CINVESTAV
Unidad Guadalajara

DOCUMENTO DE REQUISITOS Y DISEÑO DE SOFTWARE (DRDS)

Proyecto:

Sistema Multiagentes: Coordinación de Robots Hexapodos.

Versión: **1.0**

Alumnos:

Ricardo Talavera Sevilla
Emmanuel Castro Vargas
Diego Orozco Castillo
Heiler Duarte Moreno
Daniel Irineo Estrada Sánchez
Monica Fabiola Perales Tejeda
Victor Manuel Peláez Fernández

Materia:

Sistemas Distribuidos II

Profesor:

Dr. Félix Francisco Ramos Corchado

Zapopan, Jalisco, México agosto de 2024.

HOJA DE CONTROL

Organismo	CINVESTAV - Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional Unidad Guadalajara.
Proyecto	Sistema Multiagentes: Coordinación de Robots Hexapodos.
Entregable	Documento de Requisitos y Diseño de Software (DRDS)
Autor	Castro Vargas Emmanuel Duarte Moreno Heiler Estrada Sánchez Daniel Irineo Orozco Castillo Diego Peláez Fernández Victor Manuel Perales Tejeda Mónica Fabiola Talavera Sevilla Ricardo
Aprobado por	Dr. Félix Francisco Ramos Corchado
Fecha Aprobación	agosto de 2024

REGISTRO DE CAMBIOS

Versión	**Causa**	**Responsable**	**Fecha**
1.0	Versión inicial	Castro Vargas Emmanuel Duarte Moreno Heiler Estrada Sánchez Daniel Irineo Orozco Castillo Diego Peláez Fernández Victor Manuel Perales Tejeda Mónica Fabiola Talavera Sevilla Ricardo	2024-04-08

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

El presente Documento de Requisitos y Diseño de Software (DRDS), que combina las Especificaciones de Requisitos de Software (SRS) y el Diseño de Software (SDD), ofrece una descripción detallada tanto de los requerimientos funcionales como no funcionales para el proyecto titulado *Sistema de Control para la Coordinación y Ejecución de Rutinas entre Robots Hexápodos*.

1.1. Propósito

Dicho documento pretende servir como guía para el equipo de desarrollo; al facilitar el registro de los pasos a seguir en cada una de sus etapas, así como para el usuario que desee conocer el proceso y los elementos que condujeron al sistema resultante.
El propósito del sistema es demostrar una aplicación robótica del enfoque multiagente.

1.2. Alcance

El sistema debe controlar al Robot Hexápodo mediante la abstracción de un entorno para poder generar una rutina de movimiento para el Robot para llegar a un objetivo mediante la utilización de una Red LAN inalámbrica.

- Interacción con el Usuario: Proporciona una interfaz gráfica de usuario (GUI) intuitiva para interactuar con los comandos del Hexápodo.
- La generación de rutina: La rutina se generada mediante el análisis de una imagen que proporcione una ubicación objetivo.

1.3. Limitaciones

El sistema estará limitado a comandos de Avanzar, Girar a la derecha y Girar a la izquierda.

- Dependencia de conexión a una red Local: La conexión al Robot Hexápodo esta configurada para poder conectarse a un servidor que se encuentre en la misma red Wi-Fi.
- Rendimiento bajo Condiciones de Red Variables: La experiencia del usuario puede verse afectada negativamente por condiciones de red inestables o lentas.
- Escalabilidad: La escalabilidad es dependiente de la cantidad de Robots que se tenga, debido a que la generación de rutinas es creado por un servidor y este debe adecuarse para evitar colisiones entre Robot Hexapodos.
- Adaptabilidad: Este sistema solo es un coordinador para rutinas predeterminadas denominadas bailes por lo que no es apto para cooperación para tareas complejas.

1.4. Personal Involucrado

Esta sección detalla los roles y responsabilidades de los individuos y grupos involucrados en el proyecto de desarrollo de software. La identificación clara del personal asegura una asignación efectiva de tareas y facilita la comunicación dentro del equipo de proyecto.

Cuadro 1.1: Roles y Responsabilidades

Rol	**Responsabilidades**
Stakeholders	Incluyen a los patrocinadores del proyecto, usuarios finales y cualquier persona o entidad que tenga un interés directo o indirecto en el proyecto. Su responsabilidad principal es proporcionar requisitos, expectativas y retroalimentación al equipo de desarrollo.
Equipo de Desarrollo de software	Responsable de diseñar, implementar y probar el software según las especificaciones del SRS. Este equipo trabaja estrechamente con los stakeholders para asegurar que el producto final cumpla con los requisitos establecidos.
Equipo de pruebas de Hardware	Define las características de las características que cumple el equipo de hardware que se tiene, para definir los planes para el desarrollo del proyecto.
Analista de Sistemas	Responsable de comprender y documentar los requisitos del negocio y traducirlos en requisitos técnicos para el equipo de desarrollo.

Cuadro 1.2: Personal Involucrado 1

Nombre	Dr Félix Francisco Ramos Corchado
Rol	Stakeholders
Posición	Doctor en Ciencias.
Responsabilidad	Validar la efectividad del desarrollo de actividades, así como la calidad de cada tarea terminada.
Email	felix.ramos@cinvestav.mx

Cuadro 1.3: Personal Involucrado 2

Nombre	Estrada Sánchez Daniel Irineo
Rol	Equipo de pruebas de Hardware y Equipo de desarrollo
Posición	Estudiante de Maestría.
Responsabilidad	Asegurar que el hardware funcione de manera correcta para las pruebas efectuadas por el equipo de desarrollo de software. También en el desarrollo de los algoritmos y control de movimiento del robot.
Email	daniel.estrada@cinvestav.mx

Cuadro 1.4: Personal Involucrado 3

Nombre	Duarte Moreno Heiler Orozco Castillo Diego Peláez Fernández Victor Manuel Perales Tejeda Mónica Fabiola
Rol	Analista de Sistemas y Equipo de Arquitectura
Posición	Estudiantes de Doctorado.
Responsabilidad	Definir los requerimientos del sistema y que este cumpla con el modelo que se necesita.
Email	heiler.duarte@cinvestav.mx diego.orozco@cinvestav.mx victor.pelaez@cinvestav.mx monica.peralest@gmail.com

Cuadro 1.5: Personal Involucrado 4

Nombre	Castro Vargas Emmanuel Talavera Sevilla Ricardo
Rol	Equipo desarrollo de software
Posición	Estudiantes de Maestría.
Responsabilidad	Desarrollador.
Email	emmanuel.castro@cinvestav.mx ricardo.talavera@cinvestav.mx

1.5. Definiciones y Acrónimos

Esta sección proporciona definiciones de términos clave, y acrónimos utilizados en este documento para asegurar una comprensión clara y uniforme entre todas las partes interesadas.

