engineering, Education, and Technology, 2018. doi: 10.18687/LACCEI2018.1.1.498.

[28] COGNEX, "Introducción a la visión artificial: Una guía para la automatización de procesos y mejoras de calidad", 2020. [En línea]. Disponible en: https://bit.ly/3hTAPN1.

[29] A. Ollero Baturone, "Robótica manipuladores y robots móviles", Barcelona, España: Marcombo, 2001. [En línea]. Disponible en: http://bit.ly/2PZacd8.

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

[31] R. B. Kairos, *"ROS Components"* [Online]. Available: https://bit.ly/39zahLc [Accessed: 29 - March - 2020].

[32] Direct industry, "Plataforma omnidireccional". [En línea]. Disponible en: https://bit.ly/3ayygvA [Accedido: 29-Mar-2020].

[33] QUANSER, "*Autonomous vehicles research studio*", 2020. [Online]. Available: https://www.quanser.com/products/autonomous-vehicles-research-studio/.

[34] N, Cross, "Árbol de objetivos, análisis de funciones", en *Método de Diseño, Estrategias para el Diseño de Productos*, Reino Unido, 2001. [En línea]. Disponible en: https://bit.ly/38mDa00 [Accedido: 17 - Mayo - 2020].

[35] UPIITA, "Ingeniería Mecatrónica". [En línea]. Disponible en: https://www.upiita.ipn.mx/oferta-educativa/mecatronica. [Accedido: 03 - Marzo - 2020].

[36] E. Morales y L. Sucar, "Los robots del futuro y su importancia para México" en *INAOE*, 2009, pp.1-11. [En linea]. Disponible en: *https : //ccc.inaoep.mx/ emorales/Papers/*2009/*eduardo.pdf*.

[37] Y. Jia *et al.*, *"Three-Dimensional Leaderless flocking control of large-scale small unmanned aerial vehicles"*, *IFAC Papers OnLine*, vol. 50, no. 1, pp. 6208-6213, 2017. doi: 10.1016/J.IFACOL.2017.08.1016.

[38] K. Das and D. Ghose, *"Broadcast Control Mechanism for Positional Consensus in Multiagent Systems"*, *IEEE Transactions on control systems technology*, vol. 23, no. 55, pp. 1807-1826, 2015. doi: 10.1109/TCST.2015.2388732.

[39] C. Nowzari *et al.*, "*Event-triggered communication and control of networked systems for multi-agent consensus*", *Automatica*, vol. 105, pp. 1-27, 2019. doi: 10.1016/j.automatica.2019.03.009.

[40] I. Qaisar *et al.*, "*Exponential Stability for Event-triggered Consensus Control of Heterogeneous Multi-Agent Systems*", *The 31th Chinese Control and Decision Conference*, pp. 2373-2378, 2019. doi: 10.1109/CCDC.2019.8832767.

[41] J. Deng *et al.*, "*Distributed Adaptive Time-Varying Formation Tracking Control for General Linear Multi-Agent Systems Based on Event-Triggered Strategy*", *IEEE Access*, vol. XX, 2017. doi: 10.1109/ACCESS.2020.2966042.

[42] P. Yu *et al.*,"*Distributed Event-Triggered Communication and Control of Linear Multi-Agent Systems Under Tactile Communication*", *IEEE Transactions on Automatic Control*, 2018. doi: 10.1109/TAC.2018.2805682.

[43] F. Reyes, "Robótica-control de robots manipuladores". Alfaomega Grupo Editor, 2011.

[44] C. Nowzari *et al.*, "'*Event-Triggered Communication and Control for Multi-Agent Average Consensus*", 2016. [Online]. Available: https://bit.ly/3q4LFTv.

[45] J. Liang *et al.*, "*Event-Triggered Consensus Control for Linear Multi-Agent Systems*", *IEEE Access*, 2019. doi: 10.1109/ACCESS.2019.2944946.

[46] S. Benitez y J. De La Cruz, "Robot móvil (3,0) de arquitectura de control abierta considerando un sistema de odometría para su regulación en el espacio de operación", tesis de fin de grado, Univ. Instituto Politécnico Nacional, CDMX, México, 2017.

[47] S. E. Benitez García, J. L. de la Cruz Osorio y M. G. Villarreal Cervantez, "Robot móvil 3.0 una evaluación de rendimiento", *Pistas Educativas*, no. 128, pp. 230-247, México, 2018.

[48] Villarreal Cervantes, M. G. y Pantoja García, J. S., "Análisis comparativo entre un control heurístico y un PID para un sistema mecatrónico". s.l.:3th International Supercomputing Conference, 2012.

[49] Villarreal Cervantes, M. G. , "*Lecture notes: Control strategy for the consensus of a (3,0) mobile robot network"*, Departamento de posgrado en mecatrónica, Instituto politécnico nacional, 2021.

[50] GeeksforGeeks, "atan2() function in Python - GeeksforGeeks", 2021. [Online]. Available: https://www.geeksforgeeks.org/atan2-function-python/. [Accessed: 23- Feb- 2021].

