Universidad Iberoamericana

Estudios con Reconocimiento de Validez Oficial por Decreto Presidencial
Del 3 de abril de 1981

"Diseño y construcción de un robot modular multientorno"

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS DE LA INGENIERÍA

Presenta

PABLO PANIAGUA CONTRO

Director: Dr. Eduardo Gamaliel Martínez Hernández
Codirector: Dr. Guillermo Fernández Anaya

Ciudad de México 2017

“Diseño y construcción de un robot modular multientorno”

Universidad Iberoamericana

Pablo Paniagua Contro

Diciembre de 2016

Resumen

El presente proyecto trata sobre el diseño e implementación de estrategias de coordinación de movimiento de robots modulares omnidireccionales terrestres y aéreos que conforman un sistema móvil de capacidades multientorno. Se desarrollan estrategias de control de avance en formación basadas en funciones de distancia y orientación, las cuales permiten el movimiento conjunto de robots heterogéneos. Dichas estrategias se adecúan para lograr la convergencia a un punto, seguimiento grupal de trayectoria y comportamientos de unión y despliegue entre los robots. Finalmente se implementan estas leyes de control en una plataforma robótica por medio de un sistema de captación de movimiento en un ambiente controlado.

Declaración

Yo Pablo Paniagua Contro en calidad de autor de la tesis, “Diseño y construcción de un robot modular multientorno” autorizo para su utilización en la Biblioteca Francisco Xavier Clavigero en formato electrónico o impreso y sin fines de lucro, cuyo titular es la Universidad Iberoamericana Ciudad de México, por un período ilimitado a contar desde la fecha de la publicación de la presente, con expresa renuncia a los derechos derivados de su explotación en dichos medios y durante el mencionado período.

Índice general

Capítulo 1

Introducción

En la actualidad la robótica es una de las áreas con mayor importancia para la sociedad en general. Esto se debe a que desde los inicios del siglo XX, ha tomado papeles fundamentales en áreas como la automatización industrial donde brinda mejores y más seguras condiciones de trabajo. Otras aplicaciones es la ciencia aeroespacial donde se puede resaltar el trabajo que realiza al explorar lugares y entornos inalcanzables para el ser humano. Sus avances también ha alcanzado áreas en las que tiene aplicaciones sumamente comúnes y disponibles para la humanidad, por ejemplo, en la industria del juguete.

La robótica móvil es una rama de la robótica que se encarga del estudio de los robots no fijos a un punto, es decir, que tienen la capacidad de desplazarse. Su campo de trabajo nos ofrece una amplia posibilidad de aplicaciones, como pueden ser la agricultura, inspección, limpieza, logística, entretenimiento, transporte, seguridad y defensa, entre otras. Dependiendo del entorno donde el robot móvil se traslade, éste se podrá clasificar en terrestre, aéreo, acuático, sub-acuático, espacial, etc.

El estudio de los robots móviles ha ido creciendo y la información que se ha generado actualmente sobre el modelado y control de los robots terrestres, acuáticos y aéreos es muy amplia y cononcida en la literatura. Recientemente, ha comenzado a surgir una evolución natural de estos sistemas robóticos móviles (SRM) en donde se plantea la combinación de motricidades y capacidades de desplazamiento para crear sistemas robóticos móviles multientorno (SRMM), los cuales se definen como SRM capaces de desplazarse en dos o más ambientes.

Una de las muchas ventajas que ofrecen estos SRMM puede ser la realización de tareas en las que anteriormente se utilizaban dos o más SRM, ya que ahora se pueden suplir por un solo SRMM. Otra ventaja es la gran versatilidad que nos ofrece en cuanto a movilidad, ya sea para seguridad, inspección, mapeo y vigilancia, como también para soluciones de transporte, tanto de carga como de pasajeros.

Uno de los principales retos que conlleva la creación de los SRMM es su modelado y control, ya que existe una cantidad infinita de posibilidades y configuraciones en cuanto a diseño y capacidades se refiere. Cada combinación de motricidad plantea nuevas metodologías de modelado que no necesariamente encajan con las de los

modelos tradicionales. También se puede apreciar que el control que se requiere para un SRMM involucra la extensión de problemas fundamentales de posicionamiento, seguimiento de trayectorias y evasión de colisiones a distintos entornos.

Desde el punto de vista práctico, la complejidad en el diseño mecánico y electrónico también aumenta de manera considerable. Una de las razones que se generan es que algunas características básicas de cada entorno se contraponen entre sí. Por ejemplo, el bajo peso que requiere una aeronave para volar y el diseño robusto y pesado de los SRM submarinos, hacen que el diseño mecánico y electrónico busque un equilibrio en donde todos estos factores puedan ser compatibles y funcionar correctamente para un fin.

Por otro lado, podemos dividir estos SRMM en dos grandes subgrupos, los SRMM que se formados por completo de un solo cuerpo, y los SRMM modulares, los cuales constan de diferentes SRM que se unen para formar un SRMM. Los SRMM modulares presnetan ventajas sobre los SRMM de un solo cuerpo, tales como que ofrecen mayor versatilidad y adaptabilidad a diferentes tareas que deben realizar. También una alta tolerancia a fallas, ya que cada módulo independiente puede sustituirse para que el conjunto de SRMM modulares continúe operando.

Asimismo, el diseño y construcción de estos SRMM modulares conlleva retos importantes, como lograr un diseño mecánico donde los módulos se acoplen y desacoplen correctamente. Al pensar en el caso de que estos robots sean totalmente autónomos, el problema se vuelve más complejo aún, ya que se requiere implementar leyes de coordinación de movimiento preciso a través de comportamientos colaborativos y seguimiento grupal de trayectorias, que permitan el acople y desacople de los módulos, entre otras rutinas de movimiento.

Para lograr la implementación de estas leyes de control en los SRMM modulares es necesario instalar un sistema de sensado, el cual nos ofrezca los datos necesarios de localización, proximidad, etc. Es por esto que en la actualidad hay dos grandes corrientes de estudio: La primera requiere de la posición del SRMM con respecto a un marco de referencia fijo. La segunda busca realizar una coordinación de movimiento tomando en cuenta solamente las distancias y orientaciones relativas que existen entre los SRMM modulares. Ambos paradigmas requieren un conjunto distinto de sensores y sistemas de comunicación.

En la literatura, los problemas fundamentales de coordinación de movimiento se han resuelto con ayuda de los conceptos de formación, evasión de colisiones y marcha de SRM heterogéneos para ser utilizados de forma colaborativa. En este proyecto se plantea el diseño y la construcción de un SRMM modular, el cual reúna todas las ventajas ya antes mencionadas para SRMM modulares. Para esto será necesario diseñar y adecuar leyes de coordinación de movimiento en donde se definan rutinas de formación con seguimiento de líder, acoplamiento y desacoplamiento de módulos.

Esta tesis de presenta una propuesta para estudiar el caso de un SRMM modular conformado por un solo robot aéreo multi-rotor de tipo omnidireccional y un conjunto de SRM terrestres de tipo omnidireccional también. La estrategia de control planteada se basa en el consenso de robots heterogéneos utilizando únicamente dis-

tancias y orientación. Se propone agregar términos que aseguren la convergencia de los ángulos que existen entre los robots, para de esta forma lograr el acoplamiento adecuado de los módulos. Así mismo la estrategia de coordinación planteada puede ser aplicada en un caso general en el que se tenga un conjunto de SRM terrestres con diferente topología de comunicación y diferente modelo cinemático (tres o más ruedas), siempre que este sea de primer orden. Asimismo, el resultado aplica para cualquier robot aéreo de tipo omnidireccional multi-rotor (Bicóptero, tricóptero, cuadricóptero, etc.) y modelado como un robot de segundo orden.

Se reportan tanto resultados en simulación como también resultados experimentales. La plataforma experimental fue diseñada y elaborada por completo para esta tesis.

1.1. Planteamiento general del problema

Dadas las necesidades de desplazamiento de un robot en ambiente terrestre y aéreo, el presente proyecto plantea la construcción, como prueba de concepto, de un robot multientorno modular con las siguientes características físicas:

- Compuesto de un robot aéreo multi-rotor de movimiento omnidireccional. El cual debe ser capaz de vuelo estacionario y estar dotado de una combinación de servos y multi-rotores que permitan su desplazamiento instantáneo hacia todas direcciones a través de una adecuada ley de linealización por retroalimentación.
- Un grupo de robots móviles terrestres con desplazamiento omnidireccional. Preferentemente aquellos dotados de ruedas suecas o con rodillos.

Deberá ser posible la implementación de algoritmos de coordinación de movimiento a través de:

- La medición de distancias y orientaciones para posicionarse de forma relativa en el plano y el espacio respecto a otros robots.
- Información sobre una trayectoria grupal de desplazamiento.
- La conmutación de rutinas de control basadas en estrategias de coordinación de movimiento.

Estos algoritmos deberán de incluir las siguientes funciones básicas:

- La evasión de colisiones entre robots.
- Que el diseño de las estrategias sea genérico para aplicarse a cualquier robot aéreo multi-rotor de tipo omnidireccional y cualquier SRM terrestre de tipo omnidireccional, con cualquier topología de comunicación y con posibilidad de extenderse en un futuro a SRM acuáticos y subacuáticos de tipo omnidireccional.

De forma que pueda implementar las siguientes rutinas básicas para un robot multi-entorno:

- Control de formación estática y avance en formación de los robots separados, con seguimiento de trayectoria de un líder.
- Realizar unión de robot aéreo con uno o varios robots terrestres con control del ángulo de orientación, puesto que, se está considerando que para realizar esta unión deberá de existir un ángulo de orientación especifico entre ambos.
- Realizar operación de separación para cambiar de modo unido a formación estática de robots modulares separados.

El trabajo experimental se limita a lo siguiente:

- Estudio de caso de un robot aéreo de tipo tricóptero o cuadróptero, considerando su modelo dinámico traslacional y rotacional reducido.
- Módulos de robots terrestres de tres ruedas omnidireccionales, considerando su modelo cinemático.
- El espacio de trabajo será conocido y controlado y contará con un sistema de captación de movimiento, el cual actuará como la retroalimentación para las leyes de control.
- Las estrategias de control serán evaluadas en una computadora central y enviadas a cada uno de los núcleos de control de los robots.
- No se abordará con detalle los sistemas de sujeción de los módulos de los robots.

A continuación, se plantea el objetivo sujeto a las condiciones antes mencionadas.

1.2. Objetivos

1.2.1. General

- Diseño y construcción de un SRMM modular que implemente estrategias de coordinación de movimiento de robots omnidireccionales basado en mediciones de distancias y orientación y evaluado en una plataforma experimental en ambiente controlado.

1.2.2. Específicos

- OE1 Revisión bibliográfica de estrategias convencionales de coordinación de robots móviles terrestres, aéreos y acuáticos.
- OE2 Diseño y construcción de un SRMM modular con capacidad de desplazarse en tierra y aire.
- OE3 Obtención de los modelos cinemáticos y/o dinámicos de los robots tanto por separado como en conjunto.
- OE4 Diseño de leyes de control para cada etapa de funcionamiento del SRMM modular, así como la definición de su comportamiento en cada entorno.
- OE4 Comprobación de las estrategias de coordinación de movimiento en una plataforma experimental de SRMM modular.

1.3. Justificación

Actualmente la mayoría de las investigaciones en robótica móvil se centran en un solo tipo de locomoción y en un solo tipo de medio ambiente. También se han encontrado grandes ventajas en los SRMM modulares, como pueden ser su capacidad de sobreponerse a fallas, así como su escalabilidad y diversidad. Es por esto que se desea innovar en un nuevo campo de estudio con el desarrollo e investigación de un robot modular multientorno compuesto por módulos robóticos autónomos. De esta forma se espera obtener tanto las ventajas que ofrecen los SRM modulares por separado y en conjunto.

Actualmente no se ha encontrado ningún desarrollo que proponga una implementación como la que se propone. Esto es una gran oportunidad no solo de innovación, sino también de abrir paso a un nuevo campo de estudio para la robótica, en donde se comiencen a contemplar sistemas heterogéneos modulares multientorno.

Existe también la posibilidad de generar productos comerciales tales como pueden ser sistemas embebidos electrónicos, robots didácticos, robots de uso industrial, algoritmos de coordinación de movimiento y adiciones a algunas leyes de control existentes, las cuales pueden ser utilizadas en robots industriales o laboratorios de robótica en instituciones académicas y de investigación.

1.4. Contribución

La contribución principal de la tesis es el diseño, desarrollo e implementación de estrategias de coordinación de movimiento necesarias para el funcionamiento autónomo de un SRMM modular. Así también el el diseño mecatrónico de la plataforma experimental. La contribución teórica se sustenta en el desarrollo de estrategias de control utilizando solo distancias y orientación para controlar la formación de robots

heterogéneos basadas en la distancia que existe entre ellos y la convergencia de sus ángulos de orientación que permite la conexión de módulos para formar el robot multientorno. El uso de este tipo de robots permite realizar las mismas tareas que los robots que existen actualmente por separado y al unirse nos ofrece una motricidad combinada, en donde obtenemos ventajas tales como trabajo cooperativo o sistemas adaptables a las condiciones ambientales.

1.5. Estructura

En el capítulo dos de este trabajo se presenta un marco teórico de los robots multientorno y sus aplicaciones, la coordinación de movimiento de robots móviles, el modelado cinemático y dinámico de robots omnidireccionales y el estado del arte de los robots móviles multientorno y los robots modulares, así como el control de formación basado en distancias y orientación; en el tercer capítulo se exponen las estrategias de consenso de robots modulares, principal aportación teórica del trabajo. El cuarto capítulo muestra el trabajo experimental; y en el quinto capítulo se plantean conclusiones y trabajo futuro.

Capítulo 2

Fundamentos

A continuación se presentan las ideas básicas, tanto conceptuales como matemáticas, que servirán como sustento para esta tesis. Se abordará brevemente la vision general sobre la robotica movil, robotica movil multi-entorno, el concepto de SRMM modulares, las estrategias de coordinacion de SRM y el modelado de robots omnidireccionales terrestres y aéreos.

2.1. Robótica móvil multi-entorno

Los robots móviles se pueden definir como sistemas que tienen la capacidad de trasladarse por medio de actuadores a bordo, los cuales le brindan las características necesarias para desplazarse en uno o más medio ambientes (agua, aire, tierra, etc.). Algunos configuraciones clásicas de robots móviles son: de tipo uniciclo, cuadrúpedos, tipo carro convencional, submarinos, antropomórficos, entre otros [49, 23, 6].

El estudio de los SRM ha aumentado considerablemente desde fines de la década de los 70's, ya que han encontrado un gran campo de aplicación. Ofrecen ventajas en cuanto a seguridad, calidad de las condiciones de trabajo, alcance, entre otras. Por otro lado, aunque sea un campo con crecimiento en su análisis, existen aún oportunidades de investigación en las áreas de localización, seguimiento de trayectorias, optimización, etc. [10, 64, 47]. Esto crea un área de oportunidad para el estudio de nuevas propuestas de SRM y de esta forma generar ideas innovadoras las cuales ofrezan mayores y mejores posibilidades de usos de SRM para beneficio de la sociedad.

Los SRM con capacidades multientorno, son capaces de desplazarse en más de un terreno (tierra, agua, aire, etc.). Un caso particular son los robots anfibios, los cuales pueden desplazarse en agua y tierra. Algunos ejemplos reportados en la literatura son: Robots serpiente, tipo esfera, salamandra, etc. [62, 25, 13, 36]. Actualmente está surgiendo una fuerte corriente de estudio en donde se impulsa a los SRMM, ya que ofrecen mayores versatilidad de movilidad y por lo tanto mayor número de aplicaciones.

Algunos de los usos de los SRMM han sido: monitoreo ambiental en zonas peligrosas, vigilancia de campos de cultivo, operaciones de búsqueda y rescate de personas desaparecidas, operaciones militares en litorales ya sea para escoltar a las tropas desde la costa hasta el océano, o para transporte de provisiones, recoleccción de muestras en ambientes tóxicos o en donde la vida humana es insostenible, etc. [11, 33, 27, 17]

También se han comenzado a realizar investigaciones en donde este tipo de SRMM emplean estrategias de coordinación de movimiento para la realización de tareas colaborativas. Tal es el caso de la formación y seguimiento con respecto a un líder [54]. Otro ejemplo puede ser el muestreo de imágenes en forma de mosaico en ambientes tanto terrestres como acuáticos o incluso en ambientes híbridos. [26]. A diferencia de robots tipo anfibios, los cuales son inspirados por la naturaleza, existen topologías de SRMM artificiales que se han desarrollado para investigación. Tal es el caso de [63, 42, 55].

2.2. SRMM modulares

Son todos aquellos SRMM compuestos de varios modulos independientes. Cada módulo que conforma al SRMM es un robot completo por sí solo. Un ejemplo es el robot anfibio de tipo serpiente reportado en [62], el cual consta de articulaciones modulares con la capacidad de actuar por sí solas o como un conjunto. Las aplicaciones que esta serpiente están enfocadas al rescate marítimo y deteccion de ambientes anfibios, siempre cumpliendo con los requerimientos del terreno y capacidad modular.

A pesar de todas las ventajas que ofrecen los SRMM se ha encontrado que tambien presentan fuertes desventajas, entre las mas importantes se encuentran:

- Una consecuencia esperada de las plataformas que combinan varios tipos de motricidad es que para lograr el desplazamiento multientorno se comprometen algunos aspectos en el diseño. Esto se debe a que algunos requerimientos que son básicos para el funcionamiento en un entorno se contraponen con los de otro medio ambiente. Se requiere sacrificar eficiencia en ambos aspectos y llegar a un consenso en el cual se logren combinar estas características. Es por esto que presentan un bajo desempeño en cuanto a desplazamiento y eficiencia en todos los entornos en los que operan.

- Una manera de evitar el sacrificio de las capacidades del SRMM, ha sido crear y utilizar materiales con mejores prestaciones, los cuales al estar específicamente desarrollados para esta función, generan un costo y tiempo de desarrollo muy elevado, lo que los hace poco accesibles para su uso comercial.

- Existe un complejidad considerable en cuanto a diseño. Una evolución natural de estos SRMM modulares sería el lograr que el acople y desacomple de los módulos se realice de forma autónoma. Esto conlleva a una mayor complejidad teórica, mecánica, electrónica, de procesamiento y de instrumentación.

Por esta misma razón, se propone utilizar algunos de los SRM clásicos de la literatura de robots móviles, los cuales no se vean limitados en sus capacidades y cuenten con las ventajas para la conformación de un SRMM modular. La idea básica es lograr la construccion de un SRMM a través de la unión de los SRM simples y con ello aprovechar la teoría desarrollada para cada uno de los módulos de forma independiente y en conjunto a través de estrategias de coordinacion de movimiento.

2.3. Coordinacion de SRM

Se considera que hay tres áreas de particular interés al tratar el trabajo colectivo de grupos de robots[47, 8]:

- La coordinación de movimiento aborda la problemática que existe cuando un robot debe mantener una posición relativa con respecto a otros dentro de un mismo grupo [10]. Comprende todas aquellas tácticas en donde los SRM tienen la habilidad de llegar a un consenso grupal para realizar una tarea asignada. Algunas de sus aplicaciones son: mapeo, monitoreo, trabajos en ambientes peligrosos, formación de satélites, sistemas de navegación en carreteras, vigilancia, robots jugadores de soccer, coordinación de vehículos acuáticos y sub-acuáticos, etc. [2, 16, 61, 9, 64]

- La asignación de tareas describe la forma en la que los SRM distribuyen sus funciones para realizar de la mejor manera posible alguna tarea. Es decir, que no solo se enfoque en la realización correcta de la tarea, sino que también busque distribuir el trabajo entre los agentes de la mejor forma posible. [4, 24]

- La La comunicación entre robots es vital para su trabajo colectivo y puede ser de dos formas: implícita y explicita [44, 52]. Esta se ve afectada en gran parte por el sistema de sensado que utiliza en cada sistema robótico multi-agente.[44, 41, 58]

Un enfoque de coordinación multi-robot es conocido como “comportamiento emergente o artificial”, el cual se basa en la teoría de sistemas dinámicos, control y grafos de comunicación. La cinemática o dinámica de cada robot móvil y las interacciones que existen entre ellos son modeladas matemáticamente. Se estudia la coordinación de un grupo de robots móviles a través de la asignación de líderes [18, 34**? ?**]. Se han mostrado aplicaciones para las realización de mapeo y localización [57], transporte y manipulación de objetos [3, 60, 59], etc.

Existen tres problemas fundamentales dentro de la coordinacion de movimiento de SRM de particular interés para este proyecto, ya que son la base para las rutinas de movimiento del SRMM modular:

2.3.1. Control de formacion

El control de formacion es una extensión a SRM de los conceptos del llamado "problema de consenso", el cual plantea la convergencia de un conjunto de agentes móviles a un punto en común.[51]. El control de formación comprende todas aquellas estrategias en las que se desea conservar una posición relativa para cada SRM con respecto a un subconjunto de robots [45]. Estas leyes de control se diseñan para cumplir como mínimo con los siguientes requerimientos:

- Converger a patrones de formacion geométricos obedeciendo una topología de comunicación de los robots, la cual puede ser descrita utilizando la teoria de grafos[15].
- Debe evitar las colisiones entre los miembros del conjunto de robots.[10]

La teoría de formacion basada en consenso, la primera surgida históricamente, propone resolver estos problemas utilizando campos potenciales tanto atractivos como repulsivos, para asegurar la convergencia a la formacion y evasion de colisiones simultaneamente [51, 31]. Algunos ejemplos de aplicación de este enfoque han sido para seguridad, vigilancia, mapeo, etc.[5]. Dado que el patrón de formación puede ser definido de distintas maneras, existen dos grandes enfoques en los que se ha abordado el control de formación:

- La formacion basada en posiciones (FBP), cuyo patrón se establece con posiciones relativas deseadas [30].
- La formacion basada en distancias (FBD), utilizando solamente la informacion de la distancia y ángulos de orientación que existe entre cada par de robots para lograr converger a una formacion deseada.[35]

La FBP muestra algunas desventajas con respecto al enfoque de FBD, principalmente en la implementacion física, ya que se requiere tener un marco de referencia global para obtener la posicion relativa de cada uno de los SRM. Esto presenta diversas implicaciones, ya que se requieren sensores globales preinstalados en el área de trabajo, o estrategias de estimación de posiciones como la presentada en [1].

Por otro lado la FBD presenta en ocasiones un enfoque inspirado en la biología donde los SRM solo se forman con respecto a sus agentes adyacentes [22, 65]. Esta perspectiva tambien plantea grafos de comunicacion variantes, ya que su implementación no requiere forzosamente el conocimiento de datos grupales. Es importante mencionar en una FBD, el patrón de formación deseado se plantea en función a distancias entre pares de robots, lo que genera la posibilidad de diferentes configuraciones en las que se satisfacen las distancias deseadas. Por lo tanto, si se desea converger a una única formación, se requieren definir patrones rígidos de comunicación. Esto significa que por lo menos $(n-3)$ aristas deberan ser definidas en el grafo de comunicación [46, 38]. También, se estima la distancia usando cámaras en [21] incluyendo los efectos de cuantificación y retardos en la medición.

2.3.2. Control de avance en formación o marcha

Se define como marcha al seguimiento de una trayectoria por un conjunto de SRM manteniendo una frormacion.[20, 32, 50]. El reto es garantizar la convergencia a una formacion deseada y a una trayectoria al mismo tiempo, garantizando estabilidad y rigidez en la formación [39, 28]. La estrategia de lider seguidor plantea un escenario en el que solo un agente dentro del conjunto de SRM tiene conocimiento de la trayectoria a seguir y los demás robots solamente siguen al líder manteniendo la formacion, tal como lo ilustra la figura 2.1 [50, 19]. Otros ejemplos de marcha se pueden encontrar en [7, 40, 56, 12].

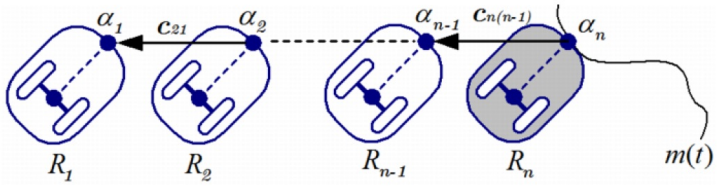


Figura 2.1: Ejemplo de marcha con esquema líder-seguidores.

2.3.3. Evasion de colisiones

Cuando los robots son movidos a ocupar su lugar en un patrón de formación deseado, es posible que los agentes puedan colisionar, es por esto que al implementar estrategias de formacion, es común encontrarse con términos de evasión de colisiones.

Uno de los metodos más utilizados es el de "funcione potenciales artificiales" (FPA). Este metodo consiste en utilizar funciones potenciales tanto atractivas como repulsivas como entradas de control tal que al aplicar un gradiente negativo se pueda llegar a la convergencia de la formacion evitando colisiones. Estas funciones potenciales artificiales se diseñan dependiendo de las distancias deseadas entre cada par de agentes, en donde el potencial repulsivo tiende a infinito al acercarse los robots y el equilibrio solo se logra cuando el error de la distancia deseada es cero.

2.4. SRMM modulares y sus aplicaciones

Debido a que el campo de SRMM modulares es relativamente nuevo no se ha desarrollado una gran cantidad de prototipos. Aún así se puede apreciar que existe una tendencia de diseño hacia los SRMM modulares de tipo serpiente. Esto se debe a que los robots de este tipo tienen muchos grados de libertad, haciéndolos ideales para desplazasrse en lugares confinados y ambientes complicados, como pueden ser las cuevs, barcos hundidos, edificios colapsados, etc.[48]

Otro ejemplo es el robot modular tipo serpiente anfibio, diseñado para rescate, detección en ambientes terresres y acuáticos. El robot se compone de 10 modulos con dos grados de libertad cada uno. La unión de los robots se realiza siempre de la misma

manera, logrando asi un desplazamiento ágil en 3D. Para simular su modelo dinámico se utilizó el software Webots. Se presentaron resultados en donde el robot se desplaza tanto en agua como en tierra, probando asi que el diseño mecánico funciona.[62]. Un SRMM modular inspirado por la locomoción de un caimán se propone en [37], el cual se compone de catorce pequeños SRM. Se muestra el modelo hidrodinámico de el robot ensmblado así como también el de cada robot por separado.

Capítulo 3

Estrategias de robot multientorno a través de consenso de robots modulares

Este capítulo contiene la aportación teórica más importante de la presente tesis. Se establece el planteamiento del problema de coordinación y se diseña una ley de control FBD que puede adecuarse para lograr las rutinas generales de cualquier SRMM modular. Como se mencionó anteriormente, la estrategia es general para el caso de n robots móviles de tipo omnidireccional con cualquier topología de comunicación bidireccional de tipo rígido. También se presentan simulaciones numéricas enfocadas al comportamiento de SRMM modulares.