1.5.1. Definiciones

Cuadro 1.6: Definiciones

Término	**Definición**
Robot Hexápodo	Un Robot Hexápodo es un vehículo mecánico que camina sobre seis patas.
BDI	Es el modelado del sistema mediante el uso de la arquitectura multiagentes para que el agente(robot) pueda modificar sus creencias y sus deseas se vean afectador por el entorno o pro lo que puede percibir de el entorno. elementos.
Agente	"Son entidades computacionales autónomas que perciben su entorno mediante sensores y actúan sobre él a través de efectores."[1].

1.6. Referencias

Esta sección proporciona referencias a documentos y estándares que son relevantes para el desarrollo y comprensión de este proyecto.

Cuadro 1.7: Referencias Documentales

Título del Documento	**Referencia**
Standard IEEE 830 - 1998	IEEE 1.6.V

1.7. Visión general del documento

Este documento está organizado en tres secciones principales, cada una diseñada para ofrecer una visión comprensiva del proyecto desde diferentes perspectivas.

La **Primer Sección** sirve como una introducción al documento, estableciendo el contexto del proyecto. Aquí se presenta una visión general, destacando los objetivos y la relevancia del sistema a desarrollar. Además, se introduce a los lectores las figuras clave involucradas en el proyecto, proporcionando una base para entender las responsabilidades y la experiencia que cada miembro aporta al equipo. Ésta sección asegura que todos los Stakeholders tengan un entendimiento común del propósito del proyecto y quiénes están detrás de su concepción y ejecución.

En la **Segunda Sección**, se aborda una descripción general del sistema propuesto, delineando sus capacidades fundamentales y las funcionalidades que se pretenden implementar. Se enfatiza en las principales funciones que el sistema llevará a cabo, proporcionando así una clara visión de lo que se espera del producto final. Esta parte también examina las restricciones y dependencias críticas que podrían influir en el desarrollo del proyecto, estableciendo así un marco de trabajo realista dentro del cual el equipo puede operar. Este segmento es crucial para alinear las expectativas y garantizar que el diseño del sistema cumpla con los requerimientos y limitaciones del entorno operativo.

La **Tercera Sección** constituye el núcleo del documento, donde se especifica el diseño y los requisitos del sistema de manera exhaustiva. Aquí se detallan tanto los requisitos funcionales, que describen las operaciones específicas y las interacciones que el sistema debe ser capaz de realizar, como los requisitos no funcionales, que abordan criterios de rendimiento, seguridad y calidad. Esta sección es fundamental para guiar el proceso de desarrollo, asegurando que el producto final no solo cumpla con las expectativas de funcionalidad sino que también adhiera a los estándares de calidad y seguridad establecidos.

Al recorrer estas tres secciones, los lectores ganan una comprensión integral del proyecto: desde su justificación y equipo hasta su implementación, diseño y los detalles técnicos específicos. Este documento actúa como una hoja de ruta, dirigiendo al equipo de desarrollo a través del complejo proceso de convertir una idea conceptual en una solución de software tangible y operativa.

2 Descripción General del Proyecto

El sistema propuesto debe coordinar, mediante un enfoque multiagente, dos robots Hexápodos que realicen movimientos de forma simultánea e independiente, empleando una arquitectura BDI ("Belief-Desire-Intention") para la representación de los estados mentales de cada uno. De igual forma, cada agente es capaz de adaptarse a su entorno y modificar sus creencias para lograr dicho objetivo.

El siguiente diagrama de flujo muestra el proceso desde que el agente busca un compañero para la ejecución de la rutina, hasta que éste se queda sin batería para continuar o completa los movimientos establecidos, pasando por el proceso que implica identificar si tiene el rol de líder, enviar y recibir la rutina.