[51] Vicon, "*Nexus 2.10 and capture.U 1.1 now available*". [Online]. Available: https://www.vicon.com/.

[52] OptiTrak, "*Motive: Optical motion capture software*", 2020. [Online]. Available: https://optitrack.com/software/.

[53] M. Claverol, E. Simó y M. Zaragoza, "Matemática Discreta teoría de grafos", Departamento Matemática Aplicada IV, EPSEVG - UPC, 2012. [En linea]. Disponible en: http://bit.ly/2xn6xPK.

[54] E. Avila Martinez, "Consenso en arreglos maestro-seguidores de agentes inerciales", tesis de maestría, Univ. Instituto Potosino de Investigación Científica y Tecnológica, A.C., 2014. [En línea]. Disponible en: https://bit.ly/2XmAU2w [Accedido: 03 - Marzo - 2020].

[55] S. Mallick., "*Why does OpenCV use BGR color format ? | Learn OpenCV*", Learn OpenCV | OpenCV, PyTorch, Keras, Tensorflow examples and tutorials, 2021. [Online]. Available: http://bit.ly/2LwNjyh [Accessed: 08 - January - 2021].

[56] Alojamientos.us.es,"Filtros", 2021. [En línea]. Disponible en: http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf [Accedido: 04 - Enero - 2021].

[57] Python OpenCV, "*Converting an image to gray scale - techtutorialsx*", techtutorialsx, 2021. [Online]. Available: http://bit.ly/3s08n0V [Accessed: 04 - January- 2021].

[58] Canny Edge Detection ,"*OpenCV-Python Tutorials 1 documentation*", Opencv-python-tutroals.readthedocs.io, 2021. [Online]. Available: http://bit.ly/3bk5Jgk [Accessed: 04 - January- 2021].

[59] Grupo.us.es., Trabajo de momentos, 2020. [En línea] Disponible en: https://bit.ly/3piM30Y [Accedido: 11 - Noviembre - 2020].

[60] "*Wind and weather statistic Ciudad de México Aeropuerto*", Windfinder.com, 2020. [Online]. Available: *https : //es.windfinder.com/windstatistics/mexico_city*. [Accessed: 03 - January - 2021].

[61] Asce.org, "*Codes and Standards | ASCE*", 2021. [Online]. Available: http://bit.ly/3bfY5Up [Accessed: 03 - January - 2021].

[62] Cross, N, "*Engineering design methods*", 2005. 3rd ed. UK: Jhon Wiley.

[63] Faires, V. M. "Diseño de elementos de máquinas". México, Editorial Limusa, 1995. 4a reimpresión.

[64] F. P. Beer, E. R. Johnston, J. T. DeWolf, and D. F. Mazurek, "*Mechanics of materials*". New York, NY: McGraw-Hill Education, 2020.

[65] P. Colarruso and E. Neil Lewis, "*Nyquist Theorem - an overview | ScienceDirect Topics*", Sciencedirect.com, 2021. [Online]. Available: https://www.sciencedirect.com/topics/engineering/nyquist-theorem [Accessed: 01 - June - 2021].

[66] Miguel G Villarreal-Cervantes. "Modelo cinemático y dinámico del robot móvil (3,0)."Technical report, CIDETEC-IPN, 2015.