3.1. Definición del grupo de robots móviles

Sea $N = [R_1, ..., R_n]$ un conjunto de SRM con una posicion en el plano dada por $\xi_i = \{\xi_1, ..., \xi_n\}$, donde $\xi_i \in \Re^2$, $i = 1, ..., n$. El grupo de SRM se compone por $n - 1$ robots terrestres cuya traslación es modelada como un slimple integrador y un robot aéreo R_n como un doble integrador. Comúnmente el modelo de doble integrador alude a modelado dinámico, mientras que el simple integrador se encuentra relacionado con los modelos de postura cinemática, la cual es útil para diseñar leyes de control simples para trabajo experimental. El robot R_n es considerado como el líder de grupo, con mayores capacidades de sensado que el resto de los robots. El robot R_n es un robot aéreo omnidireccional que se mueve en un espacio tridimensional, cuya proyección en el plano será el objeto de coordinación respecto a los robots terrestres.

Los modelos de postura de primer y sengundo orden no son restrictivos ya que modelos de robots móviles físicos pueden simplificarse a ellos a través de una linealización por retroalimentación. Para mostrar lo anterior, las siguientes secciones explican cómo robots omnidirecionales terrestres y aéreos pueden simplificarse a sistemas de primer y segundo orden, respectivamente.

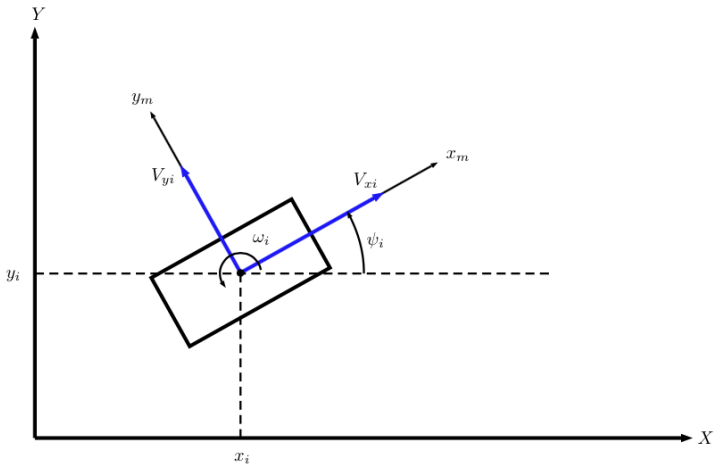


Figura 3.1: Modelo cinemático de un robot omnidireccional

3.1.1. Modelos cinemáticos de robots omnidireccionales terrestres

Considere el modelo cinemático de postura de un robot omnidireccional que se desplaza en el plano, tal como lo muestra la Figura 3.1.

Las ecuaciones cinemáticas están dadas por

$$\begin{aligned} \dot{\xi}_i &= \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\psi_i & -\sin\psi_i \\ \sin\psi_i & \cos\psi_i \end{bmatrix}}_{R(\psi_i)} \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} \\ \dot{\psi}_i &= \omega_i \end{aligned} \tag{3.1}$$

donde ξ_i es la coordenada del plano del centro del cuerpo, ψ_i es el ángulo de orientación respecto al eje horizontal, v_{x_i} y v_{y_i} son las velocidades lineales sobre los ejes del marco de referencia del robot x_m y y_m, y ω_i es la velocidad angular.

Note que definiendo la ley de control

$$\begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} = R^{-1}(\psi_i)u_i, \tag{3.2}$$

el sistema de ecuaciones (3.1) se puede reducir a

$$\begin{aligned} \dot{\xi}_i &= u_i \\ \dot{\psi}_i &= \omega_i \end{aligned} \tag{3.3}$$

La ley de control (3.2) impone una linealización por retroalimentacion para las coordenadas traslacionales del robot, y gráficamente se muestra en la figura 3.2. Por lo tanto el SRM se puede representar como un sistema de primer orden. Note que la

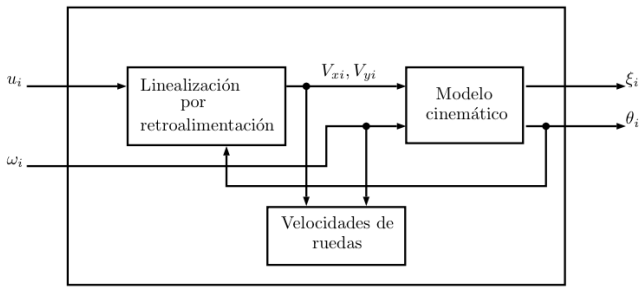


Figura 3.2: Linealización por retroalimentacion

dinamica de ψ_i queda desacoplada y más adelante se utilizará para llegar a consenso respecto a las orientaciones de otros robots.

La distribución de las velocidades lineales y la velocidad angular pueden ser transformadas a las velocidades de cada rueda, de acuerdo a la ubicación de la rueda respecto a la coordenada (x_i, y_i) del robot y al tipo de rueda [14].

Para adecuar un movimiento omnidireccional, los robots terrestres deben estar dotados de ruedas de tipo omnidireccional. Las configuraciones clásicas de la literatura son las de cuatro ruedas tipo mecanum, cuatro ruedas en cruz con rodillos con 90 grados de desviación respecto al eje de rotación de la rueda o tres ruedas omnidireccionales en triángulo equilátero.

Robot omnidireccional con cuatro ruedas mecanum

El caso de robot omnidireccional de cuatro ruedas mecanum está dibujado en la figura 3.3. Para este caso, las velocidades de las cuatro ruedas se obtienen por

$$\begin{bmatrix} \omega_{i1} \\ \omega_{i2} \\ \omega_{i3} \\ \omega_{i4} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & 1 & -(L+l) \\ -1 & 1 & (L+l) \\ -1 & 1 & -(L+l) \\ 1 & 1 & (L+l) \end{bmatrix} \begin{bmatrix} v_{xi} \\ v_{yi} \\ \omega_i \end{bmatrix}$$

donde L y l son las distancias horizontal y vertical, respectivamente de las ruedas respecto al centro del robot y r es el radio de las ruedas. Las ruedas mecanum están dotadas de rodillos a 45 grados respecto al eje de rotación de la rueda tal como lo ilustra la figura 3.4.

Robot omnidireccional con cuatro ruedas en cruz

El caso de los SRM omnidireccionales de cuatro ruedas con rodillos dispuestos a 90 grados del eje de rotacion de la rueda, es ilustrado en la figura 3.5. Para este

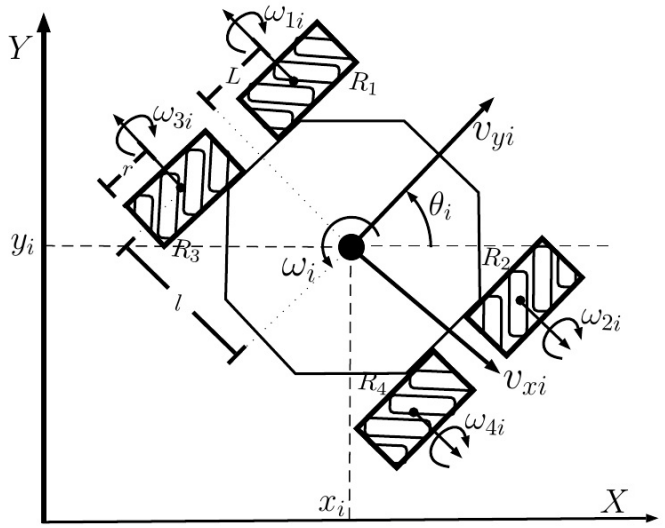


Figura 3.3: Robot omnidireccional con ruedas mecanum

Figura 3.4: Ruedas mecanum

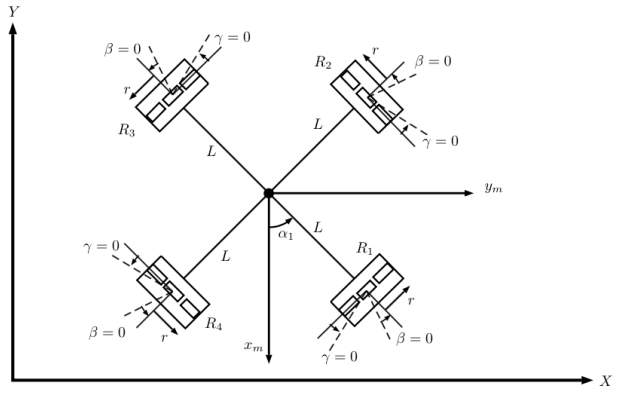


Figura 3.5: Robot omnidirectional terrestre de cuarto ruedas

caso, las velocidades de las cuatro ruedas se obtienen por

$$\begin{bmatrix} \omega_{i1} \\ \omega_{i2} \\ \omega_{i3} \\ \omega_{i4} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \sqrt{2} & -\sqrt{2} & -L \\ \sqrt{2} & \sqrt{2} & -L \\ -\sqrt{2} & \sqrt{2} & -L \\ -\sqrt{2} & -\sqrt{2} & -L \end{bmatrix} \begin{bmatrix} v_{xi} \\ v_{yi} \\ \omega_i \end{bmatrix}$$

donde L es la distancia respecto al centro del robot y r es el radio de las ruedas.

Robot omnidireccional con tres ruedas

El caso de robot omnidireccional de tres ruedas ocupa el mismo tipo de ruedas que el caso anterior y es ilustrado en la figura 3.6. Para este caso, las velocidades de las tres ruedas se obtienen por

$$\begin{bmatrix} \omega_{i1} \\ \omega_{i2} \\ \omega_{i3} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\frac{\sqrt{3}}{2} & \frac{1}{2} & L \\ 0 & -1 & L \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & L \end{bmatrix} \begin{bmatrix} v_{xi} \\ v_{yi} \\ \omega_i \end{bmatrix}$$

donde L es la distancia respecto al centro del robot y r es el radio de las ruedas.

3.1.2. Modelos cinemáticos de robots omnidireccionales aéreos

Cualquier cuerpo aéreo moviéndose en el espacio de forma omnidireccional, puede ser representado por la figura 3.7. Al tratarse de un movimiento en el espacio, considere la coordenada traslacional $[\xi_i, z_i]^T$, donde $\xi_i = [x_i, y_i]$ es la componente del plano XY, similar al caso de robots terrestres, y la coordenada z_i corresponde a la altura del drone. También considere los tres ángulos de orientación ϕ_i, θ_i, ψ_i que

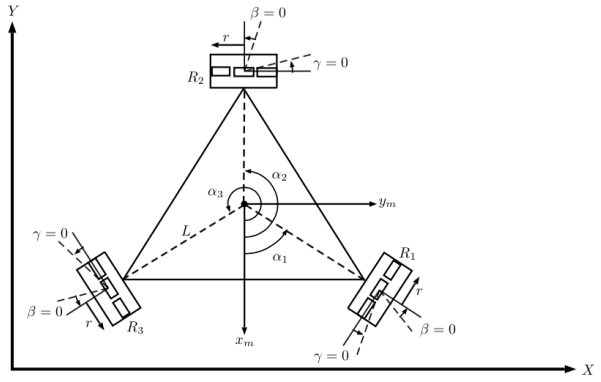


Figura 3.6: Robot omnidirectional terrestre de tres ruedas

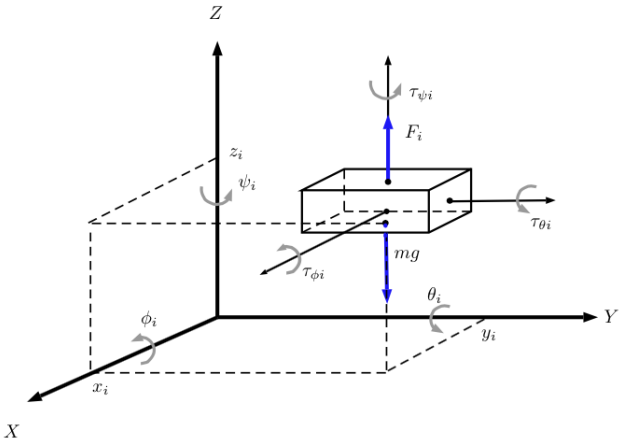


Figura 3.7: Robot Omnidirectional aéreo

corresponden a los angulos de pitch, roll y yaw, respectivamente. Sea m_i la masa del SRM y g es la constante de gravedad como parámetros constantes del drone. Entonces, de acuerdo a [29], el modelo dinámico reducido de este robot aéreo está dado por

$$m\ddot{\xi}_i = \begin{bmatrix} F_i \sin\theta_i \\ -F_i \cos\theta_i \sin\phi_i \end{bmatrix} \tag{3.4}$$

$$m\ddot{z}_i = F_i \cos\theta \cos_i \phi_i - mg \tag{3.5}$$

$$\ddot{\phi}_i = \tau_{\phi_i} \tag{3.6}$$

$$\ddot{\theta}_i = \tau_{\theta_i} \tag{3.7}$$

$$\ddot{\psi}_i = \tau_{\psi_i} \tag{3.8}$$

donde las entradas de control son F_i, como la fuerza de empuje aplicada en el centro de masa, y τ_{ϕ_i}, τ_{θ_i} y τ_{ψ_i} como los torques utilizados para controlar los ángulos de orientacion. Observe que el ángulo respecto al eje z, dado por ψ_i no influye en la dinámica traslacional.

De acuerdo con [29], el control de una aeronave omnidireccional se puede descomponer en dos niveles jerárquicos (postura y orientacion) siempre y cuando asumamos que el control rotacional converge a su valor deseado mas rápido que la dinámica traslacional.

Considerando entonces que las entradas de control para la dinámica traslacional están dadas por F_i, ϕ_i y θ_i, la linealización por retroalimentación para el lazo de traslación está dada por:

$$\begin{aligned} \phi_i^* &= \arctan\left(-\frac{u_{iy}}{u_{iz}+g}\right) \\ \theta_i^* &= \arctan\left(\frac{u_{ix}\cos\phi_i}{u_{iz}+g}\right) \\ F_i &= \frac{m(u_{iz}+g)}{\cos\theta_i \cos\phi_i} \end{aligned} \tag{3.9}$$

la cual linealiza la dinamica traslacional a

$$\ddot{\xi}_i = u_i = [u_{ix}, u_{iy}]^T \tag{3.10}$$

$$\ddot{z}_i = u_{iz} \tag{3.11}$$

Note que la dinámica sobre el plano XY queda simplificada al caso de doble integrador. Como se mencionó anteriormente, las leyes de control (3.9) generan los setpoints ϕ_i^* y θ_i^* de los ángulos ϕ_i y θ_i, respectivamente. Entonces, en el lazo interno de orientación, las siguientes leyes de control aseguran la convergencia a estos valores deseados.

$$\tau_{\phi_i} = \ddot{\phi}_i^* - k_1\left(\dot{\phi}_i - \dot{\phi}_i^*\right) - k_2\left(\phi_i - \phi_i^*\right) \tag{3.12}$$

$$\tau_{\theta_i} = \ddot{\theta}_i^* - k_1\left(\dot{\theta}_i - \dot{\theta}_i^*\right) - k_2\left(\theta_i - \theta_i^*\right) \tag{3.13}$$

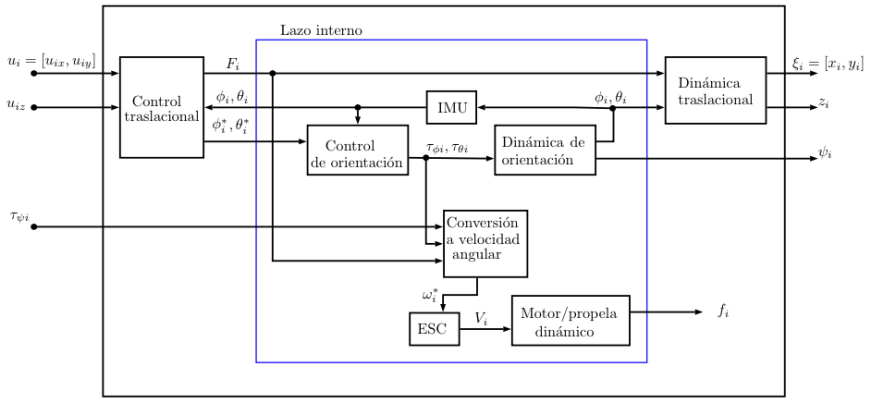


Figura 3.8: Control de un SRM aereo

con k_1 y k_2 positivas y seleccionadas adecuadamente para un cierto coeficiente de amortiguamiento y un rápido tiempo de establecimiento respecto al lazo de control externo. Esta descomposición de control de dos niveles del drone es mostrado en la figura 3.8, donde el control externo linealiza y desacopla las variables de traslación y genera los valores deseados o setpoints de los ángulos internos. Estos a su vez son regulados en un lazo interno de orientación utilizando la retroalimentación de un sensor de masa inercial (IMU por sus siglas en inglés).

Note que, a manera de bloque funcional, después de los rápidos transitorios del lazo interno y sustituyendo ambas leyes de control, el robot aéreo finalmente es visualizado de forma externa con los demás robots para fines de coordinación, como el modelo simplificado:

$$\begin{aligned} \ddot{\xi}_i &= u_i \\ \ddot{z}_i &= u_{iz} \\ \ddot{\psi}_i &= \tau_{\psi_i} \end{aligned} \tag{3.14}$$

donde la dinámica de altura y del ángulo de yaw ψ_i quedan libres para ser coordinados con otras funciones exteriores, como lo ilustra la figura 3.8.

Al igual que el caso de robots terrestres, las entradas de control F_i, τ_{ϕ_i}, τ_{θ_i} y τ_{ψ_i} pueden ser traducidas a las velocidades angulares de los motores con hélices o a ángulos de servomotores del drone. Casos típicos de la literatura son el cuadróptero o el tricóptero, mismos que se mencionan brevemente a continuación.

Cuadróptero

Para el primer caso, acorde a la figura 3.9, las velocidades de los cuatro rotores son obtenidas a través de:

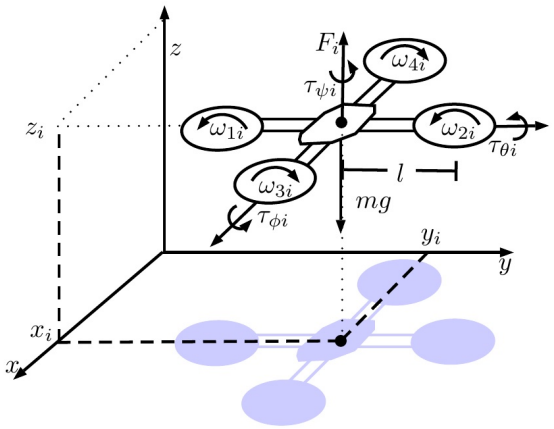


Figura 3.9: Cuadricóptero

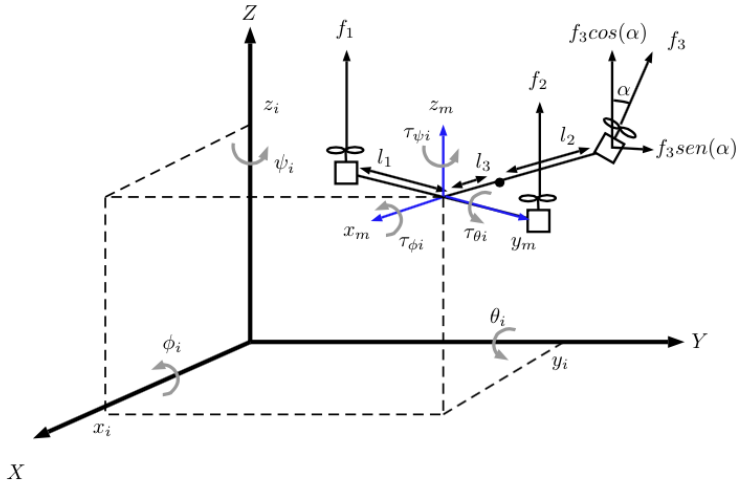


Figura 3.10: Tricóptero

$$\begin{bmatrix} \omega_{i1}^2 \\ \omega_{i2}^2 \\ \omega_{i3}^2 \\ \omega_{i4}^2 \end{bmatrix} = \frac{1}{k} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell & \ell & 0 & 0 \\ 1 & 0 & -\ell & \ell \\ -\frac{b}{k} & -\frac{b}{k} & \frac{b}{k} & \frac{b}{k} \end{bmatrix}^{-1} \begin{bmatrix} F_i \\ \tau_{\phi_i} \\ \tau_{\theta_i} \\ \tau_{\psi_i} \end{bmatrix} \tag{3.15}$$

donde ℓ es la longitud medida del centro a cada uno de los rotores, k es la fuerza de empuje individual dependiente de las propiedades de cada motor y helice y b es la constante de arrastre (drag) de todo el cuerpo.

Tricóptero

Para el caso de un tricóptero con servomotor en la cola [53], las velocidades y ángulo del servo de acuerdo a la figura 3.10, pueden ser obtenidas a través de

$$\begin{bmatrix} kw_{i1}^2 \\ kw_{i2}^2 \\ \cos(\alpha_i)kw_{i3}^2 \\ \text{sen}(\alpha_i)kw_{i3}^2 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{\ell} & 0 & 0 \\ 1 & -\frac{1}{\ell} & 0 & 0 \\ 1 & 0 & \frac{1}{\ell} & 0 \\ 0 & 0 & 0 & \frac{1}{\ell} \end{bmatrix}^{-1} \begin{bmatrix} F_i \\ \tau_{\phi_i} \\ \tau_{\theta_i} \\ \tau_{\psi_i} \end{bmatrix} \tag{3.16}$$

Donde k es nuevamente la constante de empuje de cada rotor, α es el angulo del servomotor de cola y ℓ es la distancia que existe entre el centro del SRM aereo y el rotor.

3.2. Estrategias de coordinación para robot multientorno

Retomando la configuración del grupo de los primero $n-1$ robots como terrestres dados por el modelo simplificado (3.3) y el robot líder aéreo con modelo (3.14), entonces las ecuaciones que describen al grupo de robots está dado por

$$\begin{aligned} \dot{\xi}_i &= u_i, \quad i = 1, ..., n-1 \\ \dot{\psi}_i &= \omega_i, \quad i = 1, ..., n-1 \end{aligned} \tag{3.17}$$

$$\begin{aligned} \dot{\xi}_n &= p_n, \quad \dot{p}_n = u_n \\ \dot{z}_n &= p_z, \quad \dot{p}_z = u_{nz} \\ \dot{\psi}_n &= p_\psi, \quad \dot{p}_\psi = \tau_{\psi_n} \end{aligned} \tag{3.18}$$

donde p_n es la velocidad del robot aéreo R_n, p_z es la velocidad de la coordenada z_n y p_ψ es la velocidad de ψ_n. Note que las señales de control u_i, $i = 1, ..., n$ están dedicadas al posicionamiento traslacional estratégico con fines de consenso. Las señales ω_i, $i = 1, .., (n-1)$ y τ_{ψ_n} controlan las orientaciones de los robots, mientras que u_{nz} está dedicada al control de altura del robot aéreo líder.

3.2.1. Gráfica de comunicación

Asuma que cada robot R_i, $i = 1, ..., n$ está comunicado con g_i robots que pertenecen a su conjunto adyacente $N_i \subset N$ (note que $g_i = \text{card}\,(N_i)$). Entonces, es posible construir una gráfica rígida de comunicación basada en distancias, como la definida en [30], dada por

$$G = \{Q, E, D\} \tag{3.19}$$

donde

- $Q = \{R_1, R_2, ..., R_n\}$ es el conjunto de vértices relacionados con los robots.

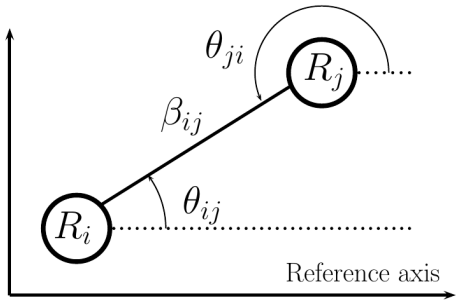


Figura 3.11: Distancia y orientación relativa entre par de robots

- $E = \{(j, i) \in Q \times Q\}$ es el conjunto de aristas que representan las posibles comunicaciones entre pares de robots. Por lo tanto $(j, i) \in E$ si $j \in N_i$.
- $D = \{d_{ji}\}, \forall (j, i) \in E$ es el conjunto de distancias deseadas entre los agentes o nodos i y j, i.e. $\|\xi_i - \xi_j\| = d_{ij} \in \mathbb{R}$, $\forall i \neq j$, $j \in N_i$ en un patrón de formación deseado.

Las gráficas de comunicación que asumiremos para el grupo de robots, tiene las siguientes restricciones:

- No tiene nodos aislados, es decir, $N_i \neq \varnothing$, $i = 1, ..., n$.
- Por simplicidad en el análisis, este trabajo se enfoca al caso de gráfica no dirigidas o de comunicación bidireccional, es decir, si $R_j \in N_i$, entonces $R_i \in N_j$, $\forall i \neq j$.
- Lo anterior conlleva a asegurar que no hay conflictos entre distancias deseadas entre cualquier par de robots, i.e. $d_{ij} = d_{ji}$, $\forall (j, i) \in E$.
- La gráfica de comunicación es rígida. Esta condición asegura que solo un patrón de formación puede alcanzarse con las aristas definidas en E. De acuerdo con [46, 38], una gráfica es rígida su tiene definido al menos $2n - 3$ aristas para n robots. Esto construye un único patrón de formación deseada y evita configuraciones finales no deseadas.

3.2.2. Definición de distancia y orientación relativa

Por otro lado, se define la distancia entre cualquier par de robots en el plano XY como

$$\beta_{ij} = \beta_{ji} = \|\xi_j - \xi_i\| \tag{3.20}$$

y considere θ_{ij} como el ángulo del vector $\xi_j - \xi_i$ con respecto a un eje de referencia (por ejemplo, el polo magnético de la tierra), de acuerdo a la figura 3.11. Por simetría $\theta_{ji} = \pi + \theta_{ij}$. Note que los valores de distancia y orientación pueden ser medidos por una combinación de sensores locales a bordo, como lídars combinados con IMU's.