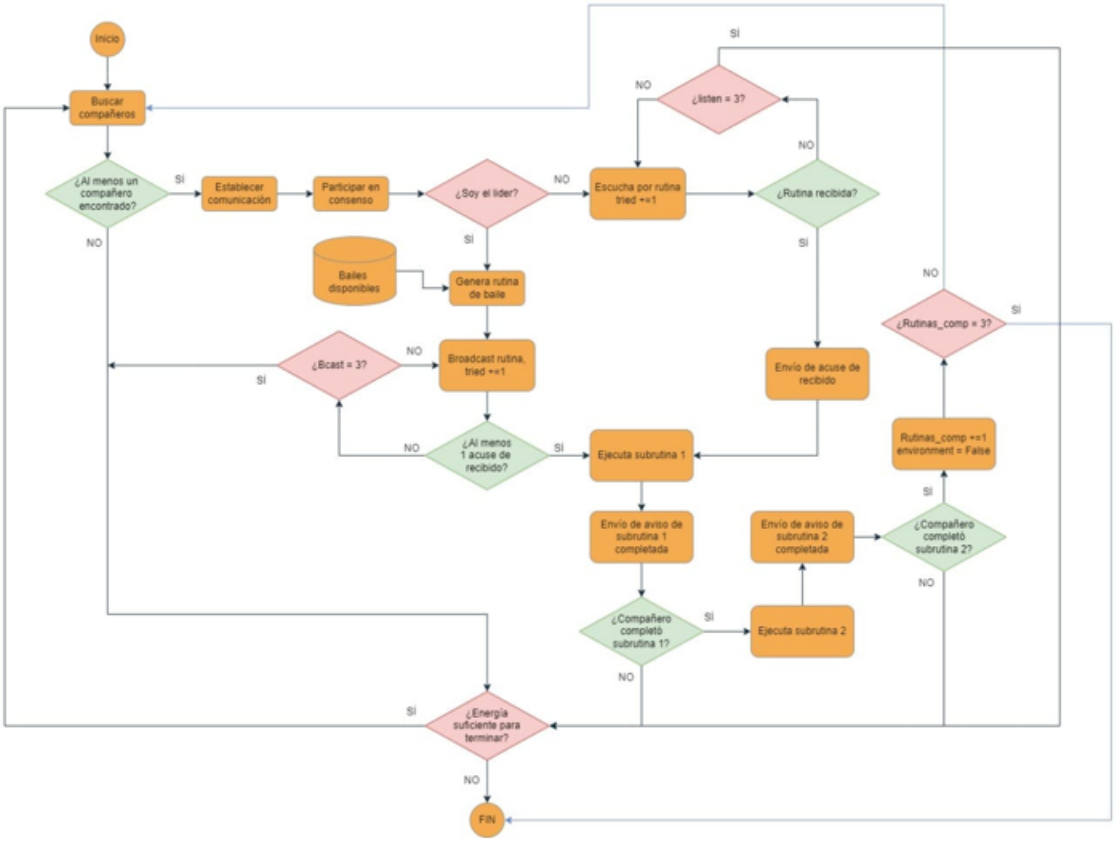

Figura 2.1: Diagrama de flujo del sistema

2.1. Visión General del Sistema

El sistema propuesto hará posible la coordinación de movimiento,a partir de un enfoque multiagentes y el uso de la arquitectura BDI y así los robot se comporten como agentes que tiene inteciones y conocimiento parcial del sistema.

2.2. Usuarios del Sistema

Los usuarios del sistema incluirán:

- **Usuarios Finales:** Individuos que requieren tener conexión a la red LAN inalámbrica a la que este conectado el Robot y serán responsables de la ejecución del programa.

2.3. Funciones del Producto

Esta sección describe las principales funciones que el sistema debe ofrecer para satisfacer las necesidades de los usuarios y cumplir con los objetivos del proyecto. Estas funciones son esenciales para el diseño y desarrollo del sistema, proporcionando una base para los requisitos funcionales detallados en capítulos posteriores.

Figura 2.2: Diagrama de Funciones

2.3.1. Seguridad

- **Red Inalámbrica LAN Privada:** Los usuarios deben tener acceso a la Red inalámbrica para poder establecer la comunicación con el Robot Hexápodo.

2.3.2. Creacion de Rutina

- **Procesamiento de Imagen:** El sistema obtendrá una imagen, de al cámara externa al robot para poder crear un reconocimiento del entorno en el que se encuentra el Robot Hexápodo.
- **Rutina de movimiento:** El sistema debe proporcionar una rutina de movimientos para que el Robot Hexápodo la ejecute.

Estas funciones del producto son fundamentales para el desarrollo del sistema de control del Robot Hexápodo.

2.4. Características de Usuario

Cuadro 2.1: Características de los Usuarios

Tipo de Usuario	**Características**
Usuarios Finales	▪ Necesitan un control de robot que cree una ruta para que la ejecute el robot. ▪ No requieren conocimientos técnicos avanzados para operar el sistema. ▪ Buscan una aplicación simple y fácil de usar.
Administradores del Sistema	▪ Responsables de la configuración, mantenimiento y actualización del sistema. ▪ Analizan y responden a incidentes. ▪ Monitorean el rendimiento del sistema y realizan ajustes para optimizarlo.
Desarrolladores	▪ Profundos conocimientos técnicos en programación y sistemas distribuidos. ▪ Contribuyen al diseño, desarrollo e implementación del sistema. ▪ Implementan características nuevas y mejoras basadas en la retroalimentación de los usuarios. ▪ Realizan pruebas para asegurar la funcionalidad, seguridad y estabilidad del sistema. ▪ Mantienen actualizada la documentación del código y guías de usuarios.

2.5. Restricciones

Este capítulo identifica las limitaciones bajo las cuales el *Sistema Multiagente: Coordinación de Robots Hexápodo* debe operar. Estas restricciones son críticas para el diseño, desarrollo e implementación del sistema, asegurando que los requisitos se alineen con las capacidades tecnológicas, políticas de seguridad y los objetivos del proyecto.

Cuadro 2.2: Restricciones del Sistema

Restricción	**Descripción**
Plataforma de Operación	El sistema está diseñado para operar específicamente en entornos Windows y Linux, Windows con los equipos que funcionan de interfaz y Linux para los dispositivos finales y de procesamiento, limitando su compatibilidad con los sistemas operativos MacOS.
Capacidad de la Red	El rendimiento del sistema está limitado por la capacidad del servidor en términos de procesamiento y ancho de banda, lo que puede afectar la velocidad de respuesta del Robot Hexápodo a la recepción de la rutina.
Tipo de Red	Se necesita una red inalámbrica (Wi-Fi) para poder generar un enlace de comunicación con el Robot Hexápodo.
Kit Freenove Hexapodo Raspberry	El proyecto esta desarrollado sobre un proyecto para principiantes proporcionado por Freenove para el uso de un robot Hexapodo y aplicación de identificación de rostros. Nuestro proyecto se enfoca en la implementación de la teoría de agentes para lograr la coordinación de dos robot Hexápodo.

3 Requisitos del Software (SRS)

3.1. Requerimientos Funcionales

A continuación se presentan los requerimientos funcionales considerados para el desarrollo del *Sistema Multiagente: Coordinación de Robots Hexápodo*.

3.2. Requerimientos No funcionales

A continuación se presentan los requerimientos no funcionales considerados para el adecuado funcionamiento del *Sistema Multiagente: Coordinación de Robots Hexapodo*.