Apéndice A

Programa implementado para el sistema de visión

```
1
2
3
4  """
5  Instituto Polit cnico Nacional
6  Unidad Profesional Interdisciplinaria en Ingenier a y Tecnolog as Avanzadas
7  CIDETEC
8
9  C digo desarrollado por :
10
11     - Garc a Mercado Faustino
12     - L pez Sayeg Fuad Alejandro
13     - Ugalde S nchez Zuriel
14
15 """
16
17 # %%librerias
18
19 import cv2
20 import numpy as np
21 import math
22 from pypylon import pylon
23 from math import atan2, degrees
24 from collections import Counter
25 import tkinter as tk
26 from tkinter import Label, Button
27 from tkinter import font as tkFont
28 from PIL import ImageTk, Image
29 from pandas import DataFrame
30 import matplotlib.pyplot as plt
31 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
32
33
34 # %%Variables globales y listas
35
36 Angulostrian = []
37 TriX = []
38 TriY = []
39
40 Angulospenta = []
41 CuadX =[]
42 CuadY = []
43
44 Angulosdelta = []
45 PentaX = []
46 PentaY = []
47
48 Cxlist = []
49 Cylist = []
50
51 Vxmin = []
52 Vxmax = []
53 Vymin = []
54 Vymax = []
55
56 Distancia = []
57 # %% Par metros de inicializacion de la camara Basler
58
59 Flag = True
```

```
60
61  #Crea el objeto camara
62  camera = pylon.InstantCamera(pylon.TlFactory.GetInstance().CreateFirstDevice())
63
64
65
66  #Graba de forma continua(video) con retraso minimo
67  camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)
68  converter = pylon.ImageFormatConverter()
69
70  camera.AcquisitionFrameRateEnable.SetValue(True);
71  camera.AcquisitionFrameRate.SetValue(400.0)
72
73  # Convierte de formato OpenCV a formato BGR
74  converter.OutputPixelFormat = pylon.PixelType_BGR8packed
75  converter.OutputBitAlignment = pylon.OutputBitAlignment_MsbAligned
76
77
78  def empty(a):
79      pass
80
81  cv2.namedWindow("Parametros de Visualizacion")
82  cv2.resizeWindow("Parametros de Visualizacion",640,150)
83  cv2.createTrackbar("Threshold1","Parametros de Visualizacion",23,255,empty)#23
84  cv2.createTrackbar("Threshold2","Parametros de Visualizacion",20,255,empty)#20
85  cv2.createTrackbar("Area","Parametros de Visualizacion",5000,30000,empty)#5000-30000
86
87  # %%Funciones necesarias para el sistema de visi n
88
89  def ListCreator(imgDil,imgContour):
90       contours, hierarchy = cv2.findContours(imgDil, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
91       for cnt in contours:
92          area = cv2.contourArea(cnt)
93          areaMin = cv2.getTrackbarPos("Area", "Parametros de Visualizacion")
94          if area> areaMin:
95              M = cv2.moments(cnt)   # compute the center of the contour
96              cX = int(M["m10"] / M["m00"])
97              cY = int(M["m01"] / M["m00"])
98              Cxlist.append(cX)
99              Cylist.append(cY)
100      return(Cxlist,Cylist)
101
102 def getPreview(imgDil,imgContour):
103
104     contours, hierarchy = cv2.findContours(imgDil, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
105
106     for cnt in contours:
107         area = cv2.contourArea(cnt)
108         areaMin = cv2.getTrackbarPos("Area", "Parametros de Visualizacion")
109         if area> areaMin:
110             cv2.drawContours(imgContour, contours, -1, (255, 0, 255), 2)
111             M = cv2.moments(cnt)   # compute the center of the contour
112             cX = int(M["m10"] / M["m00"])
113             cY = int(M["m01"] / M["m00"])
114             cv2.circle(imgContour, (cX, cY), 7, (255, 255, 255), -1)
115             peri = cv2.arcLength(cnt, True)
116             approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)
117             x_, y_, w, h = cv2.boundingRect(approx)
118             cv2.rectangle(imgContour, (x_, y_), (x_ + w, y_ + h), (0, 255, 0), 5)#Caja verde
119
120
121
122             #cv2.putText(imgContour,  str(int(cX))  + (" Xc(mm) ") +   str(int(cY))  +(" Yc(mm) "), (cX - 160,
                    cY - 10), cv2.FONT_HERSHEY_SIMPLEX, .6, (255, 255, 255), 2)
123             cv2.putText(imgDil,  "Preview", (280, 50), cv2.FONT_HERSHEY_SIMPLEX, .6, (255, 255, 255), 2)
124
125 def Tri(cXcm,cYcm):
126     TriX.append(cXcm)
127     TriY.append(cYcm)
128     return(TriX,TriY)
129
130 def Cuad(cXcm,cYcm):
131     CuadX.append(cXcm)
132     CuadY.append(cYcm)
133     return(CuadX,CuadY)
134
135 def Penta(cXcm,cYcm):
136     PentaX.append(cXcm)
137     PentaY.append(cYcm)
138     return(PentaX,PentaY)
139
140 def angulo(centro, Indicador, rotation=0, clockwise=False):
141     angle = degrees(atan2(Indicador[1] - centro[1], Indicador[0] - centro[0])) - rotation
142     if not clockwise:
143         angle = -angle
144     angle = angle % 360
145     if (angle > 358 and angle <= 360):
146         angle = 0
147     elif (angle > 0 and angle <= 2):
```

```
        angle = 0
    return angle

def getContours(imgDil,imgContour):

    contours, hierarchy = cv2.findContours(imgDil, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    distV1V4 = abs(Vymax[0]-Vymin[0])       #Distancia virtual V1 a V4
    distV4V3 = abs(Vxmax[0]-Vxmin[0])        #Distancia virtual V4 a V3

    distanciaRealV1_V4 = 1.753         #Distancia f sica real V1 a V4
    distanciaRealV4_V3 = 1.523          #Distancia f sica real V4 a V3

    razonX = (distanciaRealV1_V4)*100/(distV1V4)
    razonY = (distanciaRealV4_V3)*100/(distV4V3)


    for cnt in contours:
        area = cv2.contourArea(cnt)