La derivada temporal de β_{ij} puede ser escrita como

$$\dot{\beta}_{ij} = \frac{(\xi_j - \xi_i)^T \left(\dot{\xi}_j - \dot{\xi}_i\right)}{\beta_{ij}} \tag{3.21}$$

Como se muestra en la figura 3.11, las cantidades $(\xi_j - \xi_i)^T$ pueden ser expresadas en coordenadas polares en términos de la distancia y los ángulos de orientación como

$$[\xi_j - \xi_i]^T = \beta_{ij} \left[\cos\theta_{ij}, \sin\theta_{ij}\right] \tag{3.22}$$

y la ecuación (3.21) puede ser reescrita como

$$\dot{\beta}_{ij} = \left[\cos\theta_{ij}, \sin\theta_{ij}\right] \left(\dot{\xi}_j - \dot{\xi}_i\right) \tag{3.23}$$

3.2.3. Definición de funciones potenciales basadas en distancias

Considere el error de distacnia entre cualquier par de robots R_i y R_j como

$$\varphi_{ij} = \varphi_{ji} = \beta_{ij} - d_{ij} \tag{3.24}$$

Inspirado en [43], es posible definir una familia de funciones potenciales de distancia con comportamiento atractivo y repulsivo (FPD-AR) $V_{ij} : \mathbb{R} \to \mathbb{R}$ para los robots R_i y R_j que satisfaga las siguientes condiciones:

- V_{ij} es una función suave de φ_{ij}.
- $V_{ij} > 0$ y $V_{ij} = 0$ solo si $\varphi_{ij} = 0$.
- $V_{ij} \to \infty$ cuando $\beta_{ij} = \varphi_{ij} + d_{ij} \to 0$.
- $\frac{\partial V_{ij}}{\partial \varphi_{ij}} = \varphi_{ij}\Gamma_{ij}$ con Γ_{ij} una función suave de φ_{ij}.

Un ejemplo de FPD-AR es el reportado en [43] dado por

$$V_{ij} = \frac{\varphi_{ij}^2}{\varphi_{ij} + d_{ij}} \tag{3.25}$$

de donde

$$\Gamma_{ij} = \frac{\varphi_{ij} + d_{ij}}{(\varphi_{ij} + d_{ij})^2} \tag{3.26}$$

Note que la derivada con respecto al tiempo de V_{ij}, usando (3.23) puede ser expresada como

$$\dot{V}_{ij} = \left[\frac{\partial V_{ij}}{\partial \varphi_{ij}}\right] \left[\frac{\partial \varphi_{ij}}{\partial \beta_{ij}}\right] \dot{\beta}_{ij} = \varphi_{ij} A_{ij} \left(\dot{\xi}_j - \dot{\xi}_i\right), \tag{3.27}$$

donde

$$A_{ij} = [\Gamma_{ij} \cos\theta_{ij}, \Gamma_{ij} \sin\theta_{ij}] \tag{3.28}$$

Finalmente, observe por la figura 3.11, que se satisface $\cos\theta_{ij} = -\cos\theta_{ji}$, $\sin\theta_{ij} = -\sin\theta_{ji}$ y por lo tanto se cumple la propiedad de antisimetría $A_{ij} = -A_{ji}$.

Con la definición de distancia y orientaciones relativas, gráfica de comunicación y potenciales de distancia, la siguiente sección muestra un la ley general de control aplicables al grupo de robots que logran el avance en formación de un SRMM modular. También se dan subcasos de cómo puede ser adaptada para logra la convergencia a una formación estática o comportamientos de acoplamiento y desacoplamiento de los robots terrestres con el robot aéreo líder.

3.2.4. Formación con seguimiento de trayectoria

El caso más general de trabajo colaborativo de los SRMM que componen a un robot multientorno modular es el despliegue de los robots en un patrón de formación deseado basado en distancias, siguiendo una trayectoria de marcha y con una topología de comunicación definida en la Sección 3.2.1, desde cualquier posición inicial de los robots.

De esta forma, se busca que el drone permanezca en el aire a una altura definida constante z_n^*, mientras que su posición en el plano XY mantiene una formación por distancia respecto a los robots terrestres. Asumiendo que el robot aéreo líder R_n es el único dotado de un sensor de posicionamiento global, como un sensor GPS para el caso de exteriores, entonces es deseable que la posición XY del drone converja a una trayectoria de marcha dada por $m(t) = [m_x(t), m_y(t)]^T$. También es deseable que los ángulos de orientación ψ_i converjan a un mismo valor ψ^*. Con los requerimientos anteriores, el planteamiento del problema puede resumirse de la siguiente manera.

Planteamiento del problema. *El objetivo de control es diseñar para el sistema* (3.17)-(3.18) *un conjunto de entradas de control*

$$\Delta_1 = [u_1, ..., u_n, \omega_1, ..., \omega_{n-1}, \tau_{\psi_n}, u_{nz}], \tag{3.29}$$

tal que satisfaga simultáneamente

- $\lim_{t\to\infty} \varphi_{ij} = 0$, $\forall j \in N_i$, *midiendo los valores de* β_{ij}, θ_{ij}, $\forall j \in N_i$.
- $\beta_{ij} \neq 0$, $\forall i, j \in N$, *evitando colisión entre los robots.*
- $\lim_{t\to\infty} (\psi_i - \psi^*) = 0$, $\forall i \in N$, *i.e. que los ángulos de orientación lleguen a un valor común de* ψ^*.
- $\lim_{t\to\infty} (\xi_n - m(t)) = 0$, *i.e. que el robot aéreo converja a la trayectoria* $m(t)$.

- $\lim_{t\to\infty}(z_n - z_n^*) = 0$, *i.e. que el robot aéreo converja a una altura constante dada por* $z_n^* \in \Re$.

Teorema. *Considere el grupo de robots dado por* (3.17)-(3.18) *comunicados con una gráfica de comunicación dada por* (3.19) *y una específica FDP-AR. Defina la ley de control*

$$\Delta_1 : \begin{cases} u_i = \varrho_p A_i^T \varphi_i + \dot{m}, \quad i = 1, ..., (n-1) \\ u_n = A_n^T \varphi_n - \varrho_m(\xi_n - m) - \varrho_d(p_n - \dot{m}) + \ddot{m} \\ \omega_i = -\varrho_\omega(\psi_i - \psi^*), \quad i = 1, ..., (n-1) \\ \tau_{\psi_n} = -p_\psi - \varrho_\omega(\psi_n - \psi^*) \\ u_{nz} = -p_z - \varrho_h(z_n - z_n^*) \end{cases} \tag{3.30}$$

donde $\varrho_p, \varrho_m, \varrho_d, \varrho_\omega, \varrho_h \in \mathbb{R}$ *con* $\varrho_p, \varrho_m, \varrho_d, \varrho_\omega, \varrho_h > 0$,

$$A_i = \begin{bmatrix} A_{i1} \\ \vdots \\ A_{in} \end{bmatrix} \in \Re^{n\times 2}, \quad \varphi_i = \begin{bmatrix} a_{i1}\varphi_{i1} \\ \vdots \\ a_{in}\varphi_{in} \end{bmatrix} \in \Re^n \tag{3.31}$$

donde $a_{ij} = 1$ *si* $(j,i) \in E$ *y* $a_{ij} = 0$ *en otro caso, y* p_n, p_ψ *y* p_z *definidos en* (3.18). *Note que la adición de* a_{ij} *elimina los componentes de los errores de distancia entre los robots que no tienen comunicación en la gráfica de formación.*

Suponga que:

- *Los robots no son colineales y no colisionan en* $t = 0$, *y*
- *El equilibrio ocurre solo cuando* $\varphi_{ij} = 0$, $\forall i, j = 1, ..., n$.

Entonces en el sistema en lazo cerrado (3.17), (3.18), (3.30) *los robots satisfacen los requerimientos del Planteamiento del problema 1.*

Observación 1. *La ley de control* (3.30) *implica que el* R_n *es el único robot con la información completa acerca de la trayectoria de referencia. Las leyes de control del resto de los robots seguidores son definidas para ser dependientes de datos de distancia y orientación y la derivada de la trayectoria de marcha.*

Prueba. *Considere la funcion candidata de Lyapunov*

$$\begin{aligned} V = &\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} V_{ij} + \frac{1}{2}\|p_n - \dot{m}\|^2 + \frac{\rho_m}{2}\|\xi_n - m\|^2 + \frac{1}{2}\sum_{i=1}^{n-1}\|\psi_i - \psi^*\|^2 \\ &+ \frac{\rho_\omega}{2}\|\psi_n - \psi^*\|^2 + \frac{1}{2}\|p_\psi\|^2 + \frac{\rho_h}{2}\|z_n - z_n^*\|^2 + \frac{1}{2}\|p_z\|^2 \end{aligned} \tag{3.32}$$

donde V_{ij} *es una FDP-AR definida positiva. Note que* V *es siempre positiva y se desvanece solo cuando los robots cumplen con la formacion deseada, i.e. cuando* $\varphi_{ij} = 0$, $\forall j \in N_i$, *cuando el robot lider converje a la trayectoria y altura deseada,*

i.e. $\xi_n = m(t)$, $z_n = z_n^*$ *y cuando los ángulos converjen al valor deseado, i.e.* $\psi_i = \psi^*$, $i = 1, ..., n$. *La derivada de* V *se calcula como*

$$\dot{V} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\dot{V}_{ij} + (p_n - \dot{m})^T (\dot{p}_n - \ddot{m}) + \rho_m (\xi_n - m)^T \left(\dot{\xi}_n - \dot{m}\right) + \sum_{i=1}^{n-1} (\psi_i - \psi^*)^T \dot{\psi}_i + \rho_\omega (\psi_n - \psi^*)^T \dot{\psi}_n + p_\psi^T \dot{p}_\psi + \rho_h (z_n - z_n^*)^T \dot{z}_n + p_z^T \dot{p}_z \tag{3.33}$$

Sustituyendo (3.17) *and* (3.18) *en* (3.33), *se obtiene*

$$V = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\dot{V}_{ij} + (p_n - \dot{m})^T (u_n - \ddot{m}) + \rho_m (\xi_n - m)^T (p_n - \dot{m}) + \sum_{i=1}^{n-1} (\psi_i - \psi^*)^T \omega_i + \rho_\omega (\psi_n - \psi^*)^T p_\psi + p_\psi^T \tau_{\psi_n} + \rho_h (z_n - z_n^*)^T p_z + p_z^T u_{nz} \tag{3.34}$$

Analizando el primer término de la ecuación (3.34), *note que sustituyendo la ecuación* (3.27) *resulta en*

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\dot{V}_{ij} = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\varphi_{ij}A_{ij}\dot{\xi}_j\right] - \sum_{i=1}^{n}\sum_{j=1}^{n}\left[\varphi_{ij}A_{ij}\dot{\xi}_i\right]\right) \tag{3.35}$$

Debido a que la gráfica de comunicación es no dirigida, entonces $\varphi_{ij} = \varphi_{ji}$ *y* $A_{ij} = -A_{ji}$, *y la ecuación previa puede ser reducida a*

$$\begin{aligned}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\dot{V}_{ij} &= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\varphi_{ij}A_{ij}\dot{\xi}_j\right] + \sum_{j=1}^{n}\sum_{i=1}^{n}\left[\varphi_{ji}A_{ji}\dot{\xi}_i\right]\right) \\ &= \frac{1}{2}\left(2\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\varphi_{ij}A_{ij}\dot{\xi}_j\right]\right) = \sum_{j=1}^{n}\left[\sum_{i=1}^{n}\left(\varphi_{ji}A_{ij}\right)\dot{\xi}_j\right] \\ &= -\sum_{j=1}^{n}\left[\sum_{i=1}^{n}\left(\varphi_{ji}A_{ji}\right)\dot{\xi}_j\right]\end{aligned} \tag{3.36}$$

Note que el término $\sum_{i=1}^{n}[\varphi_{ji}A_{ij}]$ *puede ser compactado como*

$$\begin{aligned}\sum_{i=1}^{n}[\varphi_{ji}A_{ij}] &= \begin{bmatrix} \varphi_{j1} & \varphi_{j2} & \cdots & \varphi_{jn} \end{bmatrix}\begin{bmatrix} A_{1j} \\ A_{2j} \\ \vdots \\ A_{nj} \end{bmatrix} = -\begin{bmatrix} \varphi_{j1} & \varphi_{j2} & \cdots & \varphi_{jn} \end{bmatrix}\begin{bmatrix} A_{j1} \\ A_{j2} \\ \vdots \\ A_{jn} \end{bmatrix} \\ &= -\varphi_j^T A_j,\end{aligned}$$

Por lo tanto

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\dot{V}_{ij} = -\sum_{j=1}^{n}\left[\varphi_j^T A_j \dot{\xi}_j\right] \tag{3.37}$$

Sustituyendo (3.37) *en* (3.34)*, pero cambiando el subíndice* j *por* i*, la derivada de* V *puede ser reducida a*

$$\dot{V} = -\sum_{i=1}^{n}\left[\varphi_i^T A_i \dot{\xi}_i\right] + (p_n - \dot{m})^T (u_n - \ddot{m}) + \rho_m (\xi_n - m)^T (p_n - \dot{m}) + \sum_{i=1}^{n-1} (\psi_i - \psi^*)^T \omega_i + \rho_\omega (\psi_n - \psi^*)^T p_\psi + p_\psi^T \tau_{\psi_n} + \rho_h (z_n - z_n^*)^T p_z + p_z^T u_{nz} \tag{3.38}$$

La sumatoria en el primer término de la ecuación (3.38) *puede ser dividida de acuerdo al orden de los robots dado en* (3.17) *y* (3.18)*. Entonces,*

$$\begin{aligned}\dot{V} = &-\sum_{i=1}^{n-1}\left[\varphi_i^T A_i u_i\right] - \varphi_n^T A_n p_n \\ &+ (p_n - \dot{m})^T (u_n - \ddot{m}) + \rho_m (\xi_n - m)^T (p_n - \dot{m}) + \sum_{i=1}^{n-1} (\psi_i - \psi^*)^T \omega_i \\ &+ \rho_\omega (\psi_n - \psi^*)^T p_\psi + p_\psi^T \tau_{\psi_n} + \rho_h (z_n - z_n^*)^T p_z + p_z^T u_{nz}\end{aligned} \tag{3.39}$$

La ley de control (3.30) *puede ser reemplazada en la ecuación* (3.39)*, obteniéndose*

$$\begin{aligned}\dot{V} = &-\sum_{i=1}^{n-1}\left[\varphi_i^T A_i \left(\varrho_p A_i^T \varphi_i + \dot{m}\right)\right] - \varphi_n^T A_n p_n \\ &+ (p_n - \dot{m})^T \left(\left[A_n^T \varphi_n - \varrho_m(\xi_n - m) - \varrho_d(p_n - \dot{m}) + \ddot{m}\right] - \ddot{m}\right) \\ &+ \rho_m (\xi_n - m)^T (p_n - \dot{m}) - \varrho_\omega \sum_{i=1}^{n-1} (\psi_i - \psi^*)^T (\psi_i - \psi^*) \\ &+ \rho_\omega (\psi_n - \psi^*)^T p_\psi + p_\psi^T (-p_\psi - \varrho_\omega(\psi_n - \psi^*)) + \rho_h (z_n - z_n^*)^T p_z + p_z^T (-p_z - \varrho_h(z_n - z_n^*))\end{aligned}$$

Note que algunos términos son mutuamente cancelados. Por lo tanto, la expresión anterior se reduce a

$$\begin{aligned}\dot{V} = &-\varrho_p \sum_{i=1}^{n-1}\left[\left(\varphi_i^T A_i\right)\left(A_i^T \varphi_i\right)\right] - \sum_{i=1}^{n-1}\left[\left(\varphi_i^T A_i\right)\dot{m}\right] - \varphi_n^T A_n p_n + p_n^T A_n^T \varphi_n - \dot{m}^T A_n^T \varphi_n \\ &- \varrho_d \|p_n - \dot{m}\|^2 - \varrho_\omega \sum_{i=1}^{n-1} \|\psi_i - \psi^*\|^2 - \|p_\psi\|^2 - \rho_h \|z_n - z_n^*\|^2 - \|p_z\|^2\end{aligned}$$

Agrupando algunos términos en la ecuación previa, entonces

$$\begin{aligned}\dot{V} = &-\varrho_p \sum_{i=1}^{n-1} \left\|A_i^T \varphi_i\right\|^2 - \varrho_d \|p_n - \dot{m}\|^2 - \varrho_\omega \sum_{i=1}^{n-1} \|\psi_i - \psi^*\|^2 - \|p_\psi\|^2 - \rho_h \|z_n - z_n^*\|^2 \\ &- \|p_z\|^2 - \sum_{i=1}^{n-1}\left[\varphi_i^T A_i \dot{m}\right] - \dot{m}^T A_n^T \varphi_n - \varphi_n^T A_n p_n + p_n^T A_n^T \varphi_n\end{aligned} \tag{3.40}$$

Note que $\left(\dot{m}^T A_n^T \varphi_n\right) = \left(\varphi_n^T A_n \dot{m}\right)^T$ *produce el mismo valor real. Similar para el caso de* $\left(p_n^T A_n^T \varphi_n\right) = \left(\varphi_n^T A_n p_n\right)^T$*. Entonces los dos últimos de la ecuación* (3.40)

son cancelados y

$$\dot{V} = -\varrho_p \sum_{i=1}^{n-1} \left\| A_i^T \varphi_i \right\|^2 - \varrho_d \left\| p_n - \dot{m} \right\|^2 - \varrho_\omega \sum_{i=1}^{n-1} \left\| \psi_i - \psi^* \right\|^2 - \left\| p_\psi \right\|^2 - \rho_h \left\| z_n - z_n^* \right\|^2 - \left\| p_z \right\|^2 - \sum_{i=1}^{n} \left[\varphi_i^T A_i \right] \dot{m}$$

Observe que el último término $\sum_{i=1}^{n} \left[\varphi_i^T A_i \right] \dot{m} = 0$, *porque la suma de todos los términos* $\varphi_i^T A_i$ *es cero para una gráfica de comunicación no dirigida, debido a que* $A_{ij} = -A_{ji}$ *en cualquier par de robots que se comunican. Finalmente*

$$\begin{aligned} \dot{V} = & -\varrho_p \sum_{i=1}^{n-1} \left\| A_i^T \varphi_i \right\|^2 - \varrho_d \left\| p_n - \dot{m} \right\|^2 - \varrho_\omega \sum_{i=1}^{n-1} \left\| \psi_i - \psi^* \right\|^2 \\ & - \left\| p_\psi \right\|^2 - \rho_h \left\| z_n - z_n^* \right\|^2 - \left\| p_z \right\|^2 \end{aligned} \tag{3.41}$$

Note que $\dot{V}$ *es semidefinida negativa. El equilibrio ocurre cuando simultáneamente* $A_i^T \varphi_i = 0$, $i = 1, ..., n-1$, $p_n = \dot{m}$, $\psi_i = \psi^*$, $i = 1, ..., n-1$, $p_\psi = 0$, $z_n = z_n^*$ *y* $p_z = 0$. *Para la primera condición* $A_i^T \varphi_i = 0$, $i = 1, ..., n-1$ *implica que* $\varphi_i = 0$ *por la suposición inical del Teorema 3.2.4.*

Usando el Teorema de Invarianza de LaSalle, note que sustitutyendo estos equilibrios en el sistema en lazo cerrado (3.17), (3.18), (3.30). *Entonces*

$$\dot{\xi}_i = u_i = \dot{m}, i = 1, ..., (n-1) \tag{3.42}$$

$$\ddot{\xi}_n = u_n = A_n^T \varphi_n - \varrho_m (\xi_n - m) + \ddot{m} \tag{3.43}$$

$$\dot{\psi}_i = \omega_i = 0, i = 1, ..., (n-1) \tag{3.44}$$

$$\ddot{\psi}_n = \tau_{\psi_n} = -\varrho_\omega (\psi_n - \psi^*) \tag{3.45}$$

$$\ddot{z}_n = u_{nz} = 0 \tag{3.46}$$

Note que la velocidad de los primeros $n-1$ *robots converge a* $\dot{m}$ *en* (3.43). *Por lo tanto todos los robots convergen a una misma velocidad. Por otro lado, desde que* $p_n = \dot{m}$, *entonces* $\dot{p}_n = \ddot{\xi}_n = \ddot{m}$. *Entonces la ecuación* (3.43) *satisface*

$$0 = A_n^T \varphi_n - \varrho_m (\xi_n - m) \tag{3.47}$$

Consecuentemente, por la suposición inicial del Teorema 3.2.4, si los errores de formación $\varphi_i = 0$, $i = 1, ..., n-1$ *han convergido debido a que* $\varphi_i = 0$, $i = 1, ..., n-1$, *entonces debido a que la gráfica de comunicación es no dirigida y rígida, entonces necesariamente* $A_n^T \varphi_n = 0$ *cuando* $\varphi_n = 0$. *Entonces la ecuación* (3.47) *puede ser simplificada a* $\varrho_m (\xi_n - m) = 0$, *lo cual muestra que la trayectoria del robot aérero converge a* $m(t)$.

Note que la velocidad de ψ, $i = 1, ..., (n-1)$ *en* (3.44) *es cero. Por lo tanto estos ángulos convergen a una valor constante dado por el equilibrio* $\psi_i = \psi^*$. *Dado que en equilibrio se cumple que* $p_\psi = 0$, *entonces* $\dot{p}_\psi = \ddot{\psi}_n = 0$. *Por lo tanto, en la ecuación* (3.45), $\varrho_\omega (\psi_n - \psi^*) = 0$, *por lo tanto* $\psi_n = \psi^*$ *y todos los robots convergen a la misma orientación dada por* ψ^*. *La ecuación* (3.47) *muestra que* z_n *es detenido*

en el equilibrio z_n^*. Esto concluye que los robots cumplen con la formación y el seguimiento de trayectoria.

Finalmente, la función de Lyapunov tiende a infinito cuando algún de los $\beta_{ij} \to 0$. Debido a que los robots no colisionan en $t = 0$, por la suposición inicial y $\dot{V} \leq 0$, es claro que los robot pueden acercarse entre ellos pero β_{ij} $\forall i, j \in N$ nunca será igual a cero y por lo tanto los robots no colisionan. Esto completa la prueba del teorema.

Note que un equilibrio no deseado ocurre $A_i^T \varphi_i = 0$, $i = 1, ..., n-1$, y $\varphi_i \neq 0$. Esto ocurre cuando los robots son colineales. Sin embargo, por la suposición inicial, los robots no son colineales y no están atrapados en el mínimo local. ■

Simulacion numérica de líder aéreo con seguidores terrestres

La figura 3.12 muestra una simulacion del sistema en lazo cerrado (3.17), (3.18), (3.30), con $n = 4$. El grafo de comunicacion basado en distancias esta completo cuando las distancias deseadas estan dadas por $d_{12} = d_{23} = d_{31} = 5$ y $d_{14} = d_{24} = d_{34} = \frac{5}{2\cos(\frac{\pi}{6})}$, i.e., el lider es posicionado en el centro de un triangulo equilatero, en donde sus lados miden $5m$. La trayectoria de marcha esta dada por $m(t) = \left[10\sin(\frac{2\pi}{160}t), 10\cos(\frac{2\pi}{160}t)\right]$.

Los parametros de control son $\rho_p, \rho_d, \rho_m = 1$. Las trayectorias de los robots en el plano se muestran en 3.12(a), donde los simbolos O representan las posiciones iniciales.

Algunas posturas de los robots se representan en los instantes de tiempo $t = 40$, $t = 75$, $t = 100$, $t = 125$ y $t = 150$, respectivamente. Observe que los robots convergen a la formación deseada, y el líder sigue la trayectoria $m(t)$ como se muestra en las figuras 3.12(b) y 3.12(c), donde los errores de formacion φ_{ij} y el error de marcha de R_4, i.e. $e_m = \xi_4 - m$ convergen a zero, respectivamente. Tenga en cuenta que los robots no chocan porque los valores de φ_{ij} en la figura 3.12(b) no son menos de -5 para el caso de $\varphi_{12}, \varphi_{23}, \varphi_{31}$ o $-\frac{5}{2\cos(\frac{\pi}{6})} = -2{,}88$ para el caso de $\varphi_{14}, \varphi_{24}, \varphi_{34}$. Finalmente, las entradas de control se representan en la figura 3.12(d).

3.2.5. Formación con convergencia a un punto

En esta sección, la ley de control definida en la ecuación 3.30 es modificada para el caso de formación con referencia a un punto fijo en el plano $m \in \Re^2$. Esta nueva ley de control puede ser vista como un caso particular de la ley 3.30. Note que al ser un punto fijo, P $\dot{m} = [0, 0]^T$, por lo que la nueva ley de control Δ_2 puede obtenerse

de la siguiente manera:

$$\Delta_2 : \begin{cases} u_i = \varrho_p A_i^T \varphi_i, \quad i = 1, ..., (n-1) \\ u_n = A_n^T \varphi_n - \varrho_m(\xi_n - m) - \varrho_d p_n \\ \omega_i = -\varrho_\omega(\psi_i - \psi^*), \quad i = 1, ..., (n-1) \\ \tau_{\psi_n} = -p_\psi - \varrho_\omega(\psi_n - \psi^*) \\ u_{nz} = -p_z - \varrho_h(z_n - z_n^*) \end{cases} \tag{3.48}$$

Es claro que al ser un caso particular de la ecuación 3.30, la prueba de convergencia es válida también para este caso.

Simulacion numérica de convergencia a un punto

Al igual que el caso anterior, la figura 3.13 muestra una simulación del sistema en lazo cerrado (3.17), (3.18), (3.48), con $n = 4$. Donde las distancias deseadas son las mismas que el caso anterior, i.e., el lider es posicionado en el centro de un triángulo equilátero en donde sus lados miden $5m$. El punto a converger esta dado por $m = [5, 5]$.

Los parametros de control son $\rho_p, \rho_d, \rho_m = 1$. Las trayectorias de los robots en el plano se muestran en la figura 3.13(a). Observe que los robots convergen a la formación deseada y el líder converge al punto $[5, 5]$ como se refleja en la figura 3.13(b), donde los errores de formacion φ_{ij} convergen a zero. Finalmente, las entradas de control se representan en la figura 3.13(c).