Cuadro 3.1: Requerimientos Funcionales del Sistema

ID	Nombre	Prioridad	Descripción
RF-01	Conectividad a Internet	Alta	Se necesita acceso a una red inalámbrica (Wi-Fi) para generar un enlace de comunicación entre los Robots Hexápodos.
RF-02	Ejecución de rutinas de movimiento	Alta	Cada uno de los robots hexápodos deberá tener la capacidad de ejecutar una serie de pasos predefinidos.
RF-03	Asignación de roles a los robots	Alta	El sistema deberá funcionar de tal forma que se asignen roles a cada uno de los robots hexápodos; uno cumplirá el rol de iniciador o líder, mientras que el otro replicará la rutina del primero.
RF-04	Selección del robot líder por medio del Algoritmo del Bully	Alta	El Algoritmo del Bully es una técnica utilizada en sistemas distribuidos para asegurar que un único líder sea elegido entre un grupo de nodos. En este contexto, los nodos son robots que se comunican entre sí para coordinar sus acciones.
RF-05	Rol de robot líder o iniciador de rutina	Alta	El robot hexápodo que sea asignado el rol de líder o iniciador de la rutina, deberá ejecutar los movimientos preestablecidos de forma autónoma.
RF-06	Rol de robot seguidor de rutina	Alta	El robot hexápodo que sea asignado el rol de seguidor de la rutina, deberá ejecutar los movimientos preestablecidos por el robot lider.
RF-07	Asignación de UIDs	Alta	Cada robot debe tener un UID (Identificador Único).
RF-08	Detección de Fallos del Líder	Alta	Los robots deben detectar la ausencia del líder mediante mensajes de latido.
RF-09	Iniciación del Proceso de Elección	Alta	Los robots deben iniciar el proceso de elección al detectar la falla del líder.
RF-10	Enviar y Recibir Mensajes de Elección	Alta	Los robots deben poder enviar y recibir mensajes de Election.
RF-11	Responder a Mensajes de Elección	Alta	Los robots deben responder con mensajes de OK si están vivos.
RF-12	Declarar Liderazgo	Alta	El robot con el UID más alto debe poder declararse líder y notificar a los demás robots.
RF-13	Recepción de Mensajes de Victoria	Alta	Los robots deben actualizar su estado al recibir un mensaje de Victory.

Cuadro 3.2: Requerimientos No Funcionales del Sistema

ID	Nombre	Prioridad	Descripción
RNF-01	Portabilidad	Media	El sistema está diseñado para operar en entornos Windows y Linux, Windows con los equipos de procesamiento y Linux para los dispositivos finales, limitando su compatibilidad con los sistemas operativos MacOS.
RNF-02	Eficiencia	Media	El rendimiento del sistema está limitado por la capacidad del líder en términos de procesamiento y ancho de banda a la red, lo que puede afectar la velocidad de respuesta de los Robots Hexápodos a la recepción de las rutina. Deben tomarse las medidas adecuadas para reducir la probabilidad de deficiencia del Algoritmo del Bully, el cual debe ser eficiente en términos de tiempo y uso de recursos.
RNF-03	Consistencia	Media	Las rutinas predeterminadas enviadas por el sistema para la ejecución del Robot Hexápodo deben ser consistentes entre sí, de forma que las rutinas no se contradigan o dificulten el funcionamiento del sistema. De esta forma, todos los robots deben estar de acuerdo sobre quién es el líder en cualquier momento.
RNF-04	Usabilidad	Media	El sistema debe tener un grado de intuición moderado, de forma que cualquier individuo con acceso al Manual de Usuario pueda hacer uso de él con relativa facilidad.
RNF-05	Transparencia	Alta	El sistema debe garantizar la comunicación transparente para poder cumplir con los requisitos de comunicación necesarios para poder denominarse un sistema distribuido.
RNF-06	Confiabilidad	Alta	Los componentes y bibliotecas del sistema serán verificados y actualizados regularmente para proteger contra vulnerabilidades conocidas.
RNF-07	Desempeño	Alta	El sistema debe ser capaz de funcionar con tiempos de respuesta cortos, que faciliten su interacción con el usuario.
RNF-08	Escalabilidad	Alta	El sistema debe ser capaz de manejar un número creciente de robots sin pérdida significativa de rendimiento.
RNF-09	Robustez	Alta	El sistema debe ser capaz de manejar fallos de robots y recuperarse de ellos.
RNF-10	Mantenibilidad	Alta	El código debe ser claro y bien documentado para facilitar el mantenimiento y las actualizaciones.
RNF-11	Tolerancia a fallos	Alta	El sistema debe seguir funcionando correctamente en caso de que algunos robots fallen.

4 Diagramas UML

4.1. Diagrama Físico del Sistema

El diagrama de físico muestra a nivel general los componentes de hardware involucrados en el sistema.

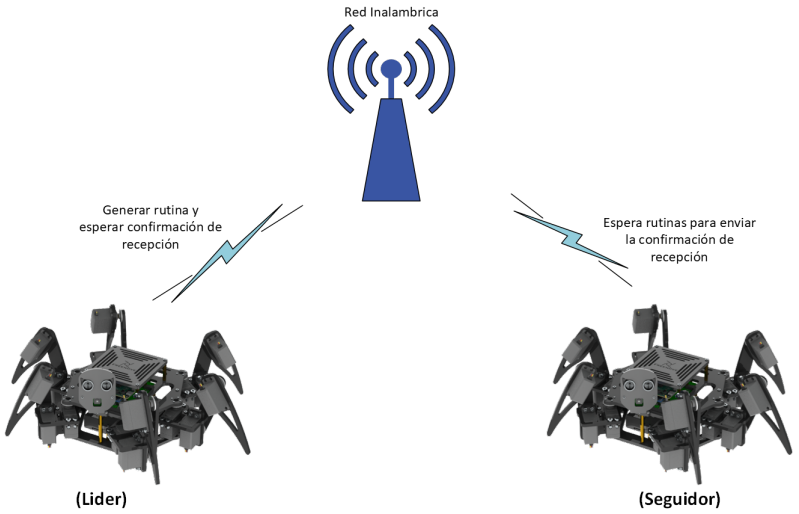

Figura 4.1: Diagrama físico del sistema.

4.2. Diagrama Casos de Uso

El diagrama de casos de uso muestra a nivel general el comportamiento que debe tener el software.

