UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Validación de un algoritmo de inteligencia de enjambre enfocado en sincronización y control de formaciones de sistemas robóticos multi-agente en un entorno físico

Protocolo de trabajo de graduación presentado por José Alejandro Rodríguez Porras, estudiante de Ingeniería Mecatrónica

Guatemala,

2023

Resumen

El proyecto que se propone consiste en la continuación de la línea de investigación de desarrollo de robótica de enjambre, específicamente con enfoque en el desarrollo y aplicación del algoritmo de sincronización y control de formaciones de sistemas robóticos multi-agente. Se trata de una continuación debido a que en la fase previa se llegó hasta pruebas del algoritmo desarrollado en simulaciones, y ahora se cuenta con la infraestructura necesaria para las pruebas en un ambiente controlado con agentes físicos.

El objetivo principal es validar el algoritmo previamente mencionado en sistemas físicos y evaluar su funcionamiento en el ecosistema Robotat. Para poder cumplir con el mismo, se utilizarán las herramientas principales que la universidad proporciona en la actualidad como apoyo, las cuales son: el sistema de captura de movimiento OptiTrack, los robots diferenciales Pololu 3Pi+ y la red de comunicación existente en el Robotat. Además, se continuará usando el software de Matlab y Webots, para ajustes o variantes del algoritmo de ser necesario.

La validación consistirá en realizar diversos experimentos para poner a prueba el algoritmo en el ambiente controlado. Como factores principales se tomará en cuenta el desempeño individual del agente, la generación de las trayectorias, el posicionamiento inicial, intermedio y final de los agentes, las configuraciones de formación, diversos escenarios de interés con obstáculos, etc. En síntesis, se examinará la ejecución del algoritmo para observar su desempeño y encontrar puntos de mejora o posibles aplicaciones, según los escenarios propuestos.

Con la validación de este algoritmo se tendrá un buen cimiento para la continuación con la creciente tecnología de la robótica de enjambre. Observar la ejecución de las trayectorias generadas por el algoritmo en un entorno controlado representará un paso significativo para el campo, abrirá las puertas a nuevas aplicaciones y un mayor potencial de este tema en el mundo moderno.

Antecedentes

Robótica de enjambre

Conforme ha avanzado la tecnología en el área de la ingeniería, el área de robótica ha ido creciendo tanto en capacidad como complejidad, lo que se puede observar en proyectos como el Atlas de Boston Dynamics [1], el cual presenta grandes capacidades de movilidad, pero que también puede llegar a ser muy costoso por las mismas razones. Habiendo mencionado sus fortalezas, cabe resaltar que este trabaja de forma individual, lo que resulta siendo una limitación para distintas aplicaciones que involucran tareas que necesitan más de un individuo, como es el caso de búsquedas de rescate, explorar un área, o incluso realizar varias tareas de forma paralela. Es en estos casos donde entra el campo de investigación de la robótica de enjambre, el cual, contrario a Atlas se basa en robótica "simple".

El área de robótica de enjambre consiste en realizar alguna tarea compleja a gran escala usando múltiples robots de construcción y capacidad simple. Estos robots suelen ser de bajo costo y consumo, de dimensiones relativamente pequeñas y su funcionamiento a nivel individual no es demasiado exigente en términos de procesamiento, pero al momento de

trabajar como conjunto estos pueden desempeñar tareas a niveles de alta complejidad [2], teniendo así una operación multiagente. Esta área de la robótica está inspirada en sistemas en la naturaleza compuestos por múltiples individuos, como es el caso de insectos sociales como las colmenas de abejas y colonias de hormigas, bancos de peces, etc. Entre las principales ventajas de la robótica de enjambre se destacan la robustez, flexibilidad y escalabilidad. A grandes rasgos esta disciplina puede tomar modelos de la naturaleza y adaptarlos al ámbito de control para los robots, o bien, se pueden crear nuevos algoritmos [3].

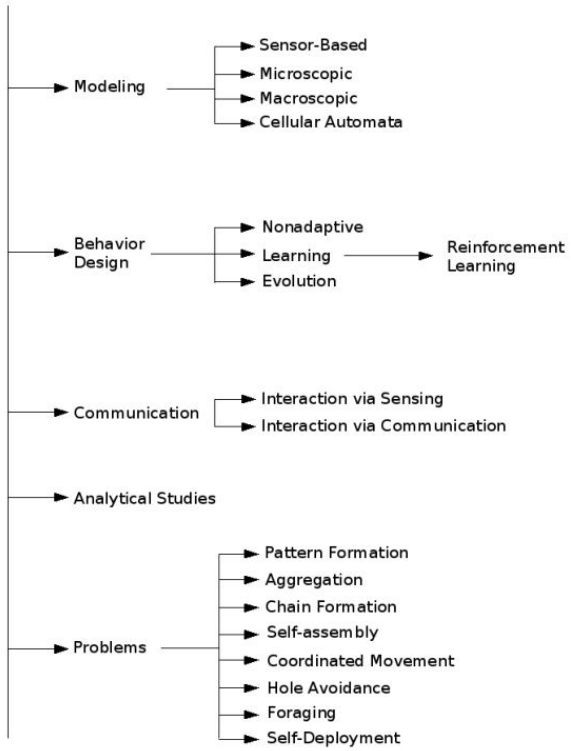

Figura 1: Taxonomía de la robótica de enjambre [3].