3.2.6. Formación con convergencia a un punto: ajuste para unión y separación del robot aéreo

Es claro notar que la ley de control (3.48) puede ser utilizada para que, una vez que los robots converjan a la formación estática, la altura del drone pueda ser modificada con el propósito de acercarse o alejarse de los robots terrestres. Esto simplemente se logra con definir la altura deseada $z_n^* = 0$, lo cual provoca que el robot aéreo se una a la formación de los robots terrestres. Este fenómeno es estudiado en el área de sistemas multirobot como fenómeno de unión (merge por su término en inglés). Conceptualmente puede ser utilizado para que el robot aéreo descanse sobre los robots terrestres y éstos últimos puedan trasladarlo. Para este caso en particular se modificó la ecuación 3.48 con $z_n^* = 0$ generando una nueva ley de control dada por:

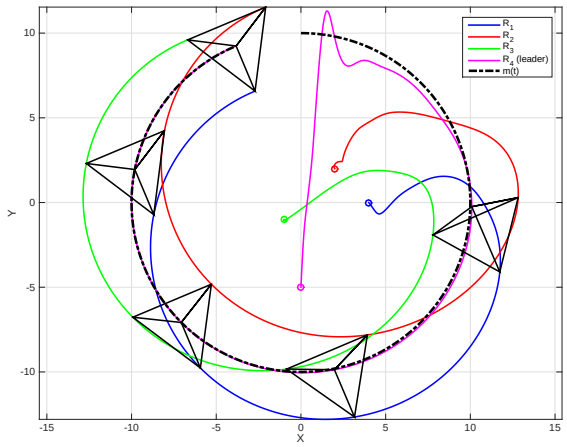
$$\Delta_3 : \begin{cases} u_i = \varrho_p A_i^T \varphi_i, \quad i = 1, ..., (n-1) \\ u_n = A_n^T \varphi_n - \varrho_m(\xi_n - m) - \varrho_d p_n \\ \omega_i = -\varrho_\omega(\psi_i - \psi^*), \quad i = 1, ..., (n-1) \\ \tau_{\psi_n} = -p_\psi - \varrho_\omega(\psi_n - \psi^*) \\ u_{nz} = -p_z - \varrho_h z_n \end{cases} \tag{3.49}$$

El caso contrario es que el robotr aéreo vaya de $z_n^* = 0$ a un valor constante positivo de altura deseada. Esto describe un fenómeno de separación (split) del robot aéreo con los robots terrestres, que puede ser alcanzado por la ley de control anteriormente presenta en la ecuación (3.48).

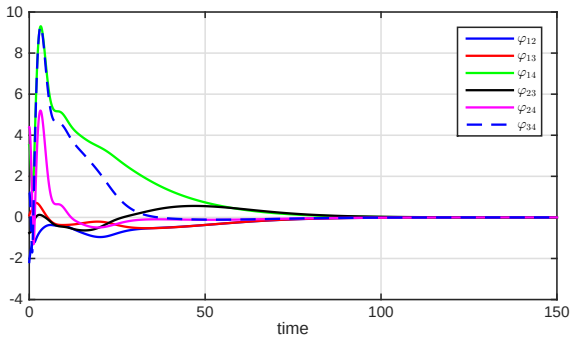
Observe que la combinación de las leyes de control (3.30), (3.48) y 3.49 pueden alternarse para lograr los siguientes comportamientos de robots multientorno.

- Operaciones de exploración de entornos con maximimización del sensado del espacio de trabajo. Esto se logra ehjecutando la ley de control (3.48) para lograr un formación terrestres-aérea y posteriormente la ley (3.30) para que los robots avancen en formación explorando el espacio de trabajo desde sus posiciones estratégicas.
- Traslado aéreo de los robots terrestres. Se logra con implementar secuencialmente las leyes (3.48), (3.49), (3.48) y (3.30), donde los robots se forman, el drone desciente en operación de merge, sujeta a los robots terrestre, sube a una altura constante y los traslada siguiendo una trayectoria deseada. Esto podría ser útil cuando los robots terrestres no puedan avanzar por obstáculos en el medio y requieran el apoyo del drone líder.
- Traslado del robot áereo por parte de los robots terrestres. Esto se logra con la secuencia de control (3.48), (3.49), (3.30), donde los robots convergen a formación, el robot áereo desciende a tierra para ser cargado por los robot terrestre siguiendo una trayectoria deseada. Esto se utiliza cuando el robot aéreo necesita apoyo de traslado terrestre por falta de batería, ahorro de energía o porque requiere cruzar por un pasaje estrecho a nivel piso.

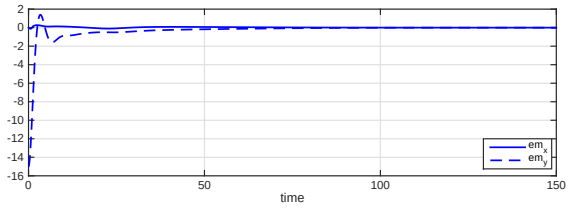
Para ilustrar las operaciones anteriores, en el siguiente capítulo se presenta la validación experiemental de las leyes de control (3.30), (3.48) y (3.49) para ilustrar su aplicación multientorno.



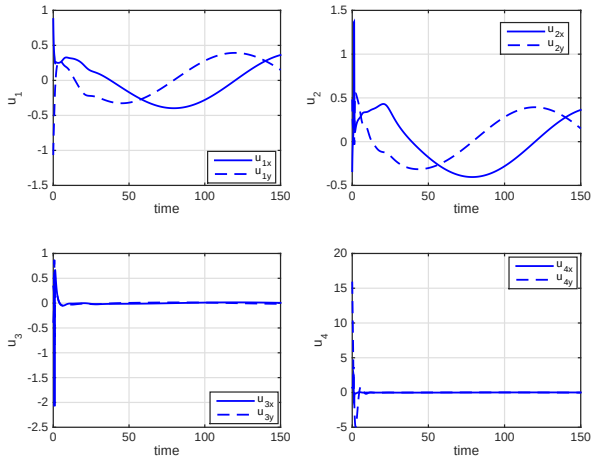
(a) Trayectoria de los robots



(b) Errores de formación

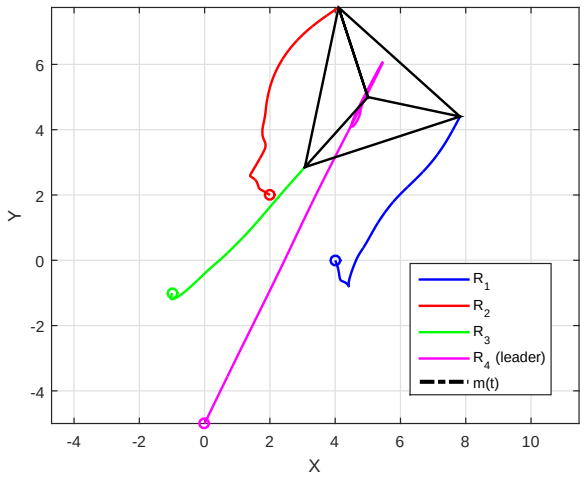


(c) Error de marcha del robot líder

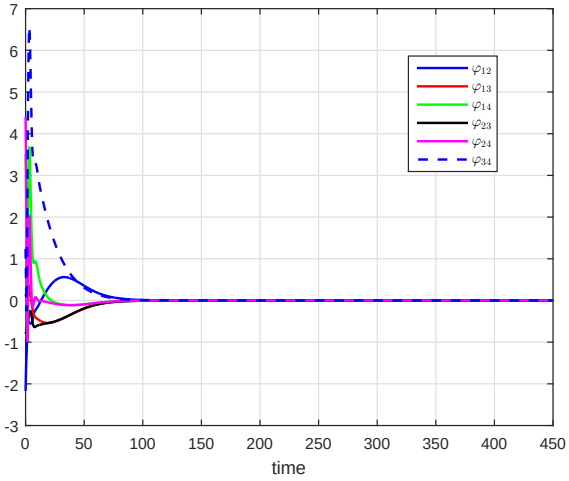


(d) Entradas de control

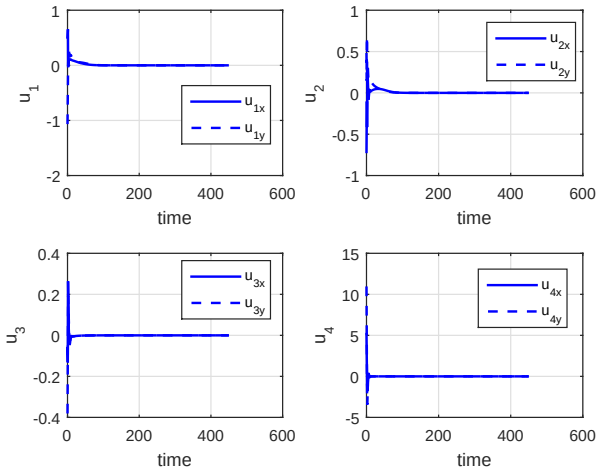
Figura 3.12: Simulación de seguimiento de formación con $n = 4$



(a) Trayectoria de los robots



(b) Errores de formación



(c) Entradas de control

Figura 3.13: Simulación de formación con concvergencia a un punto

Capítulo 4

Trabajo experimental

4.1. Descripción del área de trabajo

La figura 4.1 muestra el laboratorio de análisis de movimiento de la Universidad Iberoamericana Ciudad de México en donde se desarrollaron todos los experimentos reportados en esta tesis.

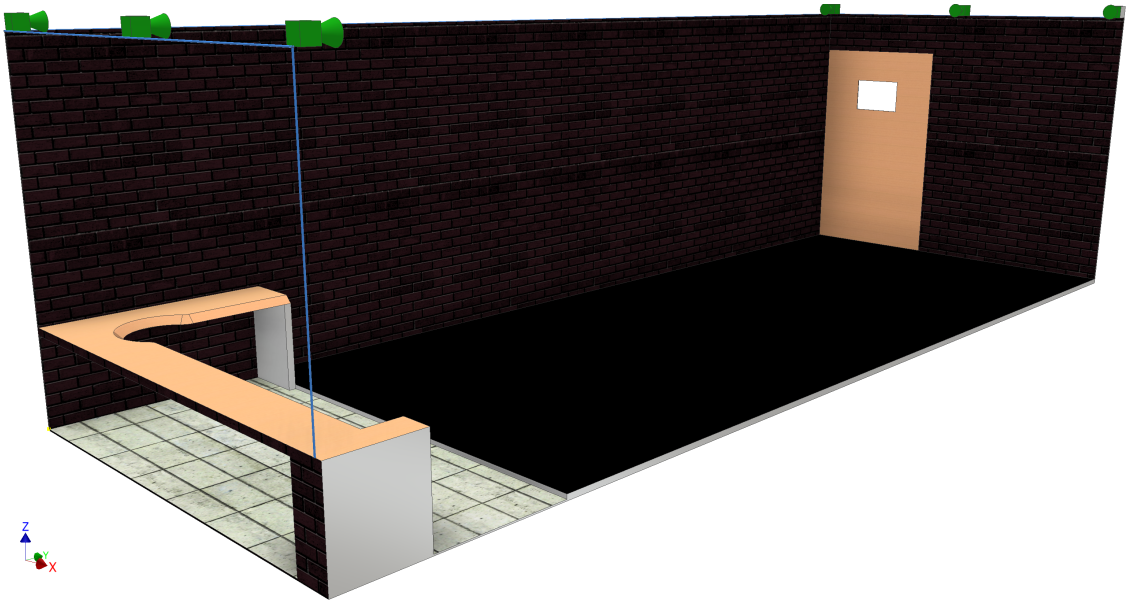


Figura 4.1: Laboratorio de análisis de movimiento

Este laboratorio cuenta con un área de trabajo de 3×7m. Está compuesto por seis cámaras de la serie Bonita® de la marca Vicon® ubicadas a 2,45m del piso. Para un mejor funcionamiento tanto de las cámaras como de la tracción de los robots físicos, el piso se encuentra cubierto de un material no reflejante y suave. Asimismo, se bloquean todas las entradas de luz solar para eliminar el ruido que puedan llegar a causar.

Estas cámaras emiten luz infrarroja, la cual se refleja en unos marcadores de forma esférica de aproximadamente 14,5mm de diámetro. El promedio de captura de imagenes es de 117 cuadros por segundo (FPS) en este sistema. Para conocer la posicion de los marcadores, se utiliza un mínimo de tres cámaras para triangular su posición en el espacio de trabajo. En este punto la información se envía por medio de cables ethernet a una computadora central. Esta computadora cuenta con las siguientes carqacteristicas: sistema operativo Windows 7 profesional de 64 bits, procesador Intel $i7$ modelo 4770 a 3,4GHz, con 16Gb de memoria RAM. También cuenta con el programa "Tracker®" donde se encuentran definidos los cuerpos rígidos a ubicar y que están compuestos por al menos tres marcadores. Es aquí en donde se procesa la información que entregan las cámaras y se calculan los datos de ubicación de los SRM en el espacio de trabajo.

Una vez procesados los datos por Tracker® se utiliza un cliente virtual desarrollado por Vicon® y re-implementado en esta tesis con programación orientada a objetos para su uso mas simple. Este provee los datos de ubicación espacial de todos los SRM en tiempo real. Esta información se utiliza como la retroalimentación del sistema, para calcular las distancias que existen entre los pares de SRM y así implementar las leyes de control propuestas.

4.2. Descripción de la plataforma experimental

En esta sección se presenta el diseño mecánico y electrónico desarrollado como prueba de concepto para esta tesis.

4.2.1. Diseño del núcleo de control

Se diseñó un núcleo de control para robots heterogéneos ya que los resultados presentados en este proyecto son ilustrativos mas no limitativos en cuanto a las capacidades de la ley de control propuesta. Es por esta razón que la ley de control implementada puede aplicarse a distintos robots omnidireccionales, como se marcó en los objetivos de esta tesis.

También se diseñó un núcleo de control electrónico universal cuyo PCB se muestra en la figura **??**, que permita controlar un gran número de posibles entes robóticos apliquen la ley de control y de esta forma pueda ser implementada fácilmente. El núcleo desarrollado lleva por nombre "HUEX" del náhuatl huexocauhtli, que significa pato silbón. Su nombre de debe a la similitud que existe entre las capacidades multientorno que poseen los patos y el SRMM modular que se presenta en este trabajo.

El HUEX cuenta con las siguientes características:

- Un MCU de la marca NXP modelo $LPC1768$.
  Con un procesador ARM® Cortex™-M3 Core

96MHz, 32KB RAM, 512KB FLASH

Soporte para la plataforma MBED®

Cuatro LED's

- Comunicación buetooth por medio del módulo HC-06
- Seis entradas analógicas
- Seis salidas PWM
- Una salida analógica
- Doce pines I/O digitales
- Salida para comunicación serial/$I2C$
- Salida serial de tipo half duplex asíncrona

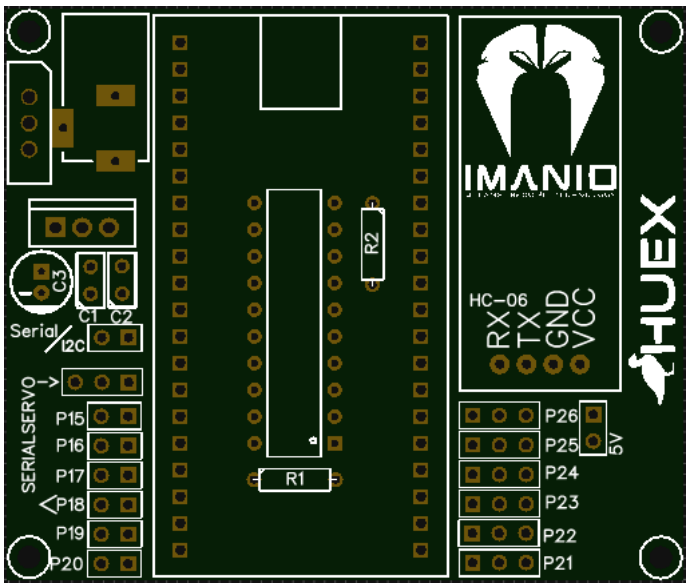


Figura 4.2: HUEX

Se observa también que las caracteristicas que presenta esta tarjeta electrónica sobrepasan ampliamente los requerimientos del SRM presentado en este proyecto. Esto se debe a que la tarjeta está preparada para implementar las leyes de control a bordo. Tambien es posible sensar distintos entornos dinamicos no estructurados por medio de una amplia gama de sensores disponibes en el mercado actual, que pueden ser conectados en los puertos disponibles de HUEX. El diagrama de el HUEX y las bibliotecas desarrolladas para el mismo se encuentran en el apendice B. La programación es en lenguaje $C++$ programada en el software en línea de productos $MBED$.

4.2.2. Diseño y construcción de robots

La plataforma robótica construida para este proyecto consta de tres SRM de tipo terrestre omnidireccional de tres ruedas y un UAV de tipo tricóptero.

SRM Terrestres

El armazón principal se conforma de acrílico de 4,5mm y estireno de 1mm. La estructura de las llantas es de acrílico de 4,5mm y las llantas omnidireccionales de acrílico de 3mm con ejes de acero inoxidable de 0,7mm. La estructura y las llantas fueron diseñadas en su totalidad. El proceso de maquinado utilizado para crear las piezas nesesarias fue corte con laser de dioxido de carbono.

Se utilizaron servomotores de la marca "Power HD" con las siguientes especificaciones:

- Modelo AR-3606HB
- Rango de voltaje tolerado de 4,8v a 6v
- Velocidad máxima de 70rpm
- Torque máximo de 6,7Kg/cm
- Peso de 40g
- Dimensión de $40{,}5 \times 20 \times 38mm$.

Los SRM terrestres utilizan una batería de litio-polímero de dos celdas (7,4v), capacidad de 0,5Ah con una descarga maxima de 20C y soporte a picos de corriente de hasta 30C. Los diagramas del ensamble estructural de los robots terrestres se encuentran en el apéndice A.

4.3. Resultados experimentales

Para todos los experimentos que se muestran a continuación se utilizaron los tres robots terrestres con tres llantas omnidireccionales descritos anteriormente, cada uno como se muestra en la figura 4.3, los cuales son formados con respecto a la proyeccion en el plano XY de un drone virtual. El drone virtual cumple con toda la dinámica descrita en el capítulo anterior y se utiliza para dar un movimiento más suave de las trayectorias del drone, en vez del uso de un drone real cuyos efectos de ruido y vibraciones afectarían la dinámica de formación y seguimiento de los robots terrestres. Obviamente, el uso de un drone real puede ser soportado porlas leyes de control diseñadas y será tema de un trabajo futuro de esta tesis. Asimismo se puede encontrar el código utilizado para los experimentos en el apéndice C.

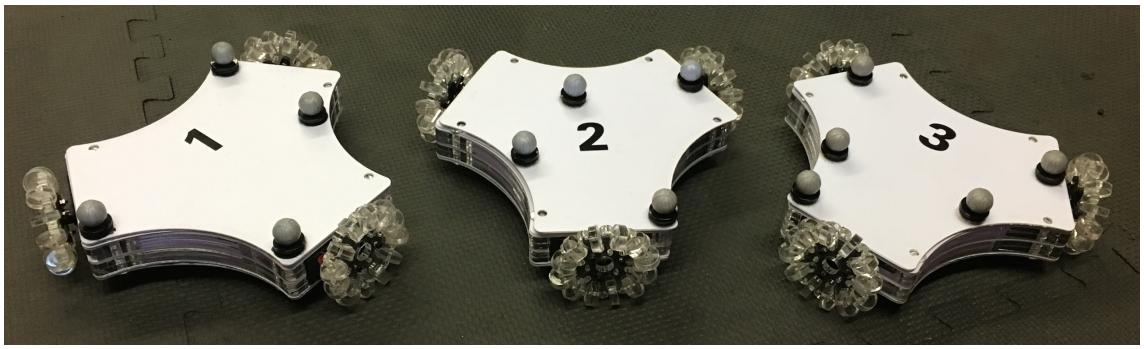


Figura 4.3: Omnidirectional wheeled mobile robot

De acuerdo con el capítulo anterior, para todo robot omnidireccional de tres ruedas moviéndose en $2D$, las velocidades angulares de cada llanta se pueden obtener de la siguiente manera.

$$\begin{bmatrix} \omega_{i1} \\ \omega_{i2} \\ \omega_{i3} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} (-0{,}5\sqrt{3}) & 0{,}5 & L \\ 0 & -1 & L \\ (0{,}5\sqrt{3}) & 0{,}5 & L \end{bmatrix} \begin{bmatrix} v_{xi} \\ v_{yi} \\ \omega_i \end{bmatrix}, \quad i = 1, .., 3 \tag{4.1}$$

donde $r = 0{,}03m$ es el radio de las llantas y $L = 0{,}05m$ es la distancia que existe entre el centro del robot y la llanta. Para el caso el drone, las velocidades angulares de sus rotores se calcularon de la siguiente manera.

$$\begin{bmatrix} \omega_{i1}^2 \\ \omega_{i1}^2 \\ \omega_{i1}^2 \\ \omega_{i1}^2 \end{bmatrix} = \frac{1}{k} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell & \ell & 0 & 0 \\ 1 & 0 & -\ell & \ell \\ -\frac{b}{k} & -\frac{b}{k} & \frac{b}{k} & \frac{b}{k} \end{bmatrix}^{-1} \begin{bmatrix} F_i \\ \tau_{\phi_i} \\ \tau_{\theta_i} \\ \tau_{\psi_i} \end{bmatrix}, \quad i = 4 \tag{4.2}$$

donde $\ell = 0{,}3m$ es la distancia al centro de masa de cada rotor, $k = 0{,}1$ es la constante de thrust y $b = 0{,}01$ es la constante de arrastre de todo el cuerpo.

4.3.1. Estrategia de líder aéreo con seguidores terrestres

En este primer experimento se valida la ley de control (3.30). El patrón de formación deseado se puede describir con: $d_{12} = d_{23} = d_{31} = 1$ y $d_{14} = d_{24} = d_{34} = \frac{0{,}5}{\cos(\frac{\pi}{6})}$ (en metros. El drone lider se encuentra siguiendo una tryectoria circular descrita por $m(t) = \left[0{,}5\sin(\frac{2\pi}{35}t), 0{,}5\cos(\frac{2\pi}{35}t)\right]$. Los parámetros de control son $\rho_p = 1000$, $\rho_d = 500$, $\rho_m = 500$, $\rho_\psi = 1$ y $\rho_h = 1$. Los valores deseados de orientación son $\psi_i^* = 0$, $i = 1, ..., 4$. la altura del drone se establece en $z_4^* = 1$m.

Las trayectorias de los robots se muestran en 4.4(a). Los errores de formacion φ_{ij} y el error de marcha para R_4 se muestra en las figuras 4.4(b) y 4.4(c), respectivamente, y las entradas de control se encuentran representadas en 4.4(d). Note que los robots alcanzan la formacion deseada. Tambien se puede apreciar un poco de ruido en las trayectorias de los robots terrestres, esto se debe a que la friccion de las llantas con el piso y a errores de medicion, entre otras variables físicas que perturban al sistema.

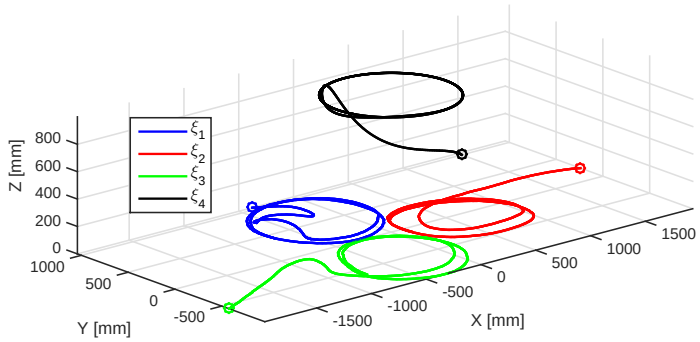
4.3.2. Estrategia de formación con convergencia a un punto

En en este segundo experimento se valida la ley de control (3.48). El patrón de formación deseado se puede describir con: $d_{12} = d_{23} = d_{31} = 1$ y $d_{14} = d_{24} = d_{34} = \frac{0{,}5}{\cos(\frac{\pi}{6})}$ (en metros). Los parámetros de control son los mismos del caso anterior, excepto $m = [0, 0]^T$ (valor de referencia fijo). Los valores deseados de orientacion son $\psi_i^* = \psi_4$, $i = 1, ..., 3$. La altura del drone se establece en $z_4^* = 1$m. Los resultados del experimento se muestran en la figura 4.5. Note que los robots convergen al punto fijo de referencia.

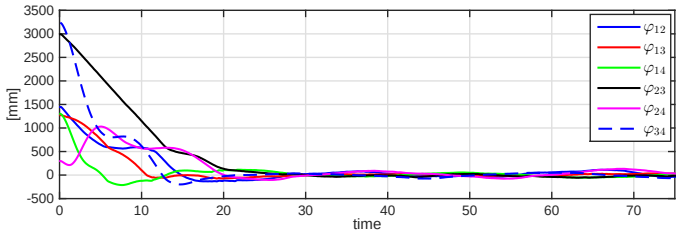
4.3.3. Estrategia de unión y separación

El experimento 3, mostrado en la figura 4.6, está dedicado a mostrar un ejemplo de unión y separación, con la ley de control (3.48), seguida de la ley de control (3.49). El patrón de formación es el mismo del caso anterior, al igual los parámetros de control. En primer lugar, el drone lider se le impone un punto a converger dado por $m(t) = [0, 0]^T$. El drone es físicamente representado por una regla de calibración del sistema VICON, emulando el movimiento del drone. De la altura de 1m, se hace descender hasta cero como se aprecia en la figura 4.6(a). Note que los robots mantienen la formación deseada. El resto de las figuras muestra los errrores de formación, los errores de ángulos y las entradas de control.