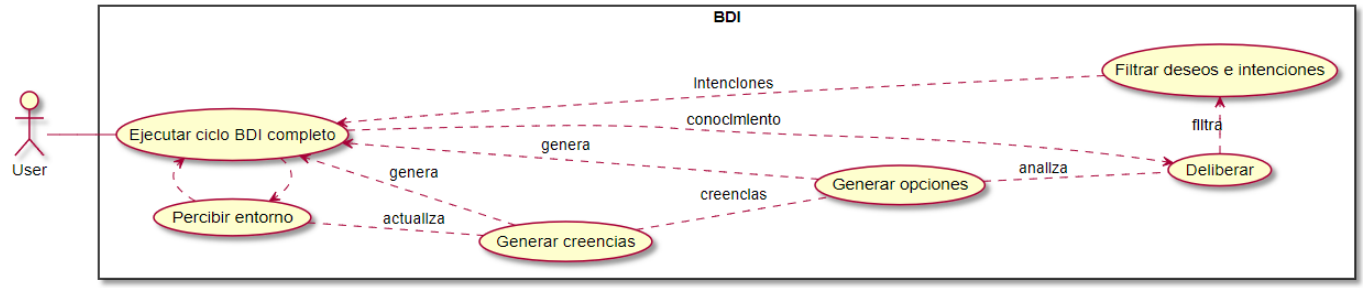

Figura 4.2: Diagrama de casos de uso

4.3. Diagrama de Clases

Representa un diagrama de clases simple para representar el enfoque de agentes que se implementó en el sistema, básicamente es una vista de la arquitectura BDI que se implementó.

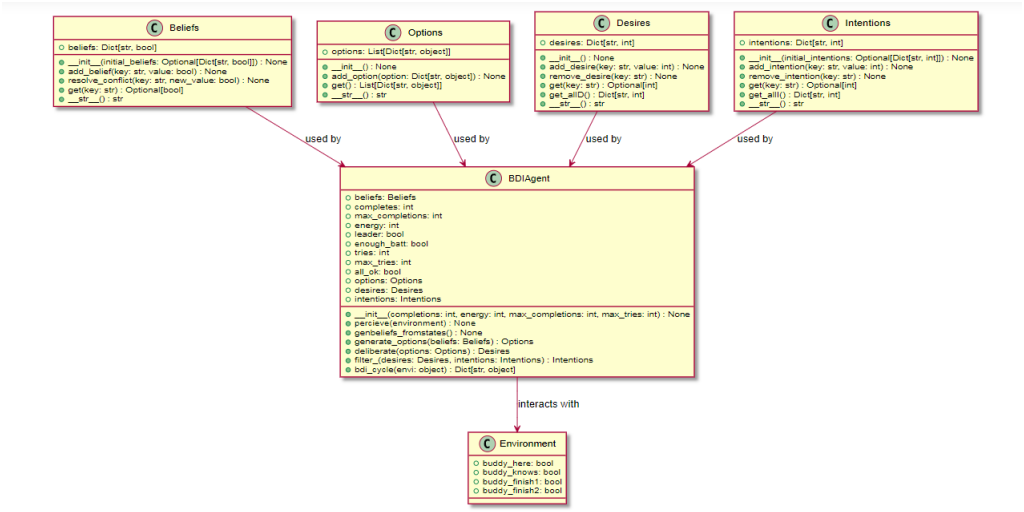

Figura 4.3: Diagrama de clases

4.4. Diagrama de secuencia

Aquí se muestra la interacción de los módulos, cuando se esta ejecutando el sistema de manera secuencial además de como el ciclo BDI se usara para cambiar el estado del ambiente y actualizar las creencias del agente con respecto a su conocimiento parcial.

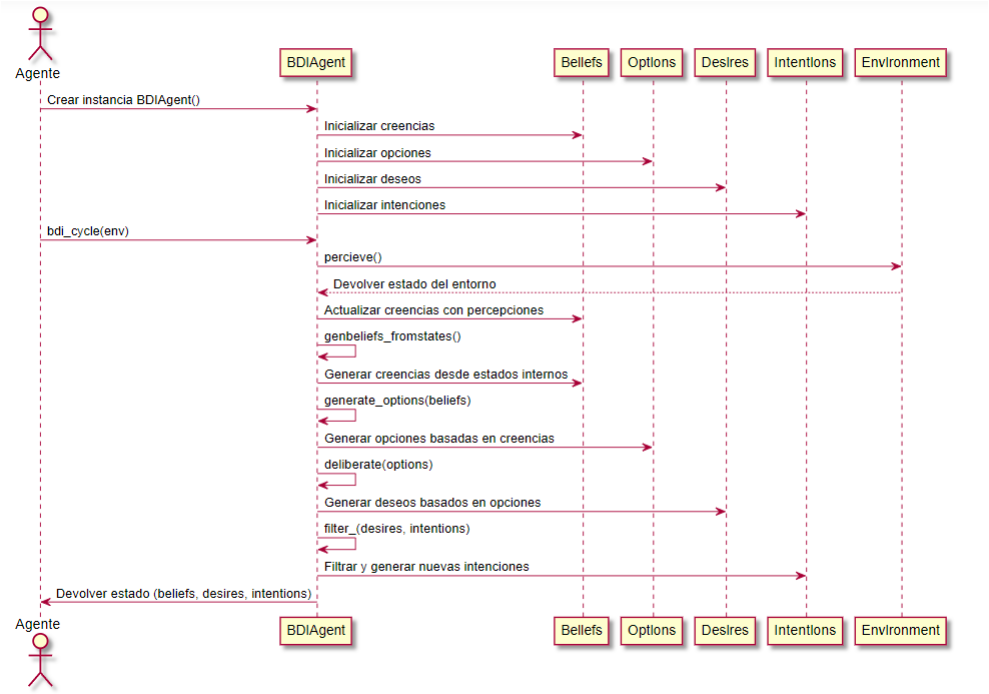

Figura 4.4: Diagrama de secuencia conexión de pagina web, usuario, Robot.

4.5. Diagrama de componentes y dependencias

Este diagrama muestra la relación entre los componentes del servidor, enfocándose en la comunicación a través de la red.