En el diagrama de la Figura 1 se muestran los principales ejes de estudio de la robótica de enjambre, los cuales tienen ramificaciones según su aplicación.

Kilobots

En 2014 la Universidad de Harvard (Instituto de Wyss) desarrolló estudios con enjambres de hasta 1024 agentes, los cuales se denominan Kilobots (Figura 2), cuyas dimensiones son muy cercanas a las de un centavo estadounidense y un costo menor a los 20 dólares. El enjambre que formó parte del estudio fue de 1024 robots con lo que se logró realizar distintas formaciones bidimensionales tanto artificiales como imitaciones de la naturaleza en entre 2 y 12 horas según la complejidad de la figura. Estos robots tienen la capacidad de comunicarse entre sí de forma local a través de señales de luz infrarroja y descentralizada, con la característica de tener una alta robustez.

El principio de movimiento de Kilobot se basa en vibraciones, estos poseen tres patas y

según a donde se necesite mover estos vibran de una manera específica para moverse en esa dirección. La meta principal detrás del desarrollo de estos robots es desarrollar agentes cada vez más pequeños para que estos se puedan ensamblar en objetos como herramientas o bien mover objetos en conjunto [4].

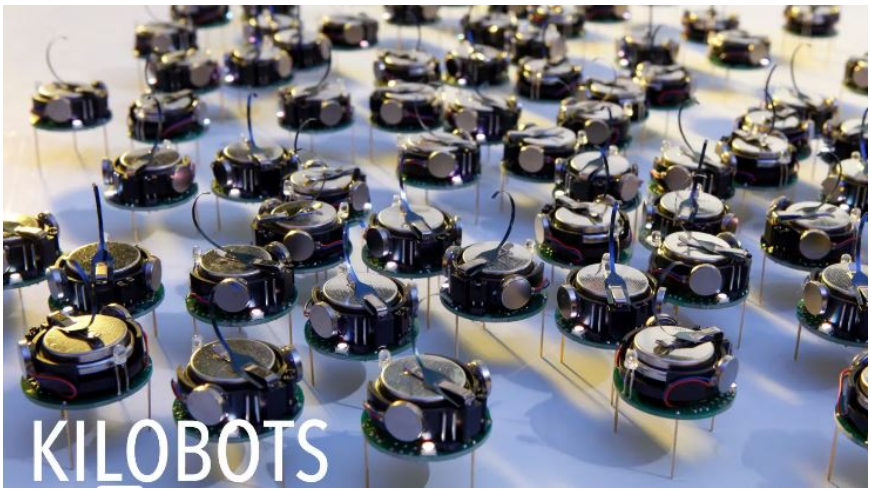

Figura 2: Kilobots [5].

WsBot

En 2019 se desarrolló el WsBot, con el objetivo principal de servir como un robot de bajo costo para aplicaciones de enjambre inteligente en la industria, principalmente pero no de forma exclusiva para fábricas inteligentes. El WsBot posee atributos similares a los de máquinas industriales, como máquinas elevadoras (forklifts), para lograr comportamientos inteligentes ya sea con tareas individuales pequeñas para lograr una mayor o trabajar en conjunto con los demás agentes para completar un trabajo global. La estructura de este agente es la de un robot diferencial basado en un ROS (Sistema Robótico Operativo), capaz de comunicarse con los demás agentes a través de Wi-Fi [6].

En la Figura 3 se puede apreciar con más detalle el Wsbot y en la Figura 4 se muestra este en un ambiente de aplicación. Con la aparición de esta plataforma robot se busca posicionar esta tecnología en un área de mayor acceso a la población y aplicaciones de uso más cotidiano.

Robotat

En la Universidad del Valle de Guatemala se realizó el megaproyecto de múltiples fases, Robotat, en las cuales tanto catedráticos como alumnos colaboraron en su desarrollo. El Robotat es un ecosistema de experimentación robótico, diseñado para que este sirva como un campo de pruebas para que los estudiantes puedan estudiar a los robots con herramientas avanzadas de las cuales destacan el sistema de captura de movimiento OptiTrack y la red de comunicación Wi-Fi.