        areaMin = cv2.getTrackbarPos("Area", "Parametros de Visualizacion")
        if area> areaMin:
            cv2.drawContours(imgContour, contours, -1, (255, 0, 255), 2)

            M = cv2.moments(cnt)   # compute the center of the contour
            cX = int(M["m10"] / M["m00"])
            cY= int(M["m01"] / M["m00"])
            cXcm = abs(cX - Cxmin)*razonX
            cYcm = abs(cY - Cymax)*razonY
            cv2.circle(imgContour, (cX, cY), 7, (28, 40, 51), -1)

            approx = cv2.approxPolyDP(cnt, 7, True)
            lados = len(approx)
            if lados == 8:
                approx[:,:]=[0,0]
            x_, y_, w, h = cv2.boundingRect(approx)


            if  (lados == 3):
                   cv2.putText(imgContour, ("Triangulo") , (x_ + w - 70, y_ + 70), cv2.FONT_HERSHEY_COMPLEX,
                        0.5,
                         (28, 40, 51), 2)
                   [TriX,TriY] = Tri(cXcm,cYcm)
                   vertices = approx.reshape((approx.shape[0], -1), order='F')   #Reshape cambia la forma de un
                        array
                   print('Los vertices son', vertices)
                   centro = [cX,cY]
                   for i in range(3):
                       D = np.sqrt((cY-vertices[i,1])**2 +  (cX-vertices[i,0])**2) #Distancia euclidianda de
                            centroide a vertices
                       Distancia.append(D)

                   VerticeMax = np.argmax(Distancia)    #Determina el vertice mas alejado del centroide
                   Indicador = [vertices[VerticeMax,0],vertices[VerticeMax,1]]

                   if len(Distancia) != 3:
                       pass
                   elif len(Distancia) ==3:
                           angle3 = angulo(centro,Indicador)
                           Angulostrian.append(angle3)
                           anguloviejotrian = Angulostrian[0]
                           angulonuevotrian = Angulostrian[-1]

                   if (abs(anguloviejotrian-angulonuevotrian)<4):
                       angulonuevotrian = anguloviejotrian
                   else:
                       Angulostrian[0] = angulonuevotrian

                   cv2.putText(imgContour, "Angulo: " + str(int(angulonuevotrian)), (x_ + w -70, y_ + 90), cv2.
                        FONT_HERSHEY_COMPLEX, 0.5, (28, 40, 51), 2)
                   cv2.circle(imgContour, (Indicador[0], Indicador[1]), 10, (0, 50, 250), -1)
                   Distancia.clear()

            if  (lados == 5):
                   cv2.putText(imgContour, ("Pentagono") , (x_ + w -70, y_ + 70), cv2.FONT_HERSHEY_COMPLEX,
                        0.5,
                         (28, 40, 51), 2)
                   [PentaX,PentaY] = Penta(cXcm,cYcm)
                   vertices = approx.reshape((approx.shape[0], -1), order='F')   #Reshape cambia la forma de un
                        array
                   centro = [cX,cY]
                   for i in range(5):
                       D = np.sqrt((cY-vertices[i,1])**2 +  (cX-vertices[i,0])**2) #Distancia euclidianda de
                            centroide a vertices
                       Distancia.append(D)

                   VerticeMax = np.argmax(Distancia)#Determina el vertice mas alejado del centroide
                   Indicador = [vertices[VerticeMax,0],vertices[VerticeMax,1]]
                   #print(Distancia)
                   if len(Distancia) != 5:
                       pass
```

```
                elif len(Distancia) == 5:
                        angle5 = angulo(centro,Indicador)
                        Angulospenta.append(angle5)
                        anguloviejopenta = Angulospenta[0]
                        angulonuevopenta = Angulospenta[-1]

                if (abs(anguloviejopenta-angulonuevopenta)<4):
                    angulonuevopenta = anguloviejopenta
                else:
                    Angulospenta[0] = angulonuevopenta

                cv2.putText(imgContour, "Angulo: " + str(int(angulonuevopenta)), (x_ + w -70, y_ + 90), cv2.
                    FONT_HERSHEY_COMPLEX, 0.5, (28, 40, 51), 2)
                cv2.circle(imgContour, (Indicador[0], Indicador[1]), 10, (0, 50, 250), -1)
                Distancia.clear()

            if  (lados == 4):
                cv2.putText(imgContour, ("Delta") , (x_ + w -70, y_ + 70), cv2.FONT_HERSHEY_COMPLEX, 0.5,
                     (28, 40, 51), 2)
                [CuadX,CuadY] = Cuad(cXcm,cYcm)
                vertices = approx.reshape((approx.shape[0], -1), order='F')   #Reshape cambia la forma de un
                     array
                centro = [cX,cY]
                for i in range(4):
                    D = np.sqrt((cY-vertices[i,1])**2 +  (cX-vertices[i,0])**2) #Distancia euclidianda de
                         centroide a vertices
                    Distancia.append(D)

                VerticeMax = np.argmax(Distancia)#Determina el vertice mas alejado del centroide
                Indicador = [vertices[VerticeMax,0],vertices[VerticeMax,1]]
               # print(Distancia)
                if len(Distancia) != 4:
                    pass
                elif len(Distancia) == 4:
                        angle4 = angulo(centro,Indicador)
                        Angulosdelta.append(angle4)
                        anguloviejodelta = Angulosdelta[0]
                        angulonuevodelta = Angulosdelta[-1]