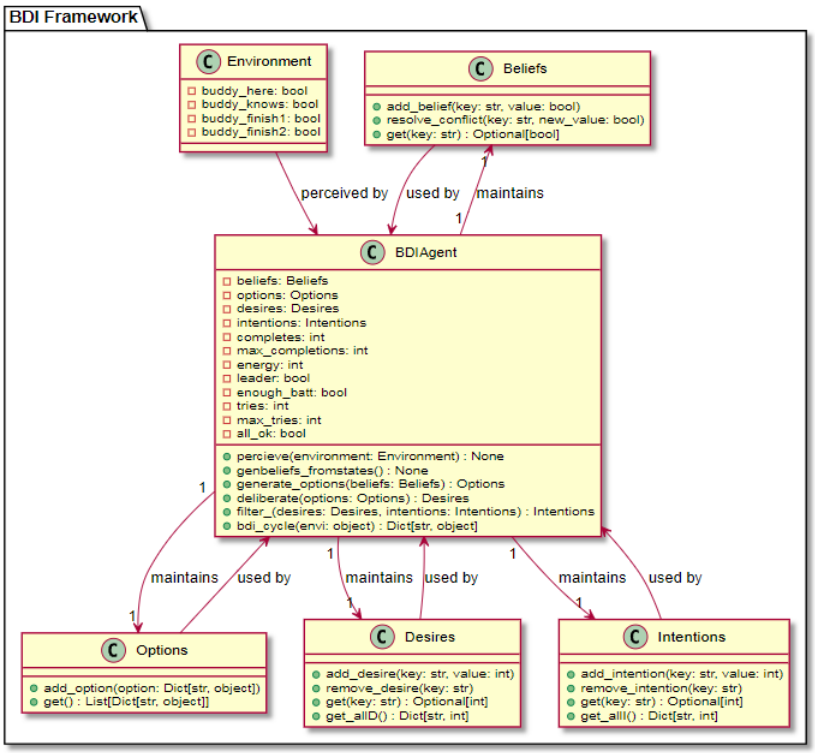

Figura 4.5: Diagrama de componentes

5 Diseño del Software (SDD)

5.1. Arquitectura del Sistema

La arquitectura de este sistema se fundamenta en un modelo BDI (Beliefs, Desires, Intentions), en el cual el agente utiliza sus creencias para filtrar sus deseos y deliberar sobre cuáles intenciones son más adecuadas para alcanzar sus objetivos. Esta estructura permite al agente tomar decisiones informadas y adaptativas en función de su estado interno y las condiciones del entorno.
Para trasladar esta lógica a los movimientos del Hexapodo, se emplea el software base de Freenove junto con las modificaciones desarrolladas en este proyecto. Estas adiciones permiten que el agente ejecute sus intenciones mediante rutinas de movimiento predefinidas, adaptando el comportamiento del Hexapodo de acuerdo con las decisiones tomadas por el modelo BDI.

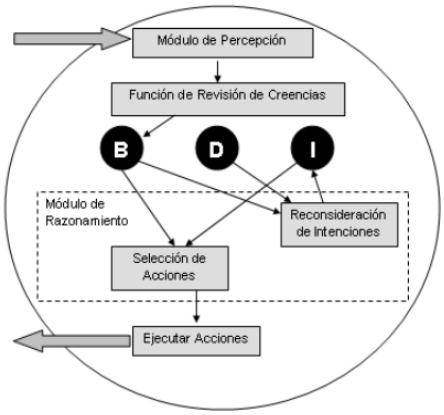

Figura 5.1: Plantilla de arquitectura BDI básica extraída de Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA) Departamento de Ciencias e Ingeniería de la Computación Universidad Nacional del Sur.*Planificación en agentes BDI*

5.1.1. Componentes Principales

- **Red:** Es el medio por el cual los mensajes se enviaran.
- **Entorno:** Proporciona la abstracción de un ambiente virtual en el cual os agentes pueden existir y interactuar.
- **Hexápodo:** Es el Robot de ejecutaría la rutina que creara el servidor.

- **BDI:** Es la estructura interna del agente que ayuda a utilizar los deseos, creencias y intensiones para deliberar sobre sus creencias y intensiones.

5.2. Diseño Detallado

5.2.1. Ejecución del sistema

El agente comienza con un conjunto de creencias iniciales y, al percibir su entorno, identifica la presencia de compañeros. Si detecta la presencia de un compañero, procede a iniciar el proceso de consenso para seleccionar un líder.
Una vez seleccionado el líder, este genera una rutina utilizando una cadena de Markov. La rutina generada se comparte con su compañero, y el líder espera la confirmación de recepción antes de proceder a ejecutar la rutina.

5.2.2. Flujos de Proceso

- Estado inicial
  - El agente inicia con sus creencias iniciales y pervive el entorno.
- Procesamiento de la percepción
  - Se actualiza el conocimiento sobre el entorno y se modifican las creencias con base en ese conocimiento.
  - Se filtran los deseos con la creencias actualizadas.
  - Se revisan las opciones de acción posibles y se determinan las intenciones que se pueden llevar a cabo.
- Ejecutar movimientos.
  - Si el Robot Hexápodo percibe la presencia de un compañero, se conecta con él e inicia el proceso de selección de líder.
  - Una vez determinado el líder, se procede a generar una rutina que será ejecutada.
  - La rutina generada se comparte con el agente seguidor, y se espera la confirmación de recepción.
  - A continuación, se ejecuta la primera mitad de la rutina, tras lo cual se espera la confirmación del compañero antes de finalizar el programa y reiniciar el ciclo BDI.

6 Algoritmos Usados

6.1. Algoritmo del Bully

El Algoritmo del Bully se basa en la idea de que el robot con el UID más alto debe ser el líder. El proceso se desarrolla en cuatro etapas, mismas que se describen a continuación.

1. Inicialización: Se conjuran los identificadores, asignando un UID a cada robot. Posteriormente se establece la comunicación, configurando la lista de pares ("peers") con los que cada robot puede comunicarse.

2. Detección de fallos: Cada robot monitoriza la presencia del líder mediante mensajes de latido (heartbeat), y si uno de ellos detecta que el líder ha fallado (es decir, no recibe mensajes de latido en un tiempo específico), inicia una elección.

3. Proceso de Elección: El robot que detecta el fallo envía mensajes de Election a todos los robots con UIDs mayores, posteriormente los robots que reciben el mensaje de Election responden con un mensaje de OK, indicando que participarán en la elección. Finalmente los robots que responden con OK inician su propia elección, enviando mensajes de Election a los robots con UIDs mayores que ellos.