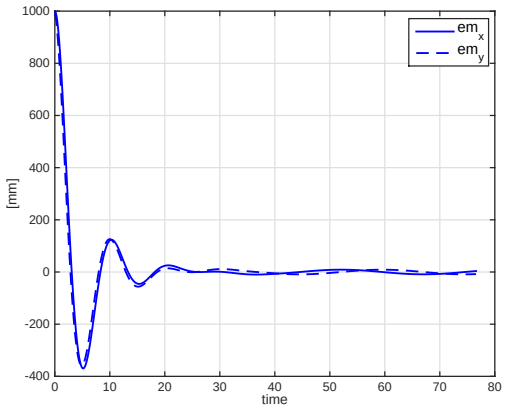
Note que el desempeño de la ley de control en los experimentos presentados cumple con los requerimientos planteados en esta tesis. Como era de esperarse, las variaciones respecto al caso ideal están estrechamente relacionados con los fenómenos de fricción no modelados, los posibles retardos en la comunicación, la resolución en la medición de las posiciones y orientaciones de los robots, entre otros. Sin embargo, esto siente un precedente para mejoras en el trabajo experimental de futuras tesis de posgra



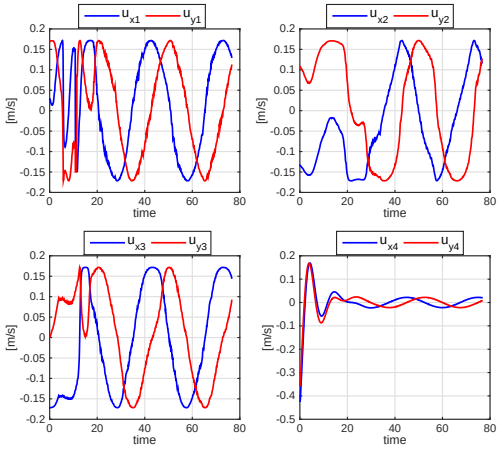
(a) Trayectorias de los robots



(b) Errores de formación

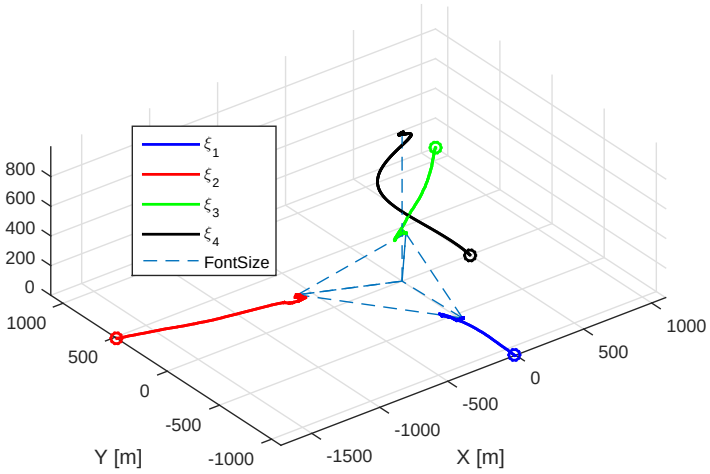


(c) Error de marcha para el robot lider

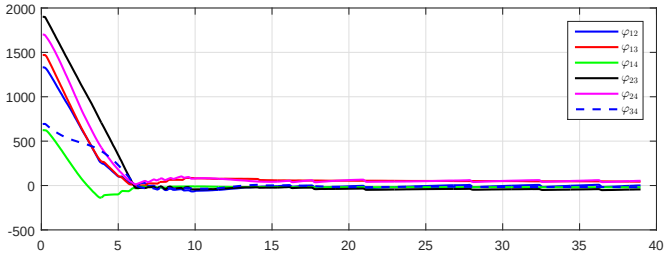


(d) Entradas de control

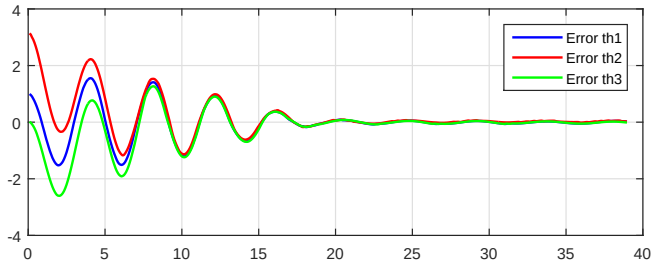
Figura 4.4: Experimento 1 de control de avance en formacion con $n = 4$



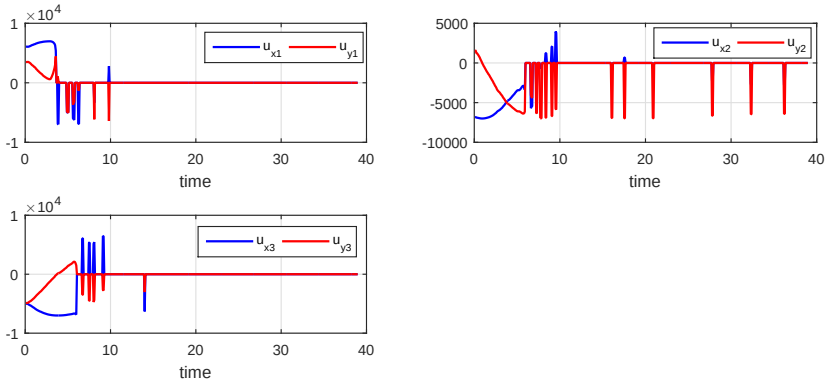
(a) Trayectorias de los robots



(b) Errores de formación

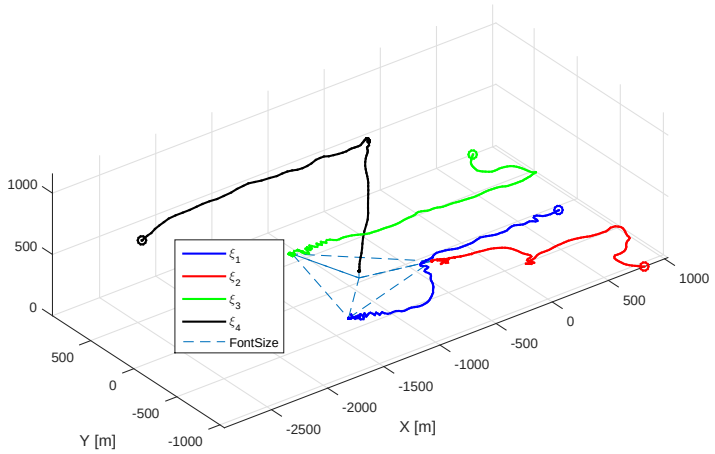


(c) Error angular

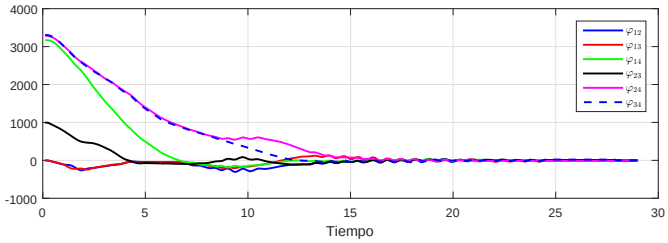


(d) Entradas de control

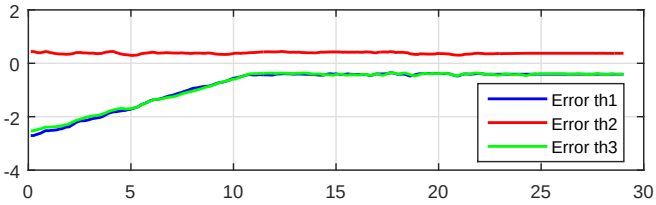
Figura 4.5: Experimento 2 de control de formacion con convergencia a un punto y $n = 4$.



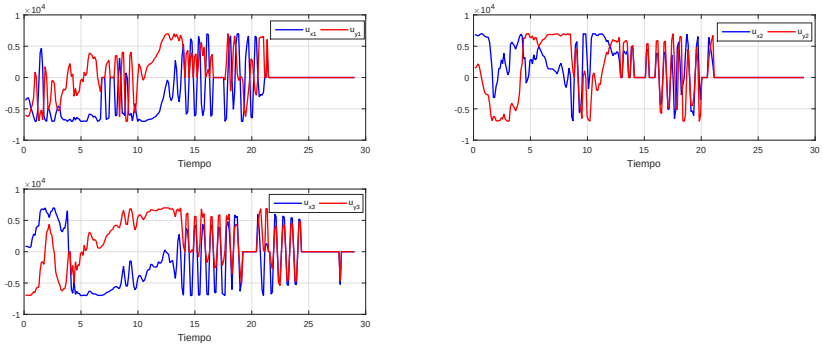
(a) Trayectorias de los robots



(b) Errores de formación



(c) Error angular



(d) Entradas de control

Figura 4.6: Experimento 3 de estrategia de unión y separación con $n = 4$.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Las conclusiones de esta tesis pueden resumirse en los siguiente puntos:

- La tecnología se ha convertido en una piedra angular del crecimiento y desarrollo humano en estos últimos siglos. La robótica ha mejorado las condiciones de vida y trabajo en las plantas de fabricación, así como también ha incrementado la calidad de los productos y ha disminuido los costos de producción. Actualmente se han enfocado los esfuerzos en el desarrollo de robots a robots móviles, ya que estos presentan mayores ventajas que los fijos para muchas aplicaciones industriales y de servicios. Así como la robótica móvil presenta grandes ventajas, también presenta grandes desafíos tanto tecnológicos como teóricos. Aunado a estos problemas se puede ver que cualquier sistema móvil se encuentra limitado actualmente a un área de trabajo en la cual puede desempeñar la tarea asignada.

- Esta tesis propone un acercamiento que permite ampliar el área de trabajo de los robots móviles, ya que permite la combinación de diversas plataformas móviles las cuales puedan ser utilizadas en conjunto para desempeñar alguna tarea en un ambiente multientorno. Asimismo se propone una estrategia de coordinación de movimiento la cual no depende intrínsecamente de sistemas de posicionamiento absoluto, lo que permite una mayor autonomía de los sistemas robóticos, ya que se puede realizar el sensado a bordo para así lograr un comportamiento completamente autónomo como es deseado.

- Al generar este tipo de comportamientos de deben tomar en cuenta ciertos aspectos de diseño para la ley de control. El primero es que pueda ser útil para diferentes plataformas robóticas, es decir, cumplir con un criterio de heterogeneidad. El segundo es que contenga términos relativos a la evasión de colisiones. Y el tercero sea modificable para cumplir con las estrategias de funcionamiento grupal o modular según sea el caso.

- La estrategia que se muestra en el Capítulo 3 cumple con estos requerimientos, como se muestra en la prueba formal presentada en la sección 3,2. Esta ley se puede adaptar con algunos cambios mínimos para cumplir con las estrategias que requiere el sistema robótico móvil multientorno modular para desplazarse en cualquier ambiente y cumplir con las tareas que se le asignen. Por lo tanto, se puede concluir que las metas y objetivos planteados en el capítulo 1 fueron alcanzados. Aun así, se considera que esta investigación generó proyectos relacionados que se desean investigar en un trabajo futuro.

- La implementación de las estrategias de control multientorno llevaron al diseño y construcción de una plataforma móvil de tipo omnidireccional. El objetivo fue ejemplificar el desempeño de las estrategias de movimiento multientorno. La plataforma fue construida utilizando distintos materiales y con un núcleo de control propio. Tanto la estructura como el controlador pueden ser utilizados para desarrollar otror robots omnidireccionales, teniendo como ventaja la homogeneidad de sus sistemas comunicación.

- Los experimentos mostrados satisfacen los requerimientos de control de forma aceptable. Al ser una evaluación preliminar del concepto multientorno, fue suficiente implementarla en espacio cerrado con la ayuda de un sistema de captación de movimiento.

5.2. Trabajo futuro

Para el trabajo futuro se propone abordar los siguientes puntos:

- Realizar algunas modificaciones al prototipo actual de robots, para que estos puedan ser sumergidon en agua y de esta forma se puedan extender las estrategias de control desarrolladas en esta tesis para robots acuáticos omnidireccionales. Así se incrementarán las capacidades multientorno del SRM.

- Evaluar experimentalmente la estabilización de aeronaves multi-rotor de forma autónoma, para que sea posible implementar las leyes de unión y separación que se presentan en la tesis de forma autónoma.

- Esta tesis se ha limita a combinar un solo robots aéreo modelado como un sistema de segundo orden, con robots de primer orden. Se ampliará la ley de control propuesta para incluir a más robots de este tipo, de forma que se ataque el caso general de formar varios robots terrestres con aéreos. Fenómenos de colaboración aéreo y terrestre en su totalidad podrán ser abordados, entre otros fenómenos estudiados en la literatura multi-robot.

- Se desea mudar el SRMM modular del espacio cerrado con el sistema de captación de movimiento actual a obtener la medición de posición y orientación con sensores a bordo. Asimismo, se deberán adecuar algunos parámetros de la ley de control para considerar contingencias de mediciones imprecisas de los sensores, como lo es la incompatibilidad de medición de distancias (distance

mismatches, por su término en inglés), errores de medición, filtrado de señales, mapeo del espacio de trabajo en tiempo. Todo esto generalmente muda este problema de instrumentación al área conocida como SLAM (Simultaneous Localization and Mapping). Esta temática necesita ser abordada ya que beneficia a esta tesis y es de vital importancia para los proyectos desarrollados en el laboratorio de análisis de movimiento.

Apéndice A

Diagramas de ensamble estructural de robots

A.1. Robot omnidireccional terrestre

A continuación se presentan los diagramas de maquinado del ensamble del robot terrestre de tres ruedas omnidireccionales. Primero se muestra un renderizado del robot. Después se muestra el ensamble general con el número de subpartes y posteriormente se muestran los diagramas de estas subpartes.