                if (abs(anguloviejodelta-angulonuevodelta)<4):
                    angulonuevodelta = anguloviejodelta
                else:
                    Angulosdelta[0] = angulonuevodelta

                cv2.putText(imgContour, "Angulo: " + str(int(angulonuevodelta)), (x_ + w -70, y_ + 90), cv2.
                    FONT_HERSHEY_COMPLEX, 0.5,                          (28, 40, 51), 2)
                cv2.circle(imgContour, (Indicador[0], Indicador[1]), 10, (0, 50, 250), -1)
                Distancia.clear()

            else:
                #cv2.putText(imgContour, "Points: " + str(lados), (x_ + w -100, y_ + 20), cv2.
                     FONT_HERSHEY_COMPLEX, 0.4,
                      #(231, 76, 60), 2)
                print(len(approx))

            cv2.rectangle(imgContour, (x_, y_), (x_ + w, y_ + h), (0, 255, 0), 5)

            #cv2.putText(imgContour, "Area: " + str(int(area)), (x_ + w - 70, y_ + 110), cv2.
                 FONT_HERSHEY_COMPLEX, 0.5,
                      #(0, 255, 0), 2)


            #Inversion de ejes
            #tempcy = cYcm
            #cYcm = np.sqrt(cXcm**2)
            #cXcm = np.sqrt(tempcy**2)
            #Retirar de ser necesario

            cv2.putText(imgContour,  "{:.2f}".format(float(cXcm)) + (" X(cm) ") , (int(cX)-40, int(cY)-60), cv2
                 .FONT_HERSHEY_SIMPLEX, .6, (28, 40, 51), 2)
            cv2.putText(imgContour,  "{:.2f}".format(float(cYcm)) + (" Y(cm) ") , (int(cX)-40, int(cY)-40), cv2
                 .FONT_HERSHEY_SIMPLEX, .6, (28, 40, 51), 2)

            cv2.putText(imgDil, "{:.2f}".format(float(Cxmin*razonX)) + "   " + "{:.2f}".format(float(Cymin*
                 razonY)) + " V2 (cm)" , (Cxmin+10, Cymin+30), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255, 255, 0), 2)
            cv2.circle(imgDil, (Cxmin, Cymin), 5, (255, 255, 255), -1)
            cv2.putText(imgDil, "{:.2f}".format(float(Cxmax*razonX)) + "   " + "{:.2f}".format(float(Cymin*
                 razonY)) + " V3 (cm)" , (Cxmax-200, Cymin+30), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255, 255, 0), 2)
            cv2.circle(imgDil, (Cxmax, Cymin), 5, (255, 255, 255), -1)
            cv2.putText(imgDil, "{:.2f}".format(float(Cxmax*razonX)) + "   " + "{:.2f}".format(float(Cymax*
                 razonY)) + " V4 (cm)" , (Cxmin+240, Cymax-30), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255, 255, 0), 2)
            cv2.circle(imgDil, (Cxmax, Cymax), 5, (255, 255, 255), -1)
            cv2.putText(imgDil, "{:.2f}".format(float(Cxmin*razonX)) + "   " + "{:.2f}".format(float(Cymax*
                 razonY)) + " V1 (cm)" , (Cxmin+10, Cymax-30), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255, 255, 0), 2)
            cv2.circle(imgDil, (Cxmin, Cymax), 5, (255, 255, 255), -1)

            cv2.putText(imgDil,  "Coordenadas Virtuales", (280, 50-10), cv2.FONT_HERSHEY_SIMPLEX, .6, (255,
                 255, 255), 2)
```

```
306 |
307 | while camera.IsGrabbing():
308 |     grabResult = camera.RetrieveResult(5000, pylon.TimeoutHandling_ThrowException)
309 |     if grabResult.GrabSucceeded():             # Accesar a la informacion de la imagen
310 |             image = converter.Convert(grabResult)
311 |             img = image.GetArray()
312 |             imgContour = img.copy()
313 |     imgBlur = cv2.GaussianBlur(img, (5, 5), 1)
314 |     imgGray = cv2.cvtColor(imgBlur, cv2.COLOR_BGR2GRAY)
315 |     threshold1 = cv2.getTrackbarPos("Threshold1","Parametros de Visualizacion")
316 |     threshold2 = cv2.getTrackbarPos("Threshold2","Parametros de Visualizacion")
317 |     imgDil = cv2.Canny(imgGray,threshold1,threshold2)
318 |
319 |     if  Flag == True:
320 |         if cv2.waitKey(1) &0xFF == ord('p'):
321 |             Flag = False
322 |             [Cxlist,Cylist]= ListCreator(imgDil, imgContour)
323 |             Cxmin = np.amin(Cxlist)
324 |             Cxmax = np.amax(Cxlist)
325 |             Cymin = np.amin(Cylist)
326 |             Cymax = np.amax(Cylist)
327 |             Vxmin.append(Cxmin)
328 |             Vxmax.append(Cxmax)
329 |             Vymin.append(Cymin)
330 |             Vymax.append(Cymax)
331 |             getContours(imgDil,imgContour)
332 |         else:
333 |             getPreview(imgDil,imgContour)
334 |     else:
335 |         getContours(imgDil,imgContour)
336 |         v1 = np.array([Vxmin,Vymin])
337 |         v2 = np.array([Vxmax,Vymin])
338 |         v3 = np.array([Vxmax,Vymax])
339 |         v4 = np.array([Vxmin,Vymax])
340 |
341 |         MatrizTriangulo = np.array([TriX,TriY])
342 |         if MatrizTriangulo.size != 0:
343 |             CentroideTriangulo = np.array([TriX[-1],TriY[-1]])
344 |
345 |         MatrizCuadrado = np.array([CuadX,CuadY])
346 |         if MatrizCuadrado.size != 0:
347 |             CentroideCuadrado = np.array([CuadX[-1],CuadY[-1]])
348 |
349 |         MatrizPentagono = np.array([PentaX,PentaY])
350 |         if MatrizPentagono.size != 0:
351 |             CentroidePentagono = np.array([PentaX[-1],PentaY[-1]])
352 |
353 |
354 |     grayImageBGRspace = cv2.cvtColor(imgDil,cv2.COLOR_GRAY2BGR)
355 |     horizontalAppendedIGrayImg = np.hstack((grayImageBGRspace,imgContour))
356 |     cv2.imshow('Visualizacion de Agentes', horizontalAppendedIGrayImg)
357 |     if cv2.waitKey(1) &0xFF == ord('q'):
358 |         cv2.destroyAllWindows()
359 |         break
```