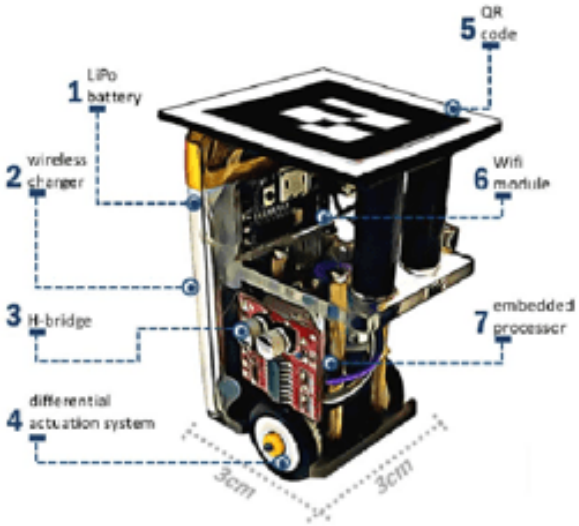

Figura 3: WsBot [6].

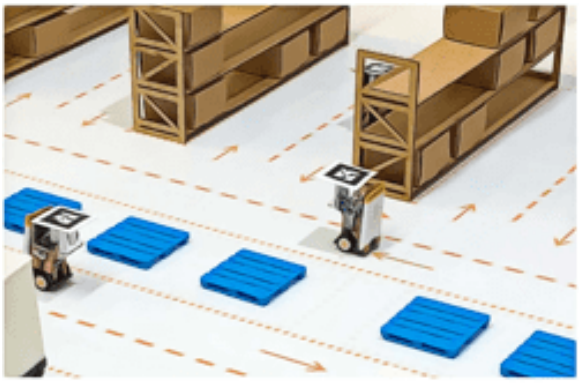

Figura 4: WsBot en ambiente de trabajo [6].

En la última tésis, de Camilo Perafán Montoya en 2021 [7], se completó la red de comunicación para múltiples agentes, expansión que permite la implementación de nuevas tecnologías que en las iteraciones previas no se había logrado, entre las cuales está principalmente la robótica de enjambre. Además se desarrolló una librería en el lenguaje de programación C para que un microcontrolador ESP32 se pudiera conectar a la red implementada, y recibir datos del OptiTrack por medio del protocolo MQTT, así como enviar datos por el mismo hacia un servidor. Es importante recalcar que una de las conclusiones principales de este proyecto fue que el número máximo de agentes que se pueden conectar en simultáneo a la red para que su decodificación de datos se mantenga eficiente es de 11 agentes. Para que agentes no robóticos puedan interactuar con el sistema se diseñó una antena inteligente.

Algoritmos de Robótica de Enjambre implementados en UVG

Particle Swarm Optimization (PSO)

Este algoritmo consiste en que un conjunto de partículas comiencen desde una posición aleatoria y arbitraria para luego reunirse en conjunto en un punto específico. En la UVG, desde 2019 se ha trabajado con este algoritmo desde una perspectiva modificada, es decir un Modified PSO (MPSO), para dejar de estudiar a los agentes como partículas solamente, sino tomar en cuenta sus características al tratarse de robots diferenciales.

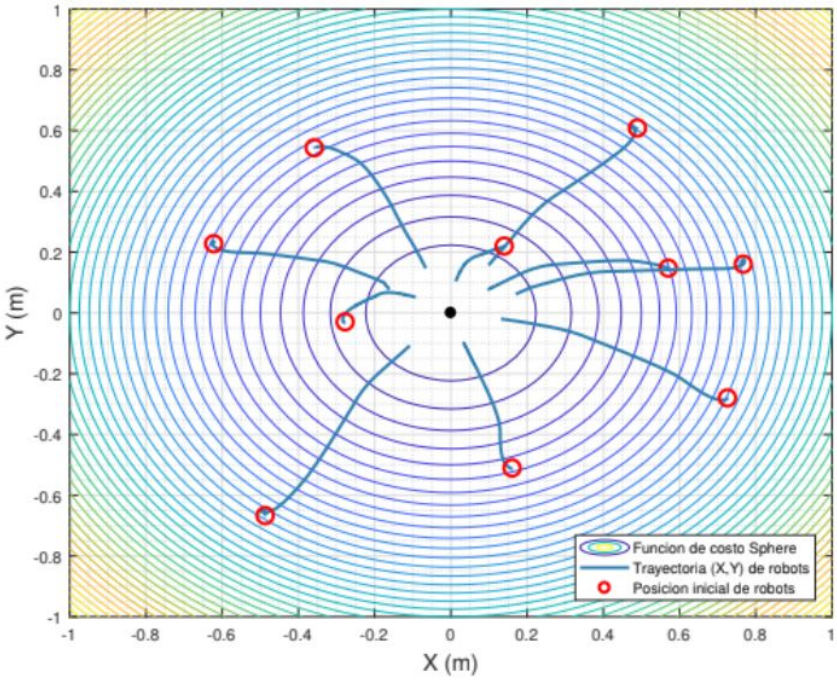

Figura 5: Trayectorias generadas de PSO para un enjambre de 10 agentes [8].