4. Declaración del líder: El robot con el UID más alto que no recibe ninguna respuesta de OK se declara líder, dicho o líder envía un mensaje de Victory a todos los robots notificando su liderazgo y los robots actualizan su estado para reconocerlo.

A continuación se presentan los cálculos matemáticos para la implementación de este algoritmo en la asignación de los UIDs, seguido de un ejemplo. Posteriormente, se describe el proceso para obtener la colisión deseada, que es de 0.5, así como la aproximación de Stirling para los factoriales.
Finalmente se muestra la utilización de un resolvedor automático para obtener k a partir de las probabilidades de no colisión y de la cantidad de dispositivos.

k : UIDs posibles
n : número de dispositivos
Combinaciones posibles sin colisión:

$$Q_{nc} = kPn = \frac{k!}{(k-n)!}$$

en este contexto, P es la permutación
Combinaciones totales

$$Q_t = k^n$$

Probabilidad de no colisión

$$P(\text{no colisión}) = \frac{Q_{nc}}{Q_t} = \frac{kPn}{k^n}$$

La probabilidad de que al menos haya una colisión es

$$P(\text{colisión}) = 1 - P(\text{no colisión}) = 1 - \frac{kPn}{k^n}$$

Figura 6.1: Cálculos matemáticos para la asignación de los UDIs

Si tenemos
$k = 10$ UIDs
$n = 2$ dispositivos
La posibilidad de que no haya colisión al elegir el mismo UID es:

$$P(\text{no colisión}) = \frac{10P2}{10^2} = \frac{90}{100} = 0.9$$

La posibilidad de elegir el mismo UID es:

$$P(\text{colisión}) = 1 - 0.9 = 0.1$$

Figura 6.2: Ejemplo de cálculos matemáticos para la asignación de los UDIs

$$P(\text{no colisión}) = \frac{k!}{k^n(k-n)!} = 0.95$$

Figura 6.3: Fórmula para obtener la colisión deseada

$$k! \approx \sqrt{2\pi k}\left(\frac{k}{e}\right)^k$$

$$(k-n)! \approx \sqrt{2\pi(k-n)}\left(\frac{k-n}{e}\right)^{(k-n)}$$

Figura 6.4: Aproximación de Stirling

$$P(\text{no colisión}) \approx \frac{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k}{\sqrt{2\pi(k-n)}\left(\frac{k-n}{e}\right)^{(k-n)}}$$

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}}\frac{\left(\frac{k}{e}\right)^k}{\left(\frac{k-n}{e}\right)^{(k-n)} \cdot k^n}$$

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}}\left(\frac{e}{k-n}\right)^{(k-n)}\left(\frac{k}{e}\right)^k \frac{1}{k^n}$$

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}}\left(\frac{e}{k-n} \cdot \frac{k}{e}\right)^{(k-n)}\left(\frac{k}{e}\right)^n \frac{1}{k^n}$$

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}}\left(\frac{k}{k-n}\right)^{(k-n)}\left(\frac{k}{e}\right)^n \frac{1}{k^n}$$

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}}\left(\frac{k}{k-n}\right)^{(k-n)}\left(\frac{1}{e}\right)^n$$

Figura 6.5: Aplicación de la Aproximación de Stirling a las Probabilidades

$$P(\text{no colisión}) \approx \sqrt{\frac{k}{k-n}} \left(\frac{k}{k-n} \right)^{(k-n)} e^{-n}$$

$$P(\text{no colisión}) \cdot e^{n} \approx \sqrt{\frac{k}{k-n}} \left(\frac{k}{k-n} \right)^{(k-n)}$$

$$(P(\text{no colisión}) \cdot e^{n})^2 \approx \frac{k}{k-n} \left(\frac{k}{k-n} \right)^{2(k-n)}$$

$$\ln((P(\text{no colisión}) \cdot e^{n})^2) = \ln \left(\frac{k}{k-n} \left(\frac{k}{k-n} \right)^{2(k-n)} \right)$$

$$\ln(P(\text{no colisión})^2) + 2n = \ln \left(\frac{k}{k-n} \right) + 2(k-n) \ln \left(\frac{k}{k-n} \right)$$

$$\ln((0.95)^2) + 2n = \ln \left(\frac{k}{k-n} \right) + 2(k-n) \ln \left(\frac{k}{k-n} \right)$$

$$\ln((0.95)^2) + 2n = (2(k-n)+1) \ln \left(\frac{k}{k-n} \right)$$

Figura 6.6: Resolvedor automático para la obtención de k

7 Código Implementado

El código sigue la secuencia descrita en la figura 2.1. Primero, establece la comunicación entre los agentes y, a continuación, se determina quién será el líder. Una vez seleccionado el líder, este crea una rutina de bailes y la comparte con su compañero. Luego, el sistema espera la confirmación de recepción antes de iniciar la ejecución de las rutinas. Durante la ejecución, se incrementa un contador que registra el número de rutinas completadas, hasta alcanzar un total de tres rutinas o hasta que se detecte que no hay suficiente energía para continuar con el objetivo.

7.1. Ciclo BDI

El ciclo BDI implementado ayuda a que el agente tengas autonomía para deliberar si es posible continuar el proceso además de poder dar un enfoque de alto nivel a la abstracción de los problemas con los que podría lidiar el agente para cumplir con su objetivo o deliberar que no es posible hacerla.

7.2. Elección de líder

A continuación, las figuras 7.1 y 7.2 muestran el proceso para la elección del robot o agente líder.