Apéndice B

Resultados obtenidos en las pruebas de validación del sistema de visión

Posición	Agente	X Real (cm)	Y Real (cm)	Xm Real (cm)	Ym Real (cm)	X (cm)	Y (cm)	Ángulo Real	Ángulo	Error ángulo (grados)
1	Triángulo	47.60	45.50	48.00	26.10	46.82	43.53	88.82	86.00	2.8188
	Pentágono	261.00	131.00	260.80	112.30	264.42	128.92	90.61	93.00	2.3872
	Trapecio	145.70	236.20	143.20	217.30	148.49	236.91	97.54	101.00	3.4649
2	Triángulo	42.60	210.60	28.20	197.70	42.61	212.23	138.14	139.00	0.8550
	Pentágono	247.50	215.80	259.80	201.30	250.29	216.54	49.69	47.00	2.6929
	Trapecio	149.90	37.70	150.00	56.90	149.26	34.47	270.30	270.00	0.2984
3	Triángulo	34.30	125.10	47.30	139.70	31.83	124.97	311.68	309.00	2.6822
	Pentágono	245.60	125.40	232.10	138.50	249.75	123.89	224.14	227.00	2.8615
	Trapecio	145.90	125.50	57.70	125.40	146.72	124.43	179.94	0.00	0.0650
4	Triángulo	145.60	129.10	146.30	109.50	146.28	127.82	87.95	90.00	2.0454
	Pentágono	258.80	229.50	247.70	210.50	261.66	229.75	120.29	122.00	1.7061
	Trapecio	30.10	35.80	19.20	52.20	28.15	34.71	236.39	235.00	1.3906
5	Triángulo	145.50	130.30	160.30	142.60	146.28	128.92	320.27	317.00	3.2707
	Pentágono	258.90	33.00	240.20	31.80	261.10	31.40	176.33	173.00	3.3283
	Trapecio	32.50	223.00	33.10	242.80	33.12	224.79	271.74	271.00	0.7357
6	Triángulo	145.70	129.30	144.70	148.60	146.53	128.37	267.03	266.00	1.0340
	Pentágono	147.70	23.40	147.80	4.10	147.63	20.94	89.70	93.00	3.2969
	Trapecio	147.90	225.80	150.30	206.40	149.28	227.54	82.95	86.00	3.0523
7	Triángulo	36.80	103.00	52.20	91.10	34.70	102.83	37.69	39.00	1.3058
	Pentágono	82.60	222.30	93.30	207.30	83.18	224.36	54.50	55.00	0.5015
	Trapecio	256.70	93.00	268.70	78.40	259.45	91.83	50.58	49.00	1.5826
8	Triángulo	27.00	129.20	7.70	129.00	24.89	129.17	179.41	181.00	1.5937
	Pentágono	245.10	42.40	226.45	41.30	248.38	40.30	176.62	174.00	2.6245
	Trapecio	244.65	215.40	225.70	216.00	248.38	215.28	181.81	185.00	3.1865
9	Triángulo	42.80	42.20	42.20	61.60	40.38	40.85	268.23	266.00	2.2285
	Pentágono	253.10	39.10	253.20	58.00	255.70	36.98	270.30	273.00	2.6969
	Trapecio	142.10	231.30	142.40	212.20	144.38	232.39	89.10	86.00	3.1001
10	Triángulo	22.10	221.20	41.50	221.10	23.23	221.35	0.30	0.00	0.2953
	Pentágono	27.10	29.90	8.40	29.40	25.45	28.70	178.47	180.00	1.5316
	Trapecio	262.00	129.50	281.20	129.60	266.64	128.06	359.70	0.00	0.2984
								Error promedio		1.96
								Error mínimo		0.0650
								Error máximo		3.4649