Figura A.1: Renderizado en CAD de robot terrestre

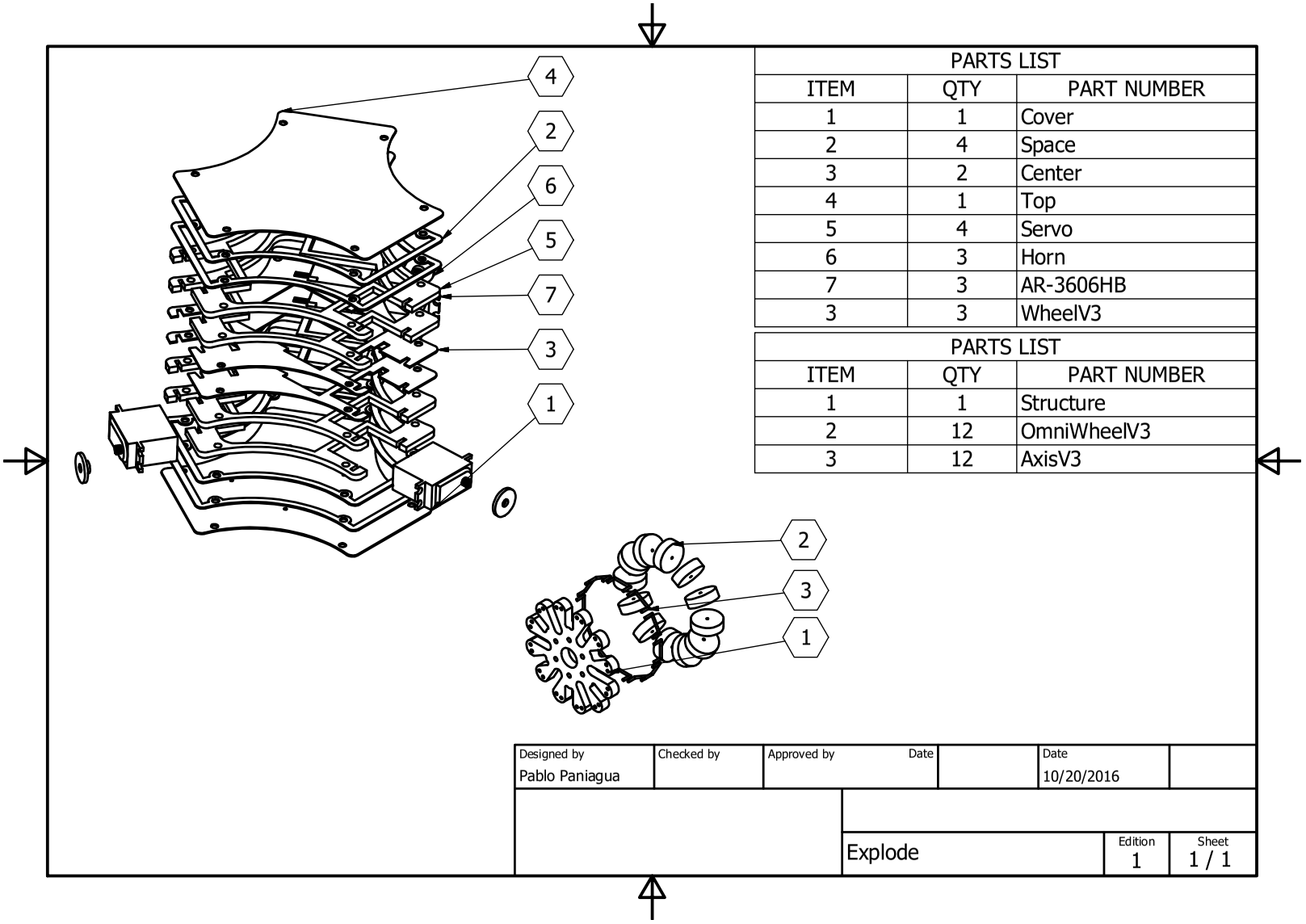


Figura A.2: Diagrama esquemático general con subpartes de ensamble

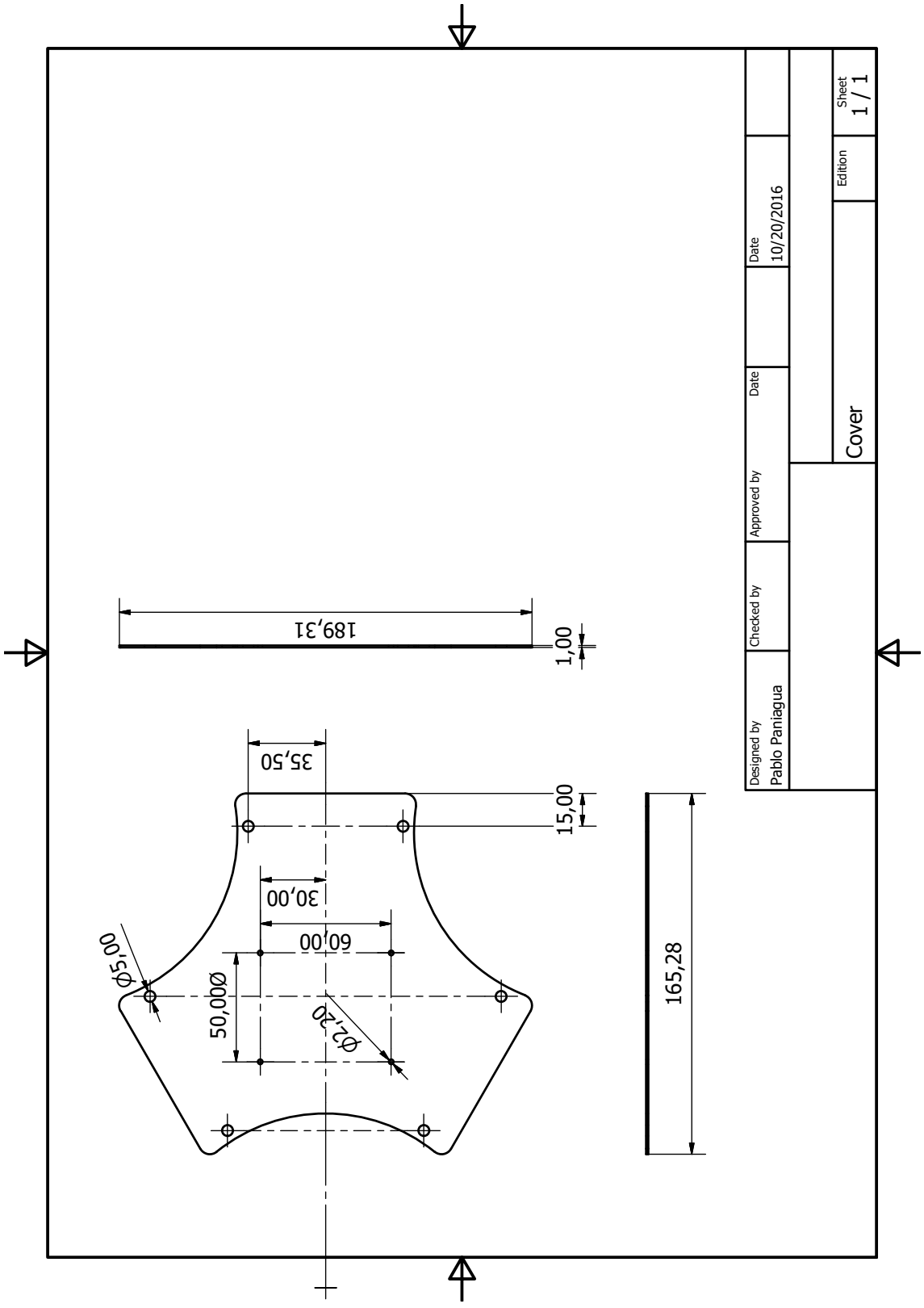


Figura A.3: Diagrama esquemático de subparte Cover

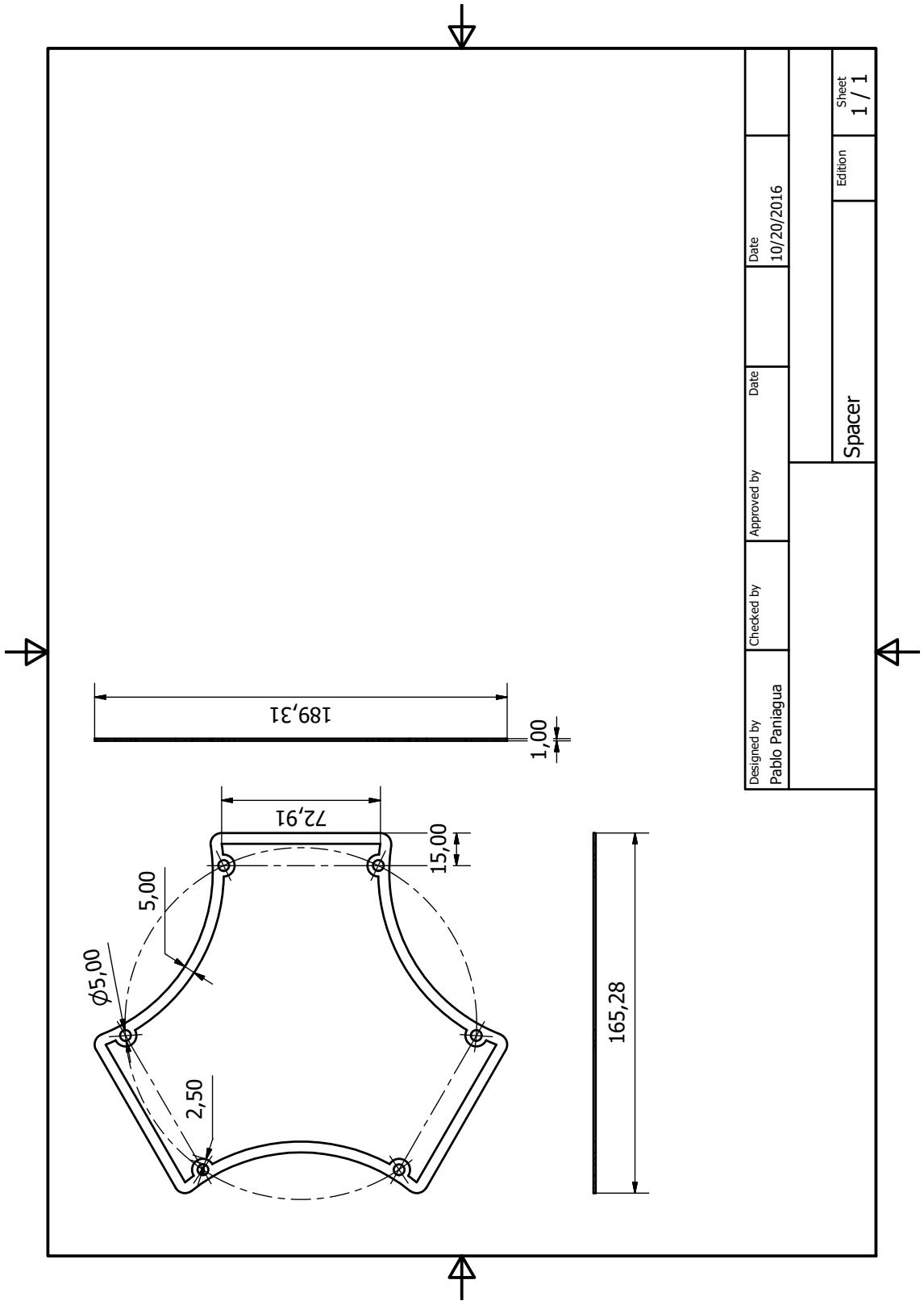


Figura A.4: Diagrama esquemático con subparte Space

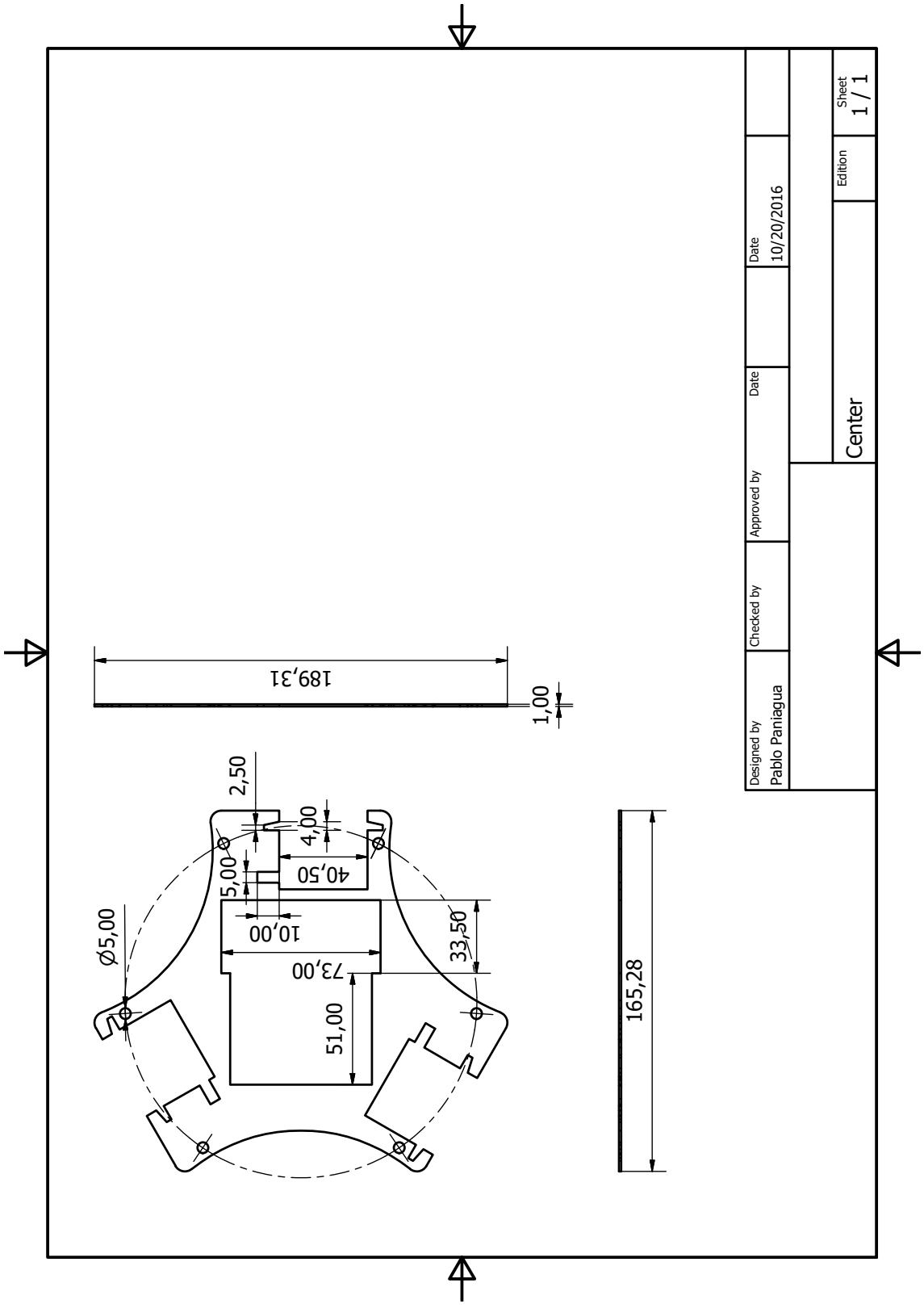


Figura A.5: Diagrama esquemático con subparte Center

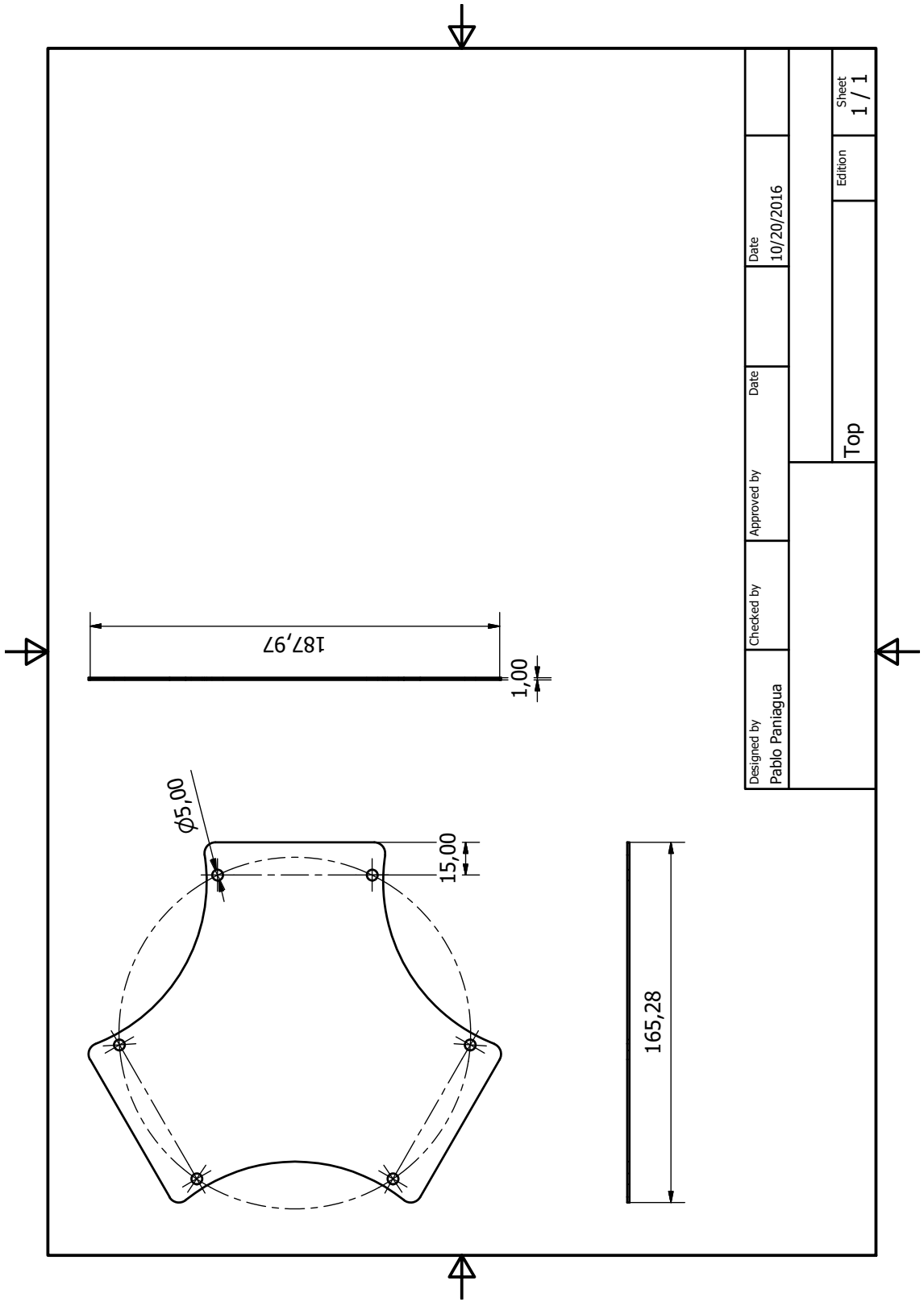


Figura A.6: Diagrama esquemático con subparte Top

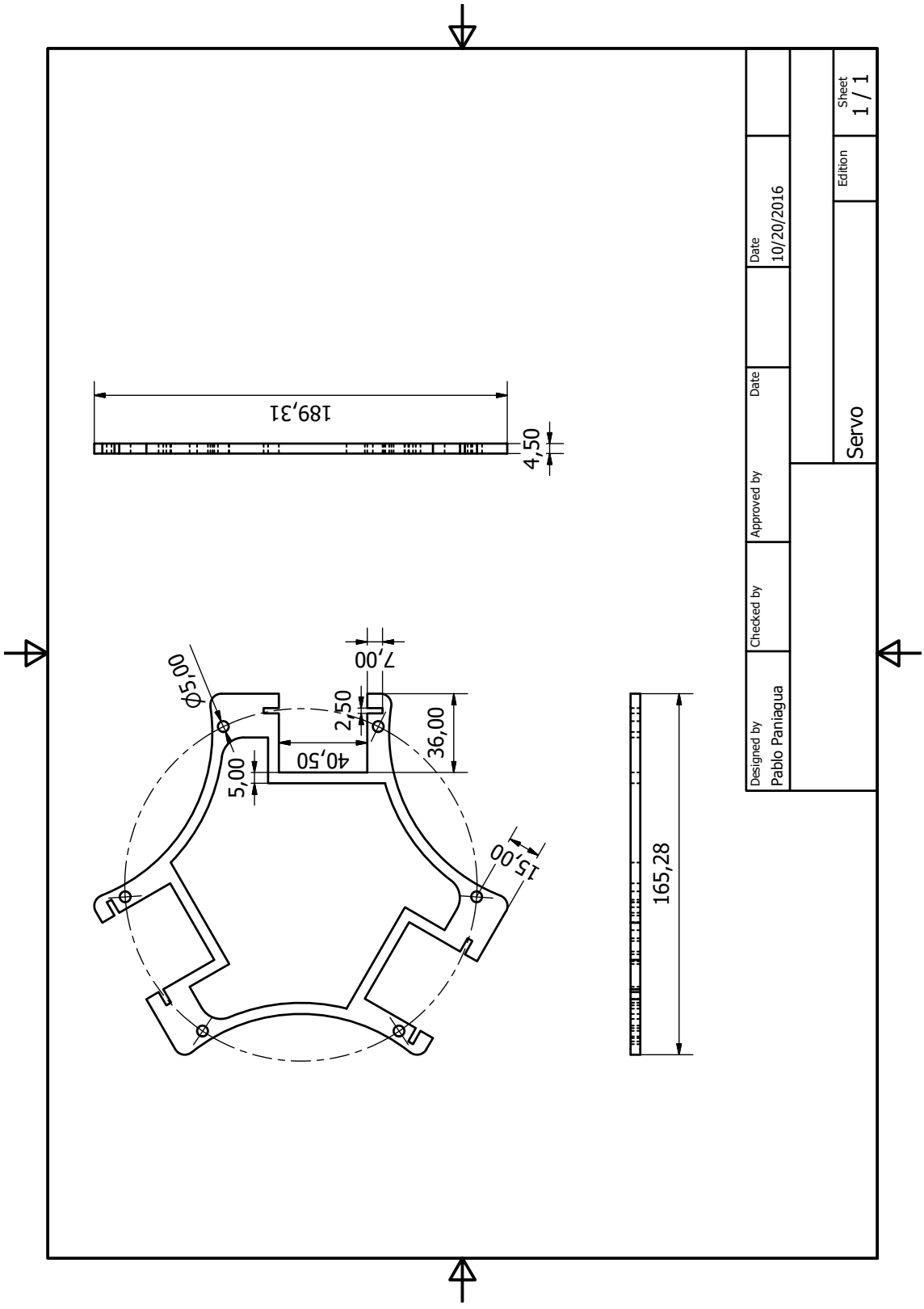


Figura A.7: Diagrama esquemático con subparte Servo

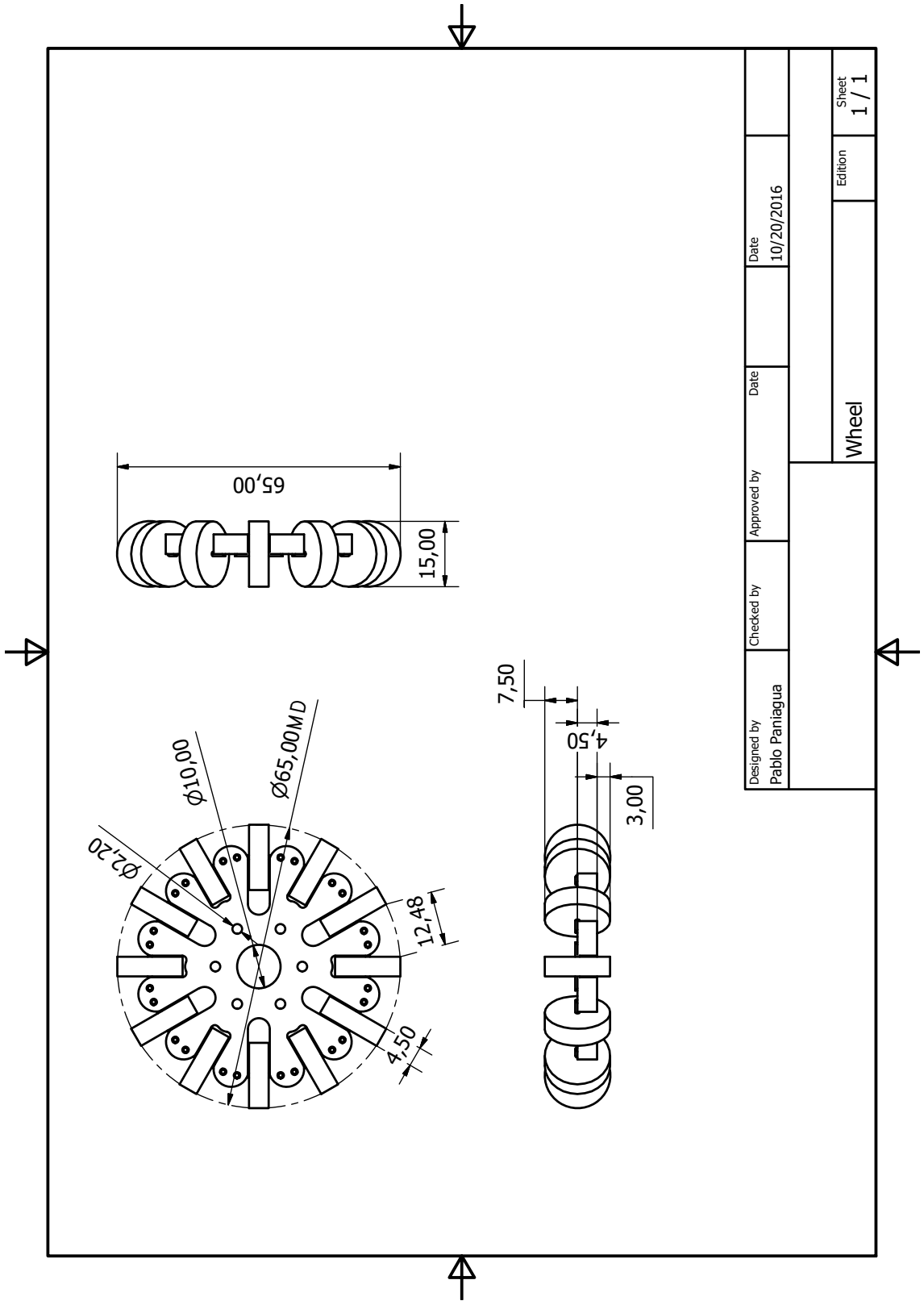


Figura A.8: Diagrama esquemático con subparte Wheel

Apéndice B

Diagrama y códigos del núcleo de control

B.1. Codigo del robot omnidireccional terrestre

```
#include "mbed.h"

DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);
DigitalOut led4(LED4);
Serial serialBT(p28,p27);
PwmOut motor1(p21),motor2(p22),motor3(p23);
Ticker safety;
bool FS=false;//FailSafe

void Joystick(char incom);

void FailSafe(){
        FS=true;
        led4 = 1;
        motor1.pulsewidth_us(1500);
        motor2.pulsewidth_us(1500);
        motor3.pulsewidth_us(1500);
}//FailSafe

void SerInterrupt(){
        led1 = !led1;
        char incom=serialBT.getc();
        if(!FS)
                Joystick(incom);
        else
                if(incom=='P'){
                        FS=false;
                        led4=0;
                }//if
}//SerInterrupt

int main(){
        //safety.attach(&FailSafe,0.15);
        serialBT.baud(9600);
        serialBT.attach(&SerInterrupt,Serial::RxIrq);
        motor1.period(0.02);
        motor2.period(0.02);
        motor3.period(0.02);
        motor1.pulsewidth_us(1500);
        motor2.pulsewidth_us(1500);
        motor3.pulsewidth_us(1500);
        led4 = 0;
        while(1){}
}//main

void Joystick(char incom){
        if((incom=='1')||(incom=='2')||(incom=='3')){
                short val=((short)serialBT.getc())|
                (((short)serialBT.getc())<<8);
                switch(incom){
                case '1':
                        motor1.pulsewidth_us(1500+val);
                        break;
                case '2':
                        motor2.pulsewidth_us(1500+val);
                        break;
                case '3':
```

```
                        motor3.pulsewidth_us(1500+val);
                        break;
                }//switch
        }else{
                while(serialBT.readable()){serialBT.getc();}
                FailSafe();
        }//else
        return;
}//Joystick
```

B.2. Biblioteca desarrollada para servos seriales

```
//Instructions for Dynamixel
#define iPing 0x01                      //No execution, is used when the controller is ready to
     receive Status Packet.  -->No. of parameters = 0
#define iReadData 0x02                  //Reads data from Dynamixel.
                                                -->No. of parameters = 2
#define iWriteData 0x03                 //Writes data to Dynamixel.
                                                  -->No. of parameters = 2 or more
#define iRegWrite 0x04                  //Writes data to dynamixel, but it waits for the action
     comand to arrive.       -->No. of parameters = 2 or more
#define iAction 0x05                    //Initiates motions registered with RegWrite.
                                          -->No. of parameters = 0
#define iReset 0x06                     //Restores the state of Dynamixel to the factory default
     settings.               -->No. of parameters = 0
#define iSyncWrite 0x83                 //Its used to control several Dynamixels simultaneously.
                                -->No. of parameters = 4 or more
//Control Table Eeprom
#define cID 0x03                        //ID of Dynamixel.
#define cBaudRate 0x04                  //Baud Rate of Dynamixel.
#define cReturnDelayTime 0x05           //Return Delay Time.
#define cCWAngleLimitL 0x06             //Lowest byte of clockwise Angle Limit.
#define cCWAngleLimitH 0x07             //Highest byte of clockwise Angle Limit.
#define cCCWAngleLimitL 0x08            //Lowest byte of counterclockwise Angle Limit.
#define cCCWAngleLimitH 0x09            //Highest byte of counterclockwise Angle Limit.
#define cTemperatureLimitH 0x0B         //Internal Limit Temperature.
#define cLowVoltageLimit 0x0C           //Lowest Limit Voltage.
#define cHighVoltageLimit 0x0D          //Highest Limit Voltage.
#define cMaxTorqueL 0x0E                //Lowest byte of Max. Torque.
#define cMaxTorqueH 0x0F                //Highest byte of Max. Torque.
#define cStatusReturnLevel 0x10         //Status Return Level.
#define cAlarmLED 0x11                  //LED for Alarm.
#define cAlarmShutDown 0x12             //Shutdown for Alarm.
//Control Table RAM
#define cTorqueEnable 0x18              //Torque On/Off.
#define cLED 0x19                       //LED On/Off.
#define cCWCompilanceMargin 0x1A        //CW Compliance margin.
#define cCCWCompilanceMargin 0x1B       //CCW Compliance margin.
#define cCWCompilanceSlope 0x1C         //CW Compliance slope.
#define cCCWCompilanceSlope 0x1D        //CCW Compliance slope.
#define cGoalPositionL 0x1E             //Lowest byte of Goal Position.
#define cGoalPositionH 0x1F             //Highest byte of Goal Position.
#define cMovingSpeedL 0x20              //Lowest byte of Moving Speed (Moving Velocity).
#define cMovingSpeedH 0x21              //Highest byte of Moving Speed (Moving Velocity).
#define cTorqueLimitL 0x22              //Lowest byte of Torque Limit (Goal Torque).
#define cTorqueLimitH 0x23              //Highest byte of Torque Limit (Goal Torque).
#define cLock 0x2F                      //Locking EEPROM.
#define cPunchL 0x30                    //Lowest byte of Punch.
#define cPunchH 0x31                    //Highest byte of Punch.

#include "mbed.h"

//Class that defines one dinamixel AX-12W
class Dynamixel{
        private:
                int position;               //Absolute position of the servo. (It goes from 0 to
                     1023)(The mode hast to be set in False).
                int velocity;               //Angular velocity of the servo. (It goes from 0 to
                     2047)(The mode hast to be set in True).
                bool led;                   //Led state on board of the dynamixel.
                bool serialDirection;       //Serial Tx/Rx selection bit (True=send  False=
                     receive).
                bool mode;                  //Servo working mode (True=Continous  False=Absolute
                     ).
                char ID;                    //Dinamixel ID (254 || 0xFE it's a podcast to all
                     dynamixels conected).
                Serial serialPort;          //Serial port
                DigitalOut directionPin;    //Direction Pin
        public:
                Dynamixel(void);            //ID=0xFE(Podcast); In Absolute mode on 0deg; LED-->
                     off; Tx pin-->p9; Rx pin-->p10; directionPin pin-->p11 on low
                Dynamixel(unsigned int);    //In Absolute mode on 0deg; LED-->off; Tx pin-->p9;
                     Rx pin-->p10; directionPin pin-->p11 on low
                Dynamixel(unsigned int,PinName,PinName,PinName,bool);
                void SetID(char newID){ID=newID;};
                void SetLED(bool);
                void SetMode(bool);
                void SetPosition(unsigned long);
                void SetVelocity(unsigned long);
                //To be tested.....................
                void SetDirection(bool);
                void move(void);
};//Dynamixel

#include "DynamixelAX12.h"
```

```
#include "mbed.h"

Dynamixel::Dynamixel(void):serialPort(p13,p14),directionPin(p12,0){
        serialPort.baud(1000000);
        position=0;
        velocity=0;
        led=false;
        serialDirection=false;
        mode=false;
        ID=0xFE;
}//Builder0

Dynamixel::Dynamixel(unsigned int NewID):serialPort(p13,p14),directionPin(p12,0){
        serialPort.baud(1000000);
        position=0;
        velocity=0;
        led=false;
        serialDirection=false;
        mode=false;
        ID=(unsigned char)(NewID & 0XFF);
}//Builder1

Dynamixel::Dynamixel(unsigned int NewID,PinName Newtx,PinName Newrx,PinName newdirectionPin,bool
     NewMode):serialPort(Newtx,Newrx),directionPin(newdirectionPin,1){
        serialPort.baud(1000000);
        position=0;
        velocity=0;
        led=false;
        serialDirection=false;
        ID=NewID;
        mode=NewMode;
        ID=(unsigned char)(NewID & 0XFF);
}//Builder2

void Dynamixel::SetLED(bool ledState){
        char TxBuf[8];
        TxBuf[0]=0xFF;              //Start of incoming packet
        TxBuf[1]=0xFF;              //Start of incoming packet
        TxBuf[2]=ID;                //ID
        TxBuf[3]=0x04;              //Length = number of parameters + 2
        TxBuf[4]=iWriteData;        //Instruction
        TxBuf[5]=cLED;              //Parameter 1
        if(ledState)
                TxBuf[6]=0x01;          //Parameter 2
        else
                TxBuf[6]=0x00;          //Parameter 2
        TxBuf[7]= ~(ID+TxBuf[3]+iWriteData+cLED+TxBuf[6]);
        directionPin.write(1);
        for(int i=0;i<8;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//SetLED

void Dynamixel::SetMode(bool Mode){
        char TxBuf[11];
        TxBuf[0]=0xFF;              //Start of incoming packet0
        TxBuf[1]=0xFF;              //Start of incoming packet1
        TxBuf[2]=ID;                //ID
        TxBuf[3]=0x07;              //Length = number of parameters + 2
        TxBuf[4]=iWriteData;        //Instruction
        TxBuf[5]=cCWAngleLimitL;    //Parameter 1
        TxBuf[6]=0x00;              //Parameter 2
        TxBuf[7]=0x00;              //Parameter 3
        if(Mode){
                TxBuf[8]=0x00;          //Parameter 4
                TxBuf[9]=0x00;          //Parameter 5
        }else{
                TxBuf[8]=0xFF;          //Parameter 4
                TxBuf[9]=0x03;          //Parameter 5
        }//else
        TxBuf[10]= ~(ID+TxBuf[3]+iWriteData+cCWAngleLimitL+TxBuf[6]+TxBuf[7]+TxBuf[8]+TxBuf[9]);
        directionPin.write(1);
        for(int i=0;i<11;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//SetMode

void Dynamixel::SetPosition(unsigned long Newposition){
        char TxBuf[9];
        TxBuf[0]=0xFF;              //Start of incoming packet0
        TxBuf[1]=0xFF;              //Start of incoming packet1
        TxBuf[2]=ID;                //ID
        TxBuf[3]=0x05;              //Length = number of parameters + 2
        TxBuf[4]=iWriteData;        //Instruction
        TxBuf[5]=cGoalPositionL;    //Parameter 1
        TxBuf[6]=(unsigned char)(Newposition & 0XFF);
        TxBuf[7]=(unsigned char)((Newposition >> 8) & 0XFF);
        TxBuf[8]= ~(ID+TxBuf[3]+iWriteData+cGoalPositionL+TxBuf[6]+TxBuf[7]);
        directionPin.write(1);
        for(int i=0;i<9;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//SetPosition
```

```
void Dynamixel::SetVelocity(unsigned long Newvelocity){
        char TxBuf[9];
        TxBuf[0]=0xFF;                  //Start of incoming packet0
        TxBuf[1]=0xFF;                  //Start of incoming packet1
        TxBuf[2]=ID;                    //ID
        TxBuf[3]=0x05;                  //Length = number of parameters + 2
        TxBuf[4]=iWriteData;             //Instruction
        TxBuf[5]=cMovingSpeedL;         //Parameter 1
        TxBuf[6]=(unsigned char)(Newvelocity & 0XFF);
        TxBuf[7]=(unsigned char)((Newvelocity >> 8) & 0XFF);
        TxBuf[8]= ~(ID+TxBuf[3]+iWriteData+TxBuf[5]+TxBuf[6]+TxBuf[7]);
        directionPin.write(1);
        for(int i=0;i<9;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//SetVelocity

void Dynamixel::SetDirection(bool Mode){
        char TxBuf[9];
        TxBuf[0]=0xFF;                  //Start of incoming packet0
        TxBuf[1]=0xFF;                  //Start of incoming packet1
        TxBuf[2]=ID;                    //ID
        TxBuf[3]=0x05;                  //Length = number of parameters + 2
        TxBuf[4]=iWriteData;            //Instruction
        TxBuf[5]=cMovingSpeedL;         //Parameter 1
        TxBuf[6]=0xFF;                  //Parameter 2
        if(Mode)
                TxBuf[7]=0x03;              //Parameter 3
        else
                TxBuf[7]=0x07;              //Parameter 3
        TxBuf[8]= ~(ID+TxBuf[3]+iWriteData+cCWAngleLimitL+TxBuf[6]+TxBuf[7]);
        directionPin.write(1);
        for(int i=0;i<9;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//SetDirection

void Dynamixel::move(){
        char TxBuf[16];
        TxBuf[0]=0xFF;                  //Start of incoming packet0
        TxBuf[1]=0xFF;                  //Start of incoming packet1
        TxBuf[2]=0xFE;                  //ID
        TxBuf[3]=0x02;                  //Length = number of parameters + 2
        TxBuf[4]=iAction;               //Instruction
        TxBuf[5]=0xFA;
        directionPin.write(1);
        for(int i=0;i<6;i++){
                serialPort.putc(TxBuf[i]);
        }//for
        wait_ms(10);
        directionPin.write(0);
}//move
```

B.3. Diagrama esquemático y PCB del controlador Huex

En esta sección se presenta una imagen del diagrama esquemático del controlador HUEX, así un renderizado de la placa electrónica diseñada.