7.3. Creación De La Rutina Para El Robot Hexapodo

La creación de la rutina se basa en un proceso de Markov, modelado mediante una cadena que seleccionará los bailes creados por nosotros. Este proceso es llevado a cabo exclusivamente por el agente líder (figura 7.1), mientras que los agentes seguidores se limitan a ejecutar dichas rutinas para alcanzar la coordinación en los movimientos, utilizando un ciclo BDI para modelar su comportamiento.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
import seaborn as sns

def equation(k, P, n):
    left_side = np.log(P**2) + 2 * n
    right_side = (2 * (k - n) + 1) * np.log(k / (k - n))
    return left_side - right_side

# Valores iniciales
P_values = np.arange(0.35, 1.05, 0.1)
inicio = 2
final = 21
step = 1
```

```python
# DataFrame para almacenar los resultados
results = pd.DataFrame()

# Iterar sobre los valores de P
for P in P_values:
    k_solutions = []
    n_values = np.arange(inicio, final, step)  # Valores de n desde 1 hasta 20
    for n in n_values:
        # Resolver la ecuación para cada n
        initial_guess = n + 1
        k_solution = fsolve(equation, initial_guess, args=(P, n))[0]
        k_solutions.append(k_solution)

    # Agregar los resultados al DataFrame
    results[f'P={P:.2f}'] = k_solutions

# Agregar los valores de n al DataFrame
results['n'] = n_values

# Configurar la gráfica
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
for P in P_values:
    plt.plot(results['n'], results[f'P={P:.2f}'], label=f'P={P:.2f}')

plt.xticks(np.arange(inicio, final, step))

plt.xlabel('Número de dispositivos')
plt.ylabel('UIDs recomendados')
plt.title('UIDs recomendados vs Número de dispositivos para diferentes valores de $P$')
plt.legend()
```

```python
plt.grid(True)
plt.show()
```

Figura 7.1: Código para la generación del gráfico que muestra la relación entre el número de dispositivos y los UIDs recomendados

7.4. Ejecución De La Rutina

Al emplear la plataforma Raspberry, los robots tienen la capacidad de utilizar módulos diseñados para microcontroladores, permitiendo el uso de señales PWM para controlar servomotores. En nuestro caso, se trata de servomotores con un rango de movimiento de 180 grados. El software base de Freenove está diseñado para que todos los servomotores tomen un punto cero como referencia, el cual se utiliza para medir los ángulos a los que se desplazan las extremidades, garantizando un conocimiento global de la posición de cada una de ellas. Después nosotros para lograr una variedad de movimiento mas amplia requerimos conocer las posiciones individuales de cada servo y así poder mostrar generar un movimiento mas presido.
Para esto nosotros movemos cada servo individualmente para combinaciones de posición de extremidades no establecidas en el código base por lo que la clase *Bailes* es una modulo que modifica el uso de los servomotores para adaptar su movimiento a nuestra conveniencia y así lograr un comportamiento menos rígido des robot.
Por lo que cada Agente tiene que ejecutar las funciones heredades de bailes para ejecutar su rutina, para que este modulo logre controlar los servomotores de la manera que permita ejecutar el control de estos y se ejecuten los movimientos que están establecidos en la rutina como muestra la figura 7.2.

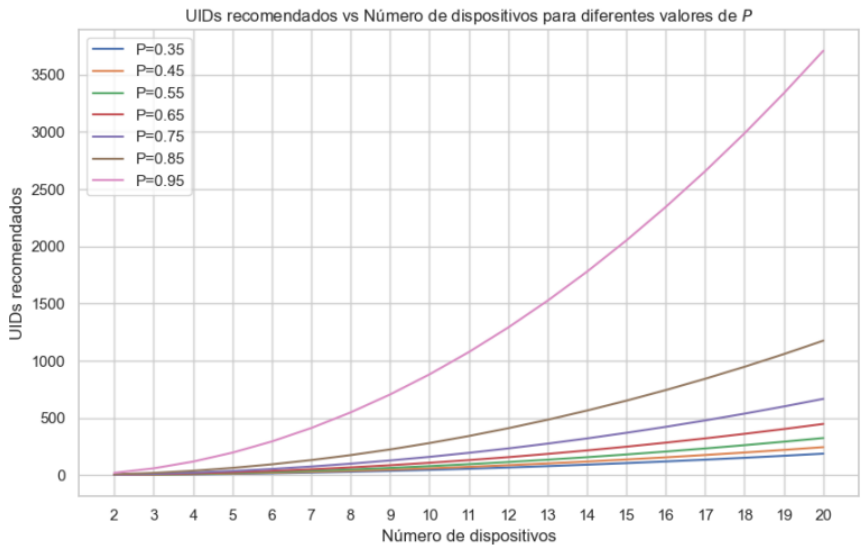

Figura 7.2: Relación entre el número de dispositivos y los UIDs recomendados

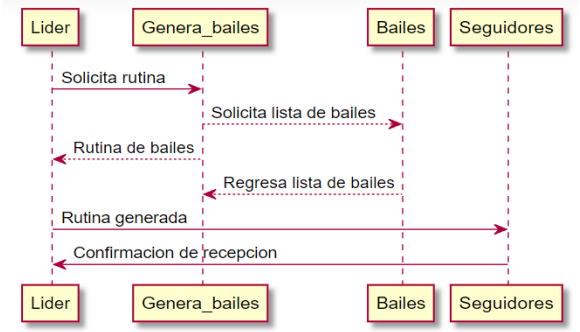

Figura 7.3: Diagrama de secuencia de la elaboración de las rutinas

7.5. Finalización del sistema

El sistema está diseñado para detener el ciclo de ejecución después de completar tres rutinas, o si no detecta la presencia de un compañero, dado que su objetivo principal es coordinar sus movimientos con otro agente similar. Si percibe la ausencia de un compañero o considera que ya no es posible alcanzar su objetivo, abandonará la tarea y finalizará el programa.

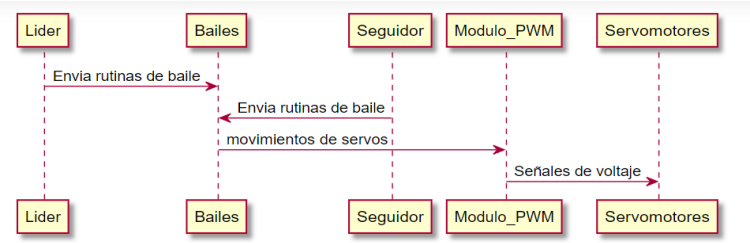

Figura 7.4: Diagrama de secuencia de ejecución de rutinas

Bibliografía

[1] Wiebe Van der Hoek and Michael Wooldridge. Multi-agent systems. *Foundations of Artificial Intelligence*, 3:1–40, 2002.