Figura B.0.1: Resultados de las pruebas de validación del sistema de visión (estático)

Error posición en X (cm)	Error posición en Y (cm)	
0.78	1.97	
3.42	2.08	
2.79	0.71	
0.01	1.63	
2.79	0.74	
0.64	3.23	
2.47	0.13	
4.15	1.51	
0.82	1.07	
0.68	1.28	
2.86	0.25	
1.95	1.09	
0.78	1.38	
2.2	1.6	
0.62	1.79	
0.83	0.93	
0.07	2.46	
1.38	1.74	
2.1	0.17	
0.58	2.06	
2.75	1.17	
2.11	0.03	
3.28	2.1	
3.73	0.12	
2.42	1.35	
2.6	2.12	
2.28	1.09	
1.13	0.15	
1.65	1.2	
4.64	1.44	
1.95	1.29	Promedio
0.01	0.03	Mínimo
4.64	3.23	Máximo

Figura B.0.2: Resultados de las pruebas de validación del sistema de visión (estático)

Posición	Agente	X Real (cm)	Y Real (cm)	Xm Real (cm)	Ym Real (cm)	X (cm)	Y (cm)	Ángulo Real	Ángulo	Error ángulo (grados)
1	Inicial	28.30	28.30	44.30	40.10	26.30	27.41	36.41	37.00	0.59
	Final	21.10	146.10	17.40	166.20	19.75	144.32	100.43	97.00	3.43
2	Inicial	21.10	146.10	17.40	166.20	19.75	144.32	100.43	97.00	3.43
	Final	127.50	150.52	147.30	150.50	125.41	151.95	359.94	355.07	4.87
3	Inicial	127.50	150.52	147.30	150.50	125.41	151.95	359.94	355.07	4.87
	Final	130.00	38.60	130.30	18.80	128.09	39.20	270.87	268.90	1.97
4	Inicial	130.00	38.60	130.30	18.80	128.09	39.20	270.87	268.90	1.97
	Final	30.10	149.00	17.25	164.45	29.51	147.10	129.75	130.00	0.25
5	Inicial	26.60	24.70	41.50	38.00	24.25	23.94	41.75	41.00	0.75
	FInal	21.00	147.90	16.20	167.80	19.02	146.39	103.56	102.31	1.25
6	Inicial	21.00	147.90	16.20	167.80	19.02	146.39	103.56	102.31	1.25
	Final	116.50	153.80	136.20	149.40	113.98	154.03	347.41	348.31	0.90
7	Inicial	116.50	153.80	136.20	149.40	113.98	154.03	347.41	348.31	0.90
	Final	129.05	32.90	130.80	12.90	126.80	33.65	275.00	275.91	0.91
8	Inicial	129.05	32.90	130.80	12.90	126.80	33.65	275.00	275.91	0.91
	Final	25.20	149.20	14.50	166.10	23.90	147.10	122.34	122.23	0.11
								Error promedio		1.77
								Error mínimo		0.1094
								Error máximo		4.8721

Figura B.0.3: Resultados de las pruebas de validación del sistema de visión (trayectorias)

Error posición en X (cm)	Error posición en Y (cm)	
2.00	0.89	
1.35	1.78	
1.35	1.78	
2.09	1.43	
2.09	1.43	
1.91	0.6	
1.91	0.6	
0.59	1.9	
2.35	0.76	
1.98	1.51	
1.98	1.51	
2.52	0.23	
2.52	0.23	
2.25	0.75	
2.25	0.75	
1.30	2.1	
1.90	1.14	Promedio
0.59	0.23	Mínimo
2.52	2.1	Máximo

Figura B.0.4: Resultados de las pruebas de validación del sistema de visión (trayectorias)

Apéndice C

Estructura de bloques de la aplicación

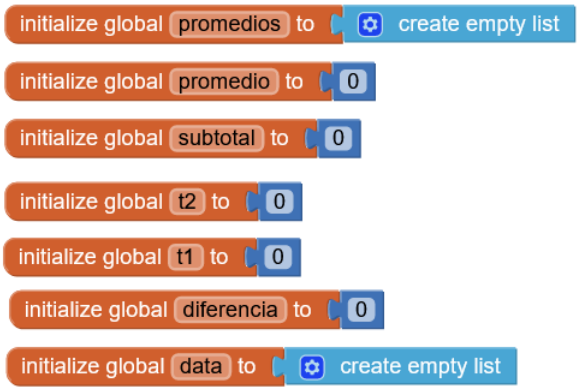


Figura C.0.1: Primera parte de los bloques del programa, inicialización de variables