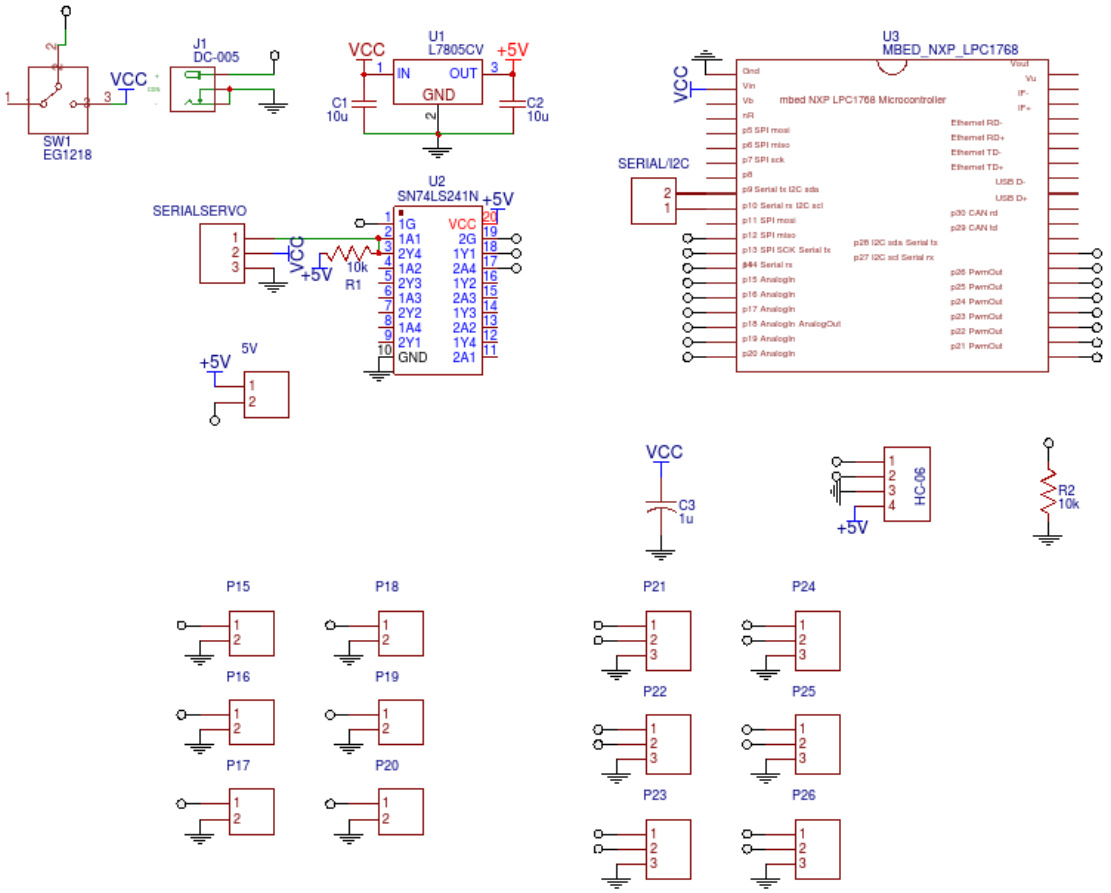


Figura B.1: Diagrama esquemáticos del controlador HUEX

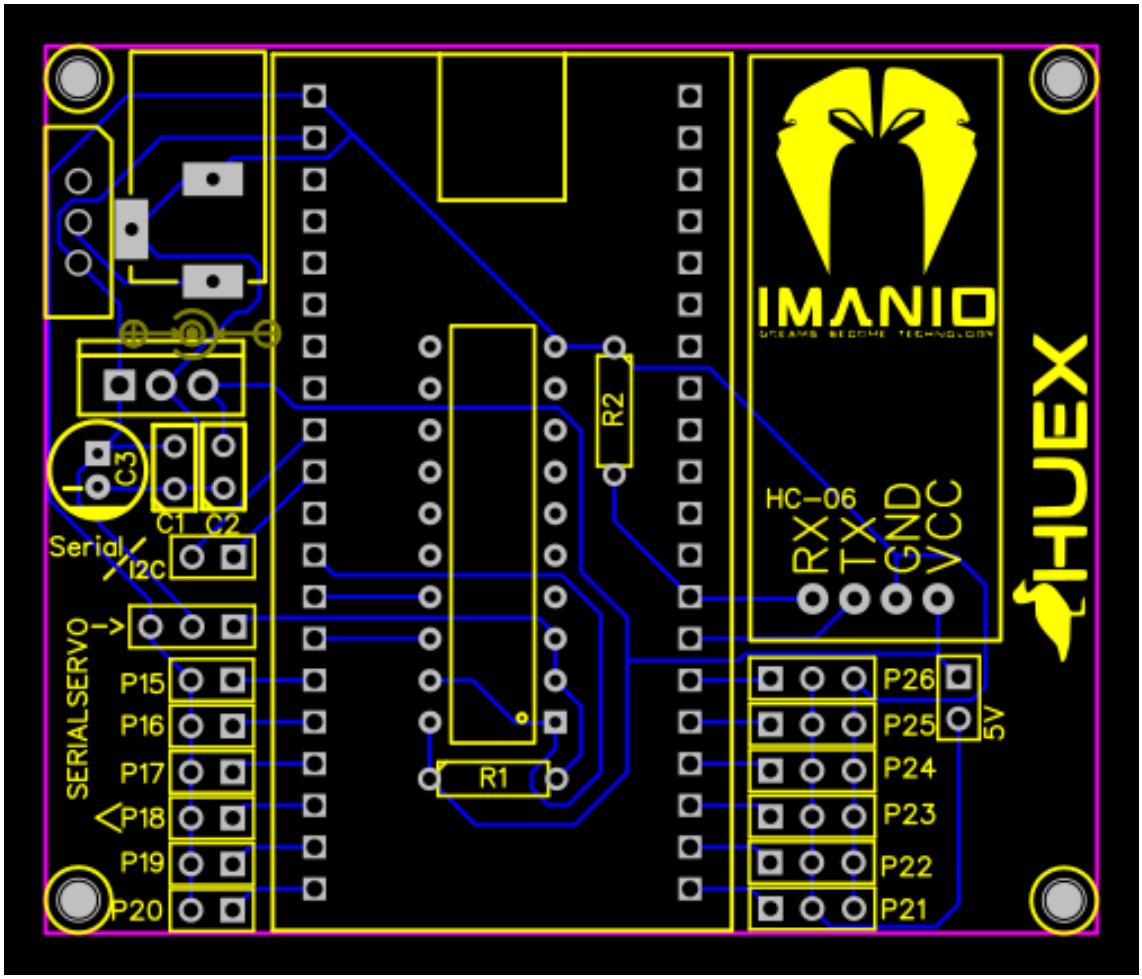


Figura B.2: Rendederizado del PCB del controlador HUEX

Apéndice C

Archivos de Matlab de simulaciones y experimentos

```
%By Pablo Paniagua
%22/06/16

classdef PAXBOT < handle
properties
type=2;             %2 as in drone
BTport;             %Bluetooth serial port
pos;                %Current position of the robot
name;               %Robot object name as it is on tracker
j=vrjoystick(1);    %Joystick
r=30;               %Radio de las ruedas
L=95;               %Longitud entre llantas
SPx=0;              %Home x
SPy=0;              %Home y
SPw=0;              %Home w

rcirc=[500;500];    %Radio de la circunferencia
C=[0;0];            %Centro del circulo
wc = (2*pi)/8;      %Frecuencia
kt=150;             %Ganancia de trayectoria
end
methods
function obj = PAXBOT(name)
obj.name=name;
obj.pos=COORDINATES();
end
function createBT(obj)
obj.BTport=Bluetooth(obj.name,1);
end
function connect(obj)
fopen(obj.BTport);
end
function disconnect(obj)
fclose(obj.BTport);
end
function lock(obj)
fwrite(obj.BTport,'P','char');
pause(1);
end
function update(obj,t)
obj.pos=t.get(obj.name);
end
function setHome(obj,x,y,w)
obj.SPx=x;
obj.SPy=y;
obj.SPw=w;
end
function rc(obj)
obj.connect;
obj.lock;
while(~button(obj.j,8))
vx=axis(obj.j,1)*20000;
vy=-axis(obj.j,2)*20000;
w=-axis(obj.j,3)*100;
move(obj,vx,vy,w);
pause(.1);
if(button(obj.j,2))
obj.lock();
end
end
obj.move(0,0,-10);
obj.lock;
obj.disconnect;
end
%          function [X,Y,Z,PHI,TH,PSI]=home(obj,t)
```

```
function home(obj,t)
obj.connect;
obj.lock;
error=100;
eps=.0001;
k=1;
i=0;
while(abs(norm(error))>5)
obj.update(t);
i=i+1;
%                   X(i)=obj.pos.x;
%                   Y(i)=obj.pos.y;
%                   Z(i)=obj.pos.z;
%                   PHI(i)=obj.pos.phi;
%                   TH(i)=obj.pos.th;
%                   PSI(i)=obj.pos.psi;
A=[cos(obj.pos.psi) -sin(obj.pos.psi) 0;sin(obj.pos.psi) cos(obj.pos.psi) 0;0 0 1];
error=[-obj.SPx+obj.pos.x;-obj.SPy+obj.pos.y;-obj.SPw+obj.pos.psi];
F=[-k*error(1);-k*error(2);-0.5*k*error(3)];
if(abs(norm([error(1); error(2)]))<=10)
N=0;
else
N=6000;
end
if(abs(error(3))<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(F(1)^2+F(2)^2+eps)*[F(1); F(2)];
FNormang=Nw/sqrt(F(3)^2+eps)*F(3);
FNorm=[FNormpos; FNormang];
U=A\FNorm;
obj.move(U(1),U(2),U(3));
pause(.1);
if(button(obj.j,8))
break;
end
end
obj.move(0,0,0);
obj.lock;
obj.disconnect;
%               plot3(X,Y,Z);
%               axis([-1500 1500 -3500 3500 0 2000]);
%               box on;
end
%          function [X,Y,Z,PHI,TH,PSI]=patrol(obj,t)
function patrol(obj,t)
obj.connect;
obj.lock;
i=0;
tic
while(~button(obj.j,8))
obj.update(t);
time=toc;
i=i+1;
%                   X(i)=obj.pos.x;
%                   Y(i)=obj.pos.y;
%                   Z(i)=obj.pos.z;
%                   PHI(i)=obj.pos.phi;
%                   TH(i)=obj.pos.th;
%                   PSI(i)=obj.pos.psi;
rc1 = obj.rcirc(1);
rc2 = obj.rcirc(2);
%rc3 = pi/2;
cx = obj.C(1);
cy = obj.C(2);
mx = cx + rc1*cos(obj.wc*time);
my = cy + rc2*sin(obj.wc*time);
mxp = -rc1*obj.wc*sin(obj.wc*time);
myp = rc2*obj.wc*cos(obj.wc*time);
mz = atan2(myp,mxp);
%     mz = SPw;
mxpp = -rc1*obj.wc*obj.wc*cos(obj.wc*time);
mypp = -rc2*obj.wc*obj.wc*sin(obj.wc*time);
mzp = (myp*mxpp-mypp*mxp)/(mxp^2+myp^2);
%     mzp = 0;
A=[cos(obj.pos.psi) -sin(obj.pos.psi) 0;sin(obj.pos.psi) cos(obj.pos.psi) 0;0 0 1];
error=[-mx+obj.pos.x;-my+obj.pos.y;-mz+obj.pos.psi];
F=[-obj.kt*error(1);-obj.kt*error(2);-0.2*obj.kt*error(3)] +[mxp;myp;mzp];
U=A\F;
obj.move(U(1),U(2),U(3));
pause(.1);
end
obj.move(0,0,-10);
obj.lock;
obj.disconnect;
%               plot3(X,Y,Z);
%               axis([-1500 1500 -3500 3500 0 2000]);
%               grid on;
%               box on;
end
function move(obj,vx,vy,w)
w=-w;
temp=-vx;
vx=-vy;
vy=temp;
J=(1/obj.r)*[(-sqrt(3)/2) 0.5 obj.L;0 -1 obj.L;(sqrt(3)/2) 0.5 obj.L]*[vx;vy;w];
w1=J(1);
```

```
w2=J(2);
w3=J(3);
if(w1>500)
w1=500;
end
if(w2>500)
w2=500;
end
if(w3>500)
w3=500;
end
if(w1<-500)
w1=-500;
end
if(w2<-500)
w2=-500;
end
if(w3<-500)
w3=-500;
end
fwrite(obj.BTport,'1','char');
fwrite(obj.BTport,w1,'int16');
fwrite(obj.BTport,'2','char');
fwrite(obj.BTport,w2,'int16');
fwrite(obj.BTport,'3','char');
fwrite(obj.BTport,w3,'int16');
end
end
end


%***************************************
%                By Pablo Paniagua      *
%                Last update 06/09/2016 *
%***************************************
%                POSITION               *
%***************************************

classdef COORDINATES < handle
properties
x;
y;
z;
phi;
th;
psi;
end
methods
function obj=COORDINATES()
obj.x=0;
obj.y=0;
obj.z=0;
obj.phi=0;
obj.th=0;
obj.psi=0;
end
function set(obj,x,y,z,phi,th,psi)
obj.x=x;
obj.y=y;
obj.z=z;
obj.phi=phi;
obj.th=th;
obj.psi=psi;
end
function setX(obj,x)
obj.x=x;
end
function setY(obj,y)
obj.y=y;
end
function setZ(obj,z)
obj.z=z;
end
function setPHI(obj,phi)
obj.phi=phi;
end
function setTH(obj,th)
obj.th=th;
end
function setPSI(obj,psi)
obj.psi=psi;
end
end
end



%*********************************
%          By Pablo Paniagua      *
%          Last update 05/09/2016 *
%*********************************
%          TRACKER                *
%*********************************
%   This program gives you acces to tracker to get the position of any    *
%   object saved.                                                         *
%                                                                         *
%   This class has only three methods:                                    *
%       1. on  -> Lets you initialize the vicon client                    *
%       2. off -> Terminates the vicon client communication               *
```

```
%       3. get -> Its defined as:                                          *
%                       position = name.get('ObjectName')                  *
%                   This method returns the position of any object saved   *
%                   on tracker.                                            *
%                   It's input is the name of the desired object as it is  *
%                   on tracker.                                            *
%                   It returns a COORDINATES class variable                *
%************************
%  ATTENTION!  Remember to always run the "on" method before trying to     *
%              retrieve any position with "get"; allso remember to close  *
%              the class with the mothod "off" at the end of your         *
%              program, avoiding these instructions may jam matlab.       *
%************************

classdef TRACKER
properties
MyClient
end
methods
function obj = TRACKER()
loadlibrary( 'ViconDataStreamSDK_MATLAB.dll' , 'ViconDataStreamSDK_MATLAB.h' )
obj.MyClient = calllib('ViconDataStreamSDK_MATLAB','ConstructInstance_v2' );
end
function on(obj)
momentum=struct('Connected',calllib('ViconDataStreamSDK_MATLAB','IsConnected_v2',obj.MyClient));
while(~momentum.Connected)
pResult = libpointer( 'uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','Connect_v2',obj.MyClient,'localhost:801',pResult );
momentum=struct('Connected',calllib('ViconDataStreamSDK_MATLAB','IsConnected_v2',obj.MyClient));
end
pResult = libpointer( 'uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','EnableSegmentData_v2',obj.MyClient,pResult);
pResult = libpointer( 'uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','EnableDeviceData_v2',obj.MyClient,pResult);
pResult = libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','SetStreamMode_v2',obj.MyClient,0,pResult);
pResult = libpointer('uint32Ptr',0);
calllib( 'ViconDataStreamSDK_MATLAB','SetAxisMapping_v2',obj.MyClient,4,2,0,pResult);
pMajor = libpointer( 'uint32Ptr',0);
pMinor = libpointer( 'uint32Ptr',0);
pPoint = libpointer( 'uint32Ptr',0);
calllib( 'ViconDataStreamSDK_MATLAB', 'GetVersion_v2', obj.MyClient, pMajor, pMinor, pPoint );
Major = get( pMajor, 'Value' );
Minor = get( pMinor, 'Value' );
Point = get( pPoint, 'Value' );
Output_GetVersion = struct( 'Major', Major, 'Minor', Minor, 'Point', Point );
sprintf( 'Vicon DataStreamSDK Version: %d.%d.%d\n', Output_GetVersion.Major,Output_GetVersion.
    Minor,Output_GetVersion.Point);
end
function off(obj)
pResult = libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','Disconnect_v2',obj.MyClient,pResult);
unloadlibrary  ViconDataStreamSDK_MATLAB
end
function pos = get(obj,objectName)
pResult=libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','GetFrame_v2',obj.MyClient,pResult);
cResult=Result(get(pResult,'Value'));
momentum=struct('Result',cResult);
while(momentum.Result.Value ~= Result.Success)
fprintf( '/nVicon Frame Error' );
pResult=libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','GetFrame_v2',obj.MyClient,pResult);
cResult=Result(get(pResult,'Value'));
momentum=struct('Result',cResult);
end
pResult = libpointer( 'uint32Ptr',0);
cSegmentName = calllib( 'ViconDataStreamSDK_MATLAB', 'GetSubjectRootSegmentName_v2', obj.
    MyClient, objectName, pResult );
cResult = Result( get( pResult, 'Value' ) );
RootName=struct('Result',cResult,'SegmentName',cSegmentName);
pResult=libpointer('uint32Ptr',0);
pTranslation=libpointer('doublePtrPtr',zeros(1,3));
pOccluded=libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','GetSegmentGlobalTranslation_v2',obj.MyClient,objectName,
    RootName.SegmentName,pTranslation,pOccluded,pResult);
cResult      = Result( get( pResult, 'Value' ) );
cTranslation = get( pTranslation, 'Value' );
cTranslation = cTranslation';
cOccluded    = get( pOccluded, 'Value' );
Position=struct('Result',cResult,'Translation',cTranslation,'Occluded',cOccluded);
pResult=libpointer('uint32Ptr',0);
pRotation=libpointer('doublePtrPtr',zeros(1,3));
pOccluded=libpointer('uint32Ptr',0);
calllib('ViconDataStreamSDK_MATLAB','GetSegmentGlobalRotationHelical_v2',obj.MyClient,objectName
    ,RootName.SegmentName,pRotation,pOccluded,pResult);
cResult   = Result( get( pResult, 'Value' ) );
cRotation = get( pRotation, 'Value' );
cRotation = cRotation';
cOccluded = get( pOccluded, 'Value' );
Rotation = struct( 'Result', cResult, 'Rotation', cRotation, 'Occluded', cOccluded );
pos=COORDINATES();
pos.set(Position.Translation(1),Position.Translation(2),Position.Translation(3),Rotation.
    Rotation(1),Rotation.Rotation(2),Rotation.Rotation(3))
end
end
end
```

```
%*******************************
%         By Pablo Paniagua      *
%         Last update 07/09/2016 *
%*******************************
%         Heterogeneous          *
%*******************************
%
%             Pbots(1)=PAXBOT('PAXBOT1');
%             Pbots(2)=PAXBOT('PAXBOT2');
%             Pbots(3)=PAXBOT('PAXBOT3');
%             Pbots(1).createBT;
%             Pbots(2).createBT;
%             Pbots(3).createBT;
%****************************
function [DRONE,SRM1,SRM2,SRM3,ControlSRM1,ControlSRM2,ControlSRM3,ControlDRONE,Time,ErrorM]=
    Tesis(Pbots,track)
%%                                      INIT Joystick
j=vrjoystick(1);
%     %%                                      INIT Tracker
%     track=TRACKER();
%     track.on;
%%                                      INIT Robots
Pbots(1).connect;
Pbots(2).connect;
Pbots(3).connect;
%%                                      Return values
maxIteration=500;
DRONE=zeros(4,maxIteration);
SRM1=zeros(3,maxIteration);
SRM2=zeros(3,maxIteration);
SRM3=zeros(3,maxIteration);
ControlSRM1=zeros(3,maxIteration);
ControlSRM2=zeros(3,maxIteration);
ControlSRM3=zeros(3,maxIteration);
ControlDRONE=zeros(4,maxIteration);
Time=zeros(1,maxIteration);
ErrorM=zeros(2,maxIteration);
%%                                      Control Parameters
SP_D=1000;
SP_DD=500/cos(pi/6);
SP_z4=1000;
%     kpO=750;
%     kr=400;
%     rhod=400;                   %Ganancia derivativa
%     kTH=.2;
%     eps=0.0001;
kpO=750;
kr=400;
rhod=400;                   %Ganancia derivativa
kTH=.8;
eps=0.01;
%%                                      TRAYECTORY
wc=2*pi/32;                 %Frecuencia
rc=500;                     %Radio de la circunferencia
C=[0 0];                    %Centro del circulo
%%                                      DRONE/WAND DATA
%     WandPos=track.get('Wand');
Drone=COORDINATES;
Drone.x=C(1)+rc;
Drone.y=C(2);
Drone.z=0;
Drone.psi=0;
Drone_past=Drone;
Drone_dt=COORDINATES;
Drone_dt_past=COORDINATES;
u4_past=COORDINATES;
%%                                      TIMER
tic;
t_past=0;
try
iteration=0;
%%                                      Plot
%     figure(1);
%     axis equal;
%     axis([-1500 1500 -3500 3500 0 2000]);
%     grid on;
%     box on;
%%                                  Loop
while((~button(j,8))&&(iteration ~= maxIteration))
iteration=iteration+1;
disp(iteration);
pause(.05);
%%                              Get Vicon
Pbots(1).update(track);
Pbots(2).update(track);
Pbots(3).update(track);
%%                              Variables
Beta12=norm([Pbots(2).pos.x,Pbots(2).pos.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta13=norm([Pbots(3).pos.x,Pbots(3).pos.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta23=norm([Pbots(3).pos.x,Pbots(3).pos.y]-[Pbots(2).pos.x,Pbots(2).pos.y]);

Beta14=norm([Drone.x,Drone.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta24=norm([Drone.x,Drone.y]-[Pbots(2).pos.x,Pbots(2).pos.y]);
Beta34=norm([Drone.x,Drone.y]-[Pbots(3).pos.x,Pbots(3).pos.y]);


theta12=atan2(Pbots(2).pos.y-Pbots(1).pos.y,Pbots(2).pos.x-Pbots(1).pos.x);
theta13=atan2(Pbots(3).pos.y-Pbots(1).pos.y,Pbots(3).pos.x-Pbots(1).pos.x);
```

```
theta21=atan2(Pbots(1).pos.y-Pbots(2).pos.y,Pbots(1).pos.x-Pbots(2).pos.x);
theta23=atan2(Pbots(3).pos.y-Pbots(2).pos.y,Pbots(3).pos.x-Pbots(2).pos.x);
theta31=atan2(Pbots(1).pos.y-Pbots(3).pos.y,Pbots(1).pos.x-Pbots(3).pos.x);
theta32=atan2(Pbots(2).pos.y-Pbots(3).pos.y,Pbots(2).pos.x-Pbots(3).pos.x);

theta14=atan2(Drone.y-Pbots(1).pos.y,Drone.x-Pbots(1).pos.x);
%               theta41=atan2(Pbots(1).pos.y-Drone.y,Pbots(1).pos.x-Drone.x);
theta24=atan2(Drone.y-Pbots(2).pos.y,Drone.x-Pbots(2).pos.x);
%               theta42=atan2(Pbots(2).pos.y-Drone.y,Pbots(2).pos.x-Drone.x);
theta34=atan2(Drone.y-Pbots(3).pos.y,Drone.x-Pbots(3).pos.x);
%               theta43=atan2(Pbots(3).pos.y-Drone.y,Pbots(3).pos.x-Drone.x);

%                                   Error
phi1=[Beta12-SP_D;Beta13-SP_D;Beta14-SP_DD];
phi2=[Beta12-SP_D;Beta23-SP_D;Beta24-SP_DD];
phi3=[Beta13-SP_D;Beta23-SP_D;Beta34-SP_DD];

%               phi4=[Beta14-SP_DD;Beta24-SP_DD;Beta34-SP_DD];

gamma12=(Beta12+SP_D)/(2*Beta12^2);
gamma13=(Beta13+SP_D)/(2*Beta13^2);
gamma23=(Beta23+SP_D)/(2*Beta23^2);

gamma14=(Beta14+SP_DD)/(2*Beta14^2);
gamma24=(Beta24+SP_DD)/(2*Beta24^2);
gamma34=(Beta34+SP_DD)/(2*Beta34^2);

k1=[gamma12*cos(theta12) gamma12*sin(theta12);...
gamma13*cos(theta13) gamma13*sin(theta13);...
gamma14*cos(theta14) gamma14*sin(theta14)];
k2=[gamma12*cos(theta21) gamma12*sin(theta21);...
gamma23*cos(theta23) gamma23*sin(theta23);...
gamma24*cos(theta24) gamma24*sin(theta24)];
k3=[gamma13*cos(theta31) gamma13*sin(theta31);...
gamma23*cos(theta32) gamma23*sin(theta32);...
gamma34*cos(theta34) gamma34*sin(theta34)];

%               k4=[gamma14*cos(theta41) gamma14*sin(theta41);...
%                   gamma24*cos(theta42) gamma24*sin(theta42);...
%                   gamma34*cos(theta43) gamma34*sin(theta43)];

%%                                 Control Variables
%               NewWandPos=track.get('Wand');
%               Drone_dt=[NewWandPos.x-WandPos.x;NewWandPos.y-WandPos.y]/(t-toc);

t=toc;
%              M=[C(1)+rc*cos(wc*t);C(2)+rc*sin(wc*t)];
%              Md=[-wc*rc*sin(wc*t);wc*rc*cos(wc*t)];
%              Mdd=[-wc*wc*rc*cos(wc*t);-wc*wc*rc*sin(wc*t)];
M=[0;0];
Md=[0;0];
Mdd=[0;0];

u4.x=Mdd(1)-0.001*rhod*(Drone_dt.x-Md(1))-0.001*kr*(Drone.x-M(1));
u4.y=Mdd(2)-(0.001*rhod*(Drone_dt.y-Md(2)))-(0.001*kr*(Drone.y-M(2)));
u4.z=-1*Drone_dt.z-(0.25*(Drone.z-SP_z4));
u4.psi=-kTH*((Drone.psi-Pbots(1).pos.psi)+(Drone.psi-Pbots(2).pos.psi)+(Drone.psi-Pbots(3).pos.
    psi));
%               u4.psi=-kTH*(Drone.psi-0)-rhod*Drone_dt.psi;

Drone_dt.x=Drone_dt_past.x+(t-t_past)*((u4.x+u4_past.x)/2);
Drone_dt.y=Drone_dt_past.y+(t-t_past)*((u4.y+u4_past.y)/2);
Drone_dt.z=Drone_dt_past.z+(t-t_past)*((u4.z+u4_past.z)/2);
Drone_dt.psi=Drone_dt_past.psi+(t-t_past)*((u4.psi+u4_past.psi)/2);
Drone.x=Drone_past.x+(t-t_past)*((Drone_dt.x+Drone_dt_past.x)/2);
Drone.y=Drone_past.y+(t-t_past)*((Drone_dt.y+Drone_dt_past.y)/2);
Drone.z=Drone_past.z+(t-t_past)*((Drone_dt.z+Drone_dt_past.z)/2);
Drone.psi=Drone_past.psi+(t-t_past)*((Drone_dt.psi+Drone_dt_past.psi)/2);
Drone_dt_past=Drone_dt;
Drone_past=Drone;
u4_past=u4;
t_past=t;

u1=[(kpO*k1'*phi1+Md);-kTH*((Pbots(1).pos.psi-Pbots(2).pos.psi)+(Pbots(1).pos.psi-Pbots(3).pos.
    psi)+(Pbots(1).pos.psi-Drone.psi))];
u2=[(kpO*k2'*phi2+Md);-kTH*((Pbots(2).pos.psi-Pbots(1).pos.psi)+(Pbots(2).pos.psi-Pbots(3).pos.
    psi)+(Pbots(2).pos.psi-Drone.psi))];
u3=[(kpO*k3'*phi3+Md);-kTH*((Pbots(3).pos.psi-Pbots(1).pos.psi)+(Pbots(3).pos.psi-Pbots(2).pos.
    psi)+(Pbots(3).pos.psi-Drone.psi))];

%               WandPos=NewWandPos;

%%                                      OMNITRI 1
A=[cos(Pbots(1).pos.psi) -sin(Pbots(1).pos.psi) 0;sin(Pbots(1).pos.psi) cos(Pbots(1).pos.psi)
    0;0 0 1];
errorpos=[u1(1); u1(2)];
errorang=u1(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u1(1)^2+u1(2)^2+eps)*[u1(1); u1(2)];
```

```
FNormang=Nw/sqrt(u1(3)^2+eps)*u1(3);
FNorm=[FNormpos; FNormang];
U1=A\FNorm;
Pbots(1).move(U1(1),U1(2),U1(3));

%%                                  OMNITRI 2
A=[cos(Pbots(2).pos.psi) -sin(Pbots(2).pos.psi) 0;sin(Pbots(2).pos.psi) cos(Pbots(2).pos.psi)
    0;0 0 1];
errorpos=[u2(1); u2(2)];
errorang=u2(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u2(1)^2+u2(2)^2+eps)*[u2(1); u2(2)];
FNormang=Nw/sqrt(u2(3)^2+eps)*u2(3);
FNorm=[FNormpos; FNormang];
U2=A\FNorm;
Pbots(2).move(U2(1),U2(2),U2(3));

%%                                  OMNITRI 3
A=[cos(Pbots(3).pos.psi) -sin(Pbots(3).pos.psi) 0;sin(Pbots(3).pos.psi) cos(Pbots(3).pos.psi)
    0;0 0 1];
errorpos=[u3(1); u3(2)];
errorang=u3(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u3(1)^2+u3(2)^2+eps)*[u3(1); u3(2)];
FNormang=Nw/sqrt(u3(3)^2+eps)*u3(3);
FNorm=[FNormpos; FNormang];
U3=A\FNorm;
Pbots(3).move(U3(1),U3(2),U3(3));

%%                                  SAVE DATA
DRONE(:,iteration)=[Drone.x;Drone.y;Drone.z;Drone.psi];
SRM1(:,iteration)=[Pbots(1).pos.x;Pbots(1).pos.y;Pbots(1).pos.psi];
SRM2(:,iteration)=[Pbots(2).pos.x;Pbots(2).pos.y;Pbots(2).pos.psi];
SRM3(:,iteration)=[Pbots(3).pos.x;Pbots(3).pos.y;Pbots(3).pos.psi];
ControlSRM1(:,iteration)=[U1(1);U1(2);U1(3)];
ControlSRM2(:,iteration)=[U2(1);U2(2);U2(3)];
ControlSRM3(:,iteration)=[U3(1);U3(2);U3(3)];
ControlDRONE(:,iteration)=[u4.x;u4.y;u4.z;u4.psi];
ErrorM(:,iteration)=[Drone.x-M(1);Drone.y-M(2)];
Time(:,iteration)=t;
%               TRAY(:,iteration)=[M(1);M(2);1000];
%               plot3(TRAY(1,:),TRAY(2,:),TRAY(3,:));
%%                                      PLOT RESULTS
%               hold on
%               plot3(DRONE(1,:),DRONE(2,:),DRONE(3,:));
%               plot3(SRM1(1,:),SRM1(2,:),SRM1(3,:));
%               plot3(SRM2(1,:),SRM2(2,:),SRM2(3,:));
%               plot3(SRM3(1,:),SRM3(2,:),SRM3(3,:));
%               hold off
%               axis equal;
%               axis([-1500 1500 -3500 3500 0 2000]);
%               grid on;
%               box on;
end
catch
disp('ERROR');
end
%%                                      FINIT Plot
%     hold off
%%                                      FINIT Robots
Pbots(1).move(0,0,0);
Pbots(2).move(0,0,0);
Pbots(3).move(0,0,0);
Pbots(1).disconnect;
Pbots(2).disconnect;
Pbots(3).disconnect;
%     %%                                    FINIT Tracker
%     track.off;
end



%****************************
%    By Pablo Paniagua      *
%    Last update 07/09/2016 *
%****************************
%    Heterogeneous          *
%****************************
```

```
%
%           Pbots(1)=PAXBOT('PAXBOT1');
%           Pbots(2)=PAXBOT('PAXBOT2');
%           Pbots(3)=PAXBOT('PAXBOT3');
%           Pbots(1).createBT;
%           Pbots(2).createBT;
%           Pbots(3).createBT;
%*****************
function [DRONE,SRM1,SRM2,SRM3,ControlSRM1,ControlSRM2,ControlSRM3,Time]=TesisRD(Pbots,track)
%%                                      INIT Joystick
j=vrjoystick(1);
%%                                      INIT Tracker
%     track=TRACKER();
%     track.on;
%%                                      INIT Robots
Pbots(1).connect;
Pbots(2).connect;
Pbots(3).connect;
%%                                      Return values
maxIteration=1000;
DRONE=zeros(4,maxIteration);
SRM1=zeros(3,maxIteration);
SRM2=zeros(3,maxIteration);
SRM3=zeros(3,maxIteration);
ControlSRM1=zeros(3,maxIteration);
ControlSRM2=zeros(3,maxIteration);
ControlSRM3=zeros(3,maxIteration);
Time=zeros(1,maxIteration);
%%                                      Control Parameters
SP_D=1000;
SP_DD=500/cos(pi/6);
%     SP_z4=1000;
kpO=1000;
%     kr=500;
%     rhod=500;                 %Ganancia derivativa
kTH=.2;
eps=0.0001;
%%                                      TRAYECTORY
%     wc=2*pi/35;               %Frecuencia
%     rc=500;                   %Radio de la circunferencia
%     C=[0 0];                  %Centro del circulo
%%                                      DRONE/WAND DATA
%     droneName='Wand';
droneName='Wand';
Drone=track.get(droneName);
Drone_past=Drone;
%     Drone=COORDINATES;
%     Drone.x=C(1)+rc;
%     Drone.y=C(2);
%     Drone.z=0;
%     Drone.psi=0;
%     Drone_past=Drone;
%     Drone_dt=COORDINATES;
%     Drone_dt_past=COORDINATES;
%     u4_past=COORDINATES;
%%                                      TIMER
tic;
t_past=0;
%     try
iteration=0;
%%                                  Loop
while((~button(j,8))&&(iteration ~= maxIteration))
iteration=iteration+1;
disp(iteration);
pause(.08);
%%                              Get Vicon
Pbots(1).update(track);
Pbots(2).update(track);
Pbots(3).update(track);
%%                              Variables
Beta12=norm([Pbots(2).pos.x,Pbots(2).pos.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta13=norm([Pbots(3).pos.x,Pbots(3).pos.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta23=norm([Pbots(3).pos.x,Pbots(3).pos.y]-[Pbots(2).pos.x,Pbots(2).pos.y]);

Beta14=norm([Drone.x,Drone.y]-[Pbots(1).pos.x,Pbots(1).pos.y]);
Beta24=norm([Drone.x,Drone.y]-[Pbots(2).pos.x,Pbots(2).pos.y]);
Beta34=norm([Drone.x,Drone.y]-[Pbots(3).pos.x,Pbots(3).pos.y]);


theta12=atan2(Pbots(2).pos.y-Pbots(1).pos.y,Pbots(2).pos.x-Pbots(1).pos.x);
theta13=atan2(Pbots(3).pos.y-Pbots(1).pos.y,Pbots(3).pos.x-Pbots(1).pos.x);
theta21=atan2(Pbots(1).pos.y-Pbots(2).pos.y,Pbots(1).pos.x-Pbots(2).pos.x);
theta23=atan2(Pbots(3).pos.y-Pbots(2).pos.y,Pbots(3).pos.x-Pbots(2).pos.x);
theta31=atan2(Pbots(1).pos.y-Pbots(3).pos.y,Pbots(1).pos.x-Pbots(3).pos.x);
theta32=atan2(Pbots(2).pos.y-Pbots(3).pos.y,Pbots(2).pos.x-Pbots(3).pos.x);

theta14=atan2(Drone.y-Pbots(1).pos.y,Drone.x-Pbots(1).pos.x);
%           theta41=atan2(Pbots(1).pos.y-Drone.y,Pbots(1).pos.x-Drone.x);
theta24=atan2(Drone.y-Pbots(2).pos.y,Drone.x-Pbots(2).pos.x);
%           theta42=atan2(Pbots(2).pos.y-Drone.y,Pbots(2).pos.x-Drone.x);
theta34=atan2(Drone.y-Pbots(3).pos.y,Drone.x-Pbots(3).pos.x);
%           theta43=atan2(Pbots(3).pos.y-Drone.y,Pbots(3).pos.x-Drone.x);

%                           Error
phi1=[Beta12-SP_D;Beta13-SP_D;Beta14-SP_DD];
phi2=[Beta12-SP_D;Beta23-SP_D;Beta24-SP_DD];
phi3=[Beta13-SP_D;Beta23-SP_D;Beta34-SP_DD];

%           phi4=[Beta14-SP_DD;Beta24-SP_DD;Beta34-SP_DD];
```

```
gamma12=(Beta12+SP_D)/(2*Beta12^2);
gamma13=(Beta13+SP_D)/(2*Beta13^2);
gamma23=(Beta23+SP_D)/(2*Beta23^2);

gamma14=(Beta14+SP_DD)/(2*Beta14^2);
gamma24=(Beta24+SP_DD)/(2*Beta24^2);
gamma34=(Beta34+SP_DD)/(2*Beta34^2);

k1=[gamma12*cos(theta12) gamma12*sin(theta12);...
gamma13*cos(theta13) gamma13*sin(theta13);...
gamma14*cos(theta14) gamma14*sin(theta14)];
k2=[gamma12*cos(theta21) gamma12*sin(theta21);...
gamma23*cos(theta23) gamma23*sin(theta23);...
gamma24*cos(theta24) gamma24*sin(theta24)];
k3=[gamma13*cos(theta31) gamma13*sin(theta31);...
gamma23*cos(theta32) gamma23*sin(theta32);...
gamma34*cos(theta34) gamma34*sin(theta34)];

%               k4=[gamma14*cos(theta41) gamma14*sin(theta41);...
%                   gamma24*cos(theta42) gamma24*sin(theta42);...
%                   gamma34*cos(theta43) gamma34*sin(theta43)];

%%                              Control Variables
t=toc;
Drone=track.get(droneName);
Md=[Drone.x-Drone_past.x;Drone.y-Drone_past.y]/(t-t_past);

%               M=[C(1)+rc*cos(wc*t);C(2)+rc*sin(wc*t)];
%               Md=[-wc*rc*sin(wc*t);wc*rc*cos(wc*t)];
%               Mdd=[-wc*wc*rc*cos(wc*t);-wc*wc*rc*sin(wc*t)];
%
%               u4.x=Mdd(1)-0.001*rhod*(Drone_dt.x-Md(1))-0.001*kr*(Drone.x-M(1));
%               u4.y=Mdd(2)-(0.001*rhod*(Drone_dt.y-Md(2)))-(0.001*kr*(Drone.y-M(2)));
%               u4.z=-1*Drone_dt.z-(0.25*(Drone.z-SP_z4));
%               u4.psi=-kTH*((Drone.psi-Pbots(1).pos.psi)+(Drone.psi-Pbots(2).pos.psi)+(Drone.psi-
    Pbots(3).pos.psi));
%               u4.psi=-kTH*(Drone.psi-0)-rhod*Drone_dt.psi;
%
%               Drone_dt.x=Drone_dt_past.x+(t-t_past)*((u4.x+u4_past.x)/2);
%               Drone_dt.y=Drone_dt_past.y+(t-t_past)*((u4.y+u4_past.y)/2);
%               Drone_dt.z=Drone_dt_past.z+(t-t_past)*((u4.z+u4_past.z)/2);
%               Drone_dt.psi=Drone_dt_past.psi+(t-t_past)*((u4.psi+u4_past.psi)/2);
%               Drone.x=Drone_past.x+(t-t_past)*((Drone_dt.x+Drone_dt_past.x)/2);
%               Drone.y=Drone_past.y+(t-t_past)*((Drone_dt.y+Drone_dt_past.y)/2);
%               Drone.z=Drone_past.z+(t-t_past)*((Drone_dt.z+Drone_dt_past.z)/2);
%               Drone.psi=Drone_past.psi+(t-t_past)*((Drone_dt.psi+Drone_dt_past.psi)/2);
%               Drone_dt_past=Drone_dt;
%               u4_past=u4;
Drone_past=Drone;
t_past=t;

SP_Psi=Drone.psi;
u1=[(kpO*k1'*phi1+Md);-kTH*(Pbots(1).pos.psi-SP_Psi)];
u2=[(kpO*k2'*phi2+Md);-kTH*(Pbots(2).pos.psi-SP_Psi)];
u3=[(kpO*k3'*phi3+Md);-kTH*(Pbots(3).pos.psi-SP_Psi)];

%%                                      OMNITRI 1
A=[cos(Pbots(1).pos.psi) -sin(Pbots(1).pos.psi) 0;sin(Pbots(1).pos.psi) cos(Pbots(1).pos.psi)
    0;0 0 1];
errorpos=[u1(1); u1(2)];
errorang=u1(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u1(1)^2+u1(2)^2+eps)*[u1(1); u1(2)];
FNormang=Nw/sqrt(u1(3)^2+eps)*u1(3);
FNorm=[FNormpos; FNormang];
U1=A\FNorm;
Pbots(1).move(U1(1),U1(2),U1(3));

%%                                      OMNITRI 2
A=[cos(Pbots(2).pos.psi) -sin(Pbots(2).pos.psi) 0;sin(Pbots(2).pos.psi) cos(Pbots(2).pos.psi)
    0;0 0 1];
errorpos=[u2(1); u2(2)];
errorang=u2(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u2(1)^2+u2(2)^2+eps)*[u2(1); u2(2)];
FNormang=Nw/sqrt(u2(3)^2+eps)*u2(3);
FNorm=[FNormpos; FNormang];
U2=A\FNorm;
```

```
Pbots(2).move(U2(1),U2(2),U2(3));

%%                                      OMNITRI 3
A=[cos(Pbots(3).pos.psi) -sin(Pbots(3).pos.psi) 0;sin(Pbots(3).pos.psi) cos(Pbots(3).pos.psi)
    0;0 0 1];
errorpos=[u3(1); u3(2)];
errorang=u3(3);
if(abs(norm(errorpos))<=50)
N=0;
else
N=7000;
end

if(abs(errorang)<=0.0873)
Nw=0;
else
Nw=10;
end
FNormpos=N/sqrt(u3(1)^2+u3(2)^2+eps)*[u3(1); u3(2)];
FNormang=Nw/sqrt(u3(3)^2+eps)*u3(3);
FNorm=[FNormpos; FNormang];
U3=A\FNorm;
Pbots(3).move(U3(1),U3(2),U3(3));

%%                                      SAVE DATA
DRONE(:,iteration)=[Drone.x;Drone.y;Drone.z;Drone.psi];
SRM1(:,iteration)=[Pbots(1).pos.x;Pbots(1).pos.y;Pbots(1).pos.psi];
SRM2(:,iteration)=[Pbots(2).pos.x;Pbots(2).pos.y;Pbots(2).pos.psi];
SRM3(:,iteration)=[Pbots(3).pos.x;Pbots(3).pos.y;Pbots(3).pos.psi];
ControlSRM1(:,iteration)=[U1(1);U1(2);U1(3)];
ControlSRM2(:,iteration)=[U2(1);U2(2);U2(3)];
ControlSRM3(:,iteration)=[U3(1);U3(2);U3(3)];
Time(:,iteration)=t;
end
%     catch
%         disp('ERROR');
%     end
%%                                      FINIT Robots
Pbots(1).move(0,0,0);
Pbots(2).move(0,0,0);
Pbots(3).move(0,0,0);
Pbots(1).disconnect;
Pbots(2).disconnect;
Pbots(3).disconnect;
%%                                      FINIT Tracker
%     track.off;
end
```