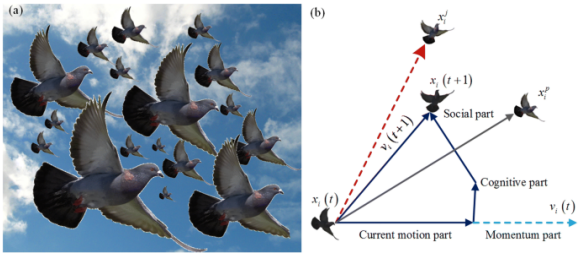

Figura 6: PSO en la naturaleza [9].

Primero se estudió en la tésis de Aldo Aguilar [8], donde se avanzó hasta simulaciones de Matlab y Webots. Luego, en años siguientes se continuó con el desarrollo de estos algoritmos para refinarlos hasta llegar a intentos de implementación en físico, como es el caso del trabajo de Alex Maas [10] usando una plataforma Raspberry Pi para obtener las poses en tiempo real y la comunicación entre robots por medio de UDP, aunque cabe resaltar que existieron limitaciones relacionadas a la falta de de una plataforma móvil por lo que la validación se

tuvo que realizar de forma que se movieran los marcadores manualmente, haciendo varias capturas para emular el movimiento que hubiera presentado un robot móvil, la verificación de las poses/posiciones se hizo mediante un algoritmo de visión de computadora.

Los últimos avances de la implementación de este algoritmo se dieron en 2022, con la tésis de Rubén Lima [11], al comenzar con la validación del algoritmo en físico usando el equipo de captura de movimiento del Robotat, continuando con el diseño de una plataforma móvil llamada ByteBot3D compuesta por una Raspberry Pi. Se incursionó con la comunicación TCP/IP. Debido a que no se usaron encoders el rendimiento del comportamiento mecánico no fue el esperado por lo que se optó por mover la plataforma manualmente.

Ant Colony Optimization

El algoritmo se basa en el comportamiento que tienen las colonias de hormigas para encontrar rutas hacia comida mediante feromonas, cada hormiga (agente) explora el terreno y si una encuentra comida (meta) va marcando el camino con lo que lo encontró con feromonas (probabilidad) y las demás hormigas siguen este camino eventualmente. Si por el contrario una hormiga no encuentra alimento en su exploración, su rastro se desvanece y termina siguiendo el rastro de feromonas de otra hormiga.

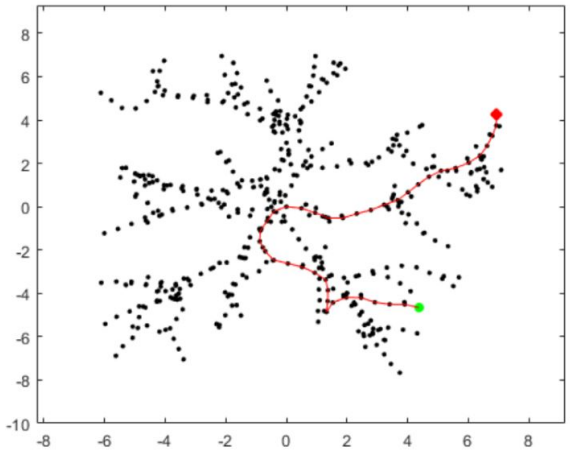

Figura 7: Diagrama de Feromona y camino encontrado [12].

En la tésis de Gabriela Iriarte [12] se desarrolló un algoritmo de *Ant Colony* en UVG por primera vez y en otra iteración Daniela Baldizón [13] realizó una versión modificada, *Modified Ant Colony Optimization (ACO)*, para adaptarlo a exploración de terrenos, planificación de trayectorias, reconocer y evadir obstáculos. Se implementó en Matlab a nivel de simulación con validación usando 3 mapas distintos para probar el algoritmo.

El trabajo realizado por Walter Sierra [14] continuó con el algoritmo de Ant System (AS), implementado previamente en la universidad solamente a nivel de simulación. En este caso, se expandió la implementación a una plataforma Raspberry Pi para probar su funcionamiento en un ambiente físico. Debido a que no se contaba con una plataforma móvil se realizaron las pruebas de forma manual analizando mediante marcadores en una mesa de pruebas y visión de computadora.