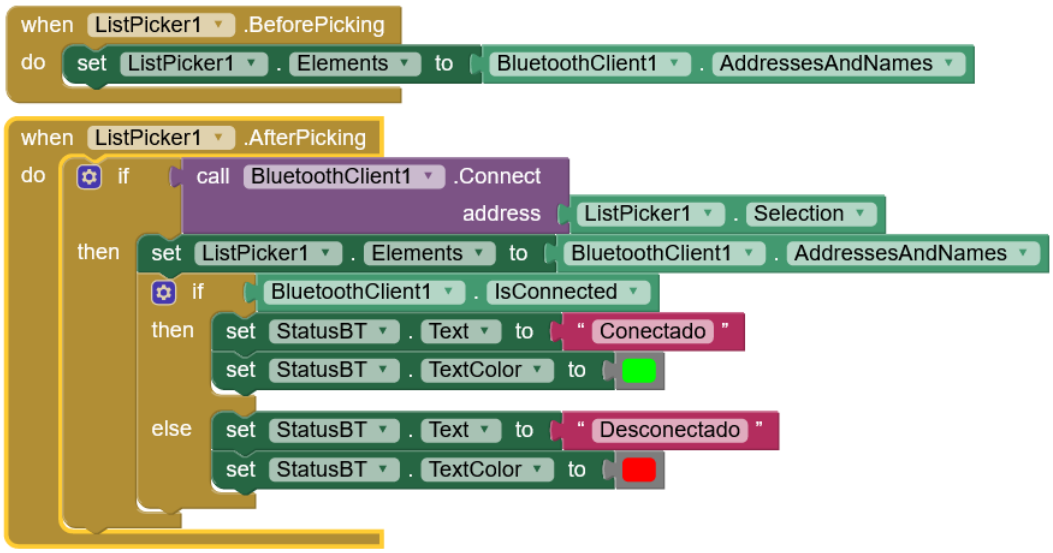


Figura C.0.2: Segunda parte de los bloques del programa, configuración de la aplicación

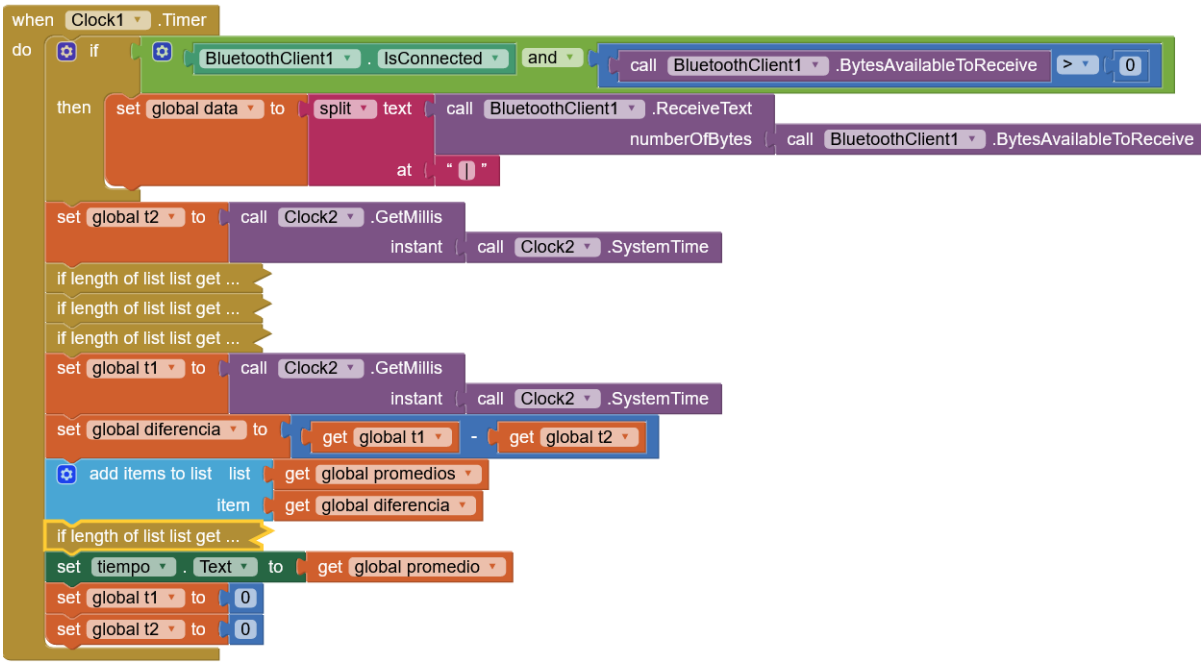


Figura C.0.3: Tercera parte de los bloques del programa, recepción y cálculo de Δ_t