Bibliografía

[1] G. Antonelli, F. Arrichiello, F. Caccavale, and A. Marino. Decentralized time-varying formation control for multi-robot systems. *Journal of Robotics Research*, 33(7):1029–1043, 2014.

[2] T. Arai, E. Pagello, and L. Parker. Guest editorial advances in multirobot systems. *Robotics and Automation, IEEE Transactions on*, 18(5):655–661, 2002.

[3] Y. Asahiro, H. Asama, I. Suzuki, and M. Yamashita. Improvement of distributed control algorithms for robots carrying an object. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 608–613. IEEE, 1999.

[4] K. R. Baghaei and A. Agah. Task allocation methodologies for multi-robot systems, 2003.

[5] T. Balch and R. Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[6] L. Barthuber, K. Fyraj, and P. Firsching. Test concept for a mobile robot with optimized traction. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–7, June 2016.

[7] F. Belkhouche and B. Belkhouche. Modeling and controlling a robotic convoy using guidance laws strategies. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(4):813–825, Aug 2005.

[8] Y. Cao, A. Fukunaga, A. Kahng, and F. Meng. Cooperative mobile robotics: antecedents and directions. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 226–234 vol.1, 1995.

[9] Y. U. Cao, A. S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Auton. Robots*, 4(1):34–46, Mar. 1997.

[10] Y.-Q. Chen and Z. Wang. Formation control: a review and a new consideration. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3181–3186, 2005.

[11] T. R. Consi, B. R. Ardaugh, T. R. Erdmann, M. Matsen, M. Peterson, J. Ringstad, A. Vechart, and C. Verink. An amphibious robot for surf zone science and environmental monitoring. In *OCEANS 2009*, pages 1–7, Oct 2009.

[12] L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques. On the control of a leader-follower formation of nonholonomic mobile robots. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5992–5997, Dec 2006.

[13] A. Crespi, K. Karakasiliotis, A. Guignard, and A. J. Ijspeert. Salamandra robotica ii: An amphibious robot to study salamander-like swimming and walking gaits. *IEEE Transactions on Robotics*, 29(2):308–320, April 2013.

[14] C. C. de Wit, G. Bastin, and B. Siciliano, editors. *Theory of Robot Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[15] J. Desai. A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525, 2002.

[16] J. P. Desai, J. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 2864–2869 vol.4, May 1998.

[17] S. Dhull, D. Canelon, A. Kottas, J. Dancs, A. Carlson, and N. Papanikolopoulos. Aquapod: A small amphibious robot with sampling capabilities. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 100–105, Oct 2012.

[18] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[19] T. Dierks and S. Jagannathan. Control of nonholonomic mobile robot formations: Backstepping kinematics into dynamics. In *2007 IEEE International Conference on Control Applications*, pages 94–99, Oct 2007.

[20] K. D. Do. Formation tracking control of unicycle-type mobile robots. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2391–2396, April 2007.

[21] B. Fidan, V. Gazi, S. Zhai, and N. Cen. Single-view distance-estimation-based formation control of robotic swarms. *IEEE Transactions on Industrial Electronics*, 60(12):5781–5791, 2013.

[22] J. Fredslund and M. Mataric. General algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[23] Y. C. Fu, S. Hirose, G. Endo, and E. F. Fukushima. Design and basic experiments of a waterproof mobile robot propelled by rs-wave mechanism. In *2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 218–223, Nov 2013.

[24] B. Gerkey and M. Mataric. Pusher-watcher: an approach to fault-tolerant tightly-coupled robot coordination. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 464–469 vol.1, 2002.

[25] J. Guo, G. Wu, and S. Guo. Study on movement stability for the spherical amphibious robot. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 55–60, Aug 2016.

[26] S. Guo, S. Sun, and J. Guo. The application of image mosaic in information collecting for an amphibious spherical robot system. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 1547–1552, Aug 2016.

[27] P. D. Healy and B. E. Bishop. Sea-dragon: An amphibious robot for operation in the littorals. In *2009 41st Southeastern Symposium on System Theory*, pages 266–270, March 2009.

[28] J. M. Hendrickx, C. Yu, B. Fidan, and B. D. O. Anderson. Rigidity and persistence of meta-formations. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5980–5985, Dec 2006.

[29] E. Hernandez-Martinez, G. Fernandez-Anaya, E. Ferreira, J. Flores-Godoy, and A. Lopez-Gonzalez. Trajectory tracking of a quadcopter uav with optimal translational control**the authors acknowledgments the financial support from cona-cyt and universidad iberomaericana, mexico city. *IFAC-PapersOnLine*, 48(19):226 – 231, 2015.

[30] E. Hernandez-Martinez, J. Flores-Godoy, and G. Fernandez-Anaya. Decentralized discrete-time formation control for multirobot systems. *Discrete Dynamics in Nature and Society*, 2013:1–8, 2013.

[31] E. Hernández-Martínez and E. Aranda-Bricaire. Convergence and collision avoidance in formation control: A survey of the artificial potential functions approach. InTech.

[32] E. G. Hernández-Martínez and E. Aranda-Bricaire. Marching control of unicycles based on the leader-followers scheme. In *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, pages 2265–2270, Nov 2009.

[33] R. Ikhankar, V. Kuthe, S. Ulabhaje, S. Balpande, and M. Dhadwe. Pibot: The raspberry pi controlled multi-environment robot for surveillance amp; live streaming. In *Industrial Instrumentation and Control (ICIC), 2015 International Conference on*, pages 1402–1405, May 2015.

[34] L. Iocchi, D. Nardi, and M. Salerno. Reactivity and deliberation: a survey on multi-robot systems. In *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 9–32. Springer, 2000.

[35] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[36] L. Jia, Z. Hu, L. Geng, Y. Yang, and C. Wang. The concept design of a mobile amphibious spherical robot for underwater operation. In *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 411–415, June 2016.

[37] X. Jia, M. Frenger, Z. Chen, W. R. Hamel, and M. Zhang. An alligator inspired modular robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1949–1954, May 2015.

[38] L. Krick, M. Broucke, and B. Francis. Stabilization of infinitesimally rigid formations of multi-robot networks. In *IEEE Conference on Decision and Control*, pages 477–482, 2008.

[39] G. Lafferriere, J. Caughman, and A. Williams. Graph theoretic methods in the stability of vehicle formations. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3729–3734 vol.4, June 2004.

[40] N. LéChevin, C. A. Rabbath, and P. Sicard. Trajectory tracking of leader-follower formations characterized by constant line-of-sight angles. *Automatica*, 42(12):2131–2141, Dec. 2006.

[41] Y. Li and X. Chen. Stability on multi-robot formation with dynamic interaction topologies. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 394–399. IEEE, 2005.

[42] X. Liang, M. Xu, L. Xu, P. Liu, X. Ren, Z. Kong, J. Yang, and S. Zhang. The amphihex: A novel amphibious robot with transformable leg-flipper composite propulsion mechanism. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3667–3672, Oct 2012.

[43] A. Lopez-Gonzalez, E. Ferreira, E. Hernandez-Martinez, J. Flores-Godoy, G. Fernandez-Anaya, and P. Paniagua-Contro. Multi-robot formation control using distance and orientation. *Advanced Robotics*, 30(14):901–913, 2016.

[44] M. McPartland, S. Nolfi, and H. A. Abbass. Emergence of communication in competitive multi-agent systems: a pareto multi-objective approach. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 51–58, New York, NY, USA, 2005. ACM.

[45] K.-K. Oh, M.-C. Park, and H.-S. Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[46] R. Olfati-Saber and R. Murray. Distributed cooperative control of multiple vehicle formations using structural potential functions. In *IFAC World Congress*, pages 346–352, 2002.

[47] L. Parker. Current state of the art in distributed autonomous mobile robotics. Oak Ridge National Laboratory.

[48] S. Popesku, E. Meister, F. Schlachter, and P. Levi. Active wheel - an autonomous modular robot. In *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 97–102, Nov 2013.

[49] M. Z. A. Rashid, M. S. M. Aras, H. N. M. Shah, W. T. Lim, and Z. Ibrahim. Design and system parameter's validation of the unicycle mobile robot. In *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*, pages 311–316, Nov 2012.

[50] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and C. T. Abdallah. Experimental implementation of flocking algorithms in wheeled mobile robots. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 4917–4922 vol. 7, June 2005.

[51] W. Ren and R. Beard. *Distributed Consensus in Multi-vehicle Cooperative Control*. Springer, London, 2008.

[52] P. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini. Communication strategies in multi-robot search and retrieval. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, pages 301–310, Toulouse, France, 2004.

[53] S. Salazar-Cruz and R. Lozano. Stabilization and nonlinear control for a novel trirotor mini-aircraft. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2612–2617, April 2005.

[54] L. Shi, K. Tang, S. Guo, X. Bao, S. Pan, and P. Guo. Leader-follower cooperative movement method for multiple amphibious spherical robots. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 593–598, Aug 2016.

[55] Y. Tang, A. Zhang, and J. Yu. Modeling and optimization of wheel-propeller-leg integrated driving mechanism for an amphibious robot. In *2009 Second International Conference on Information and Computing Science*, volume 4, pages 73–76, May 2009.

[56] H. G. Tanner, G. J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455, June 2004.

[57] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35, 2002.

[58] V. Trianni, T. H. Labella, and M. Dorigo. Evolution of direct communication for a swarm-bot performing hole avoidance. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 130–141. Springer, 2004.

[59] Z.-D. Wang, Y. Takano, Y. Hirata, and K. Kosuge. A pushing leader based decentralized control method for cooperative object transportation. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 1035–1040. IEEE, 2004.

[60] S. Yamada and J. Saito. Adaptive action selection without explicit communication for multirobot box-pushing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(3):398–404, 2001.

[61] H. Yamaguchi. A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations. *Robotics and Autonomous Systems*, 43(4):257–282, 2003.

[62] B. Yang, L. Han, G. Li, W. Xu, and B. Hu. A modular amphibious snake-like robot: Design, modeling and simulation. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1924–1929, Dec 2015.

[63] Y. Yi, Z. Geng, Z. Jianqing, C. Siyuan, and F. Mengyin. Design, modeling and control of a novel amphibious robot with dual-swing-legs propulsion mechanism. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 559–566, Sept 2015.

[64] M. Yogeswaran and S. Ponnambalam. Swarm robotics:an extensive research review. InTech.

[65] J. Yu and S. M. LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *IEEE Conference on Decision and Control*, pages 4023–4028, 2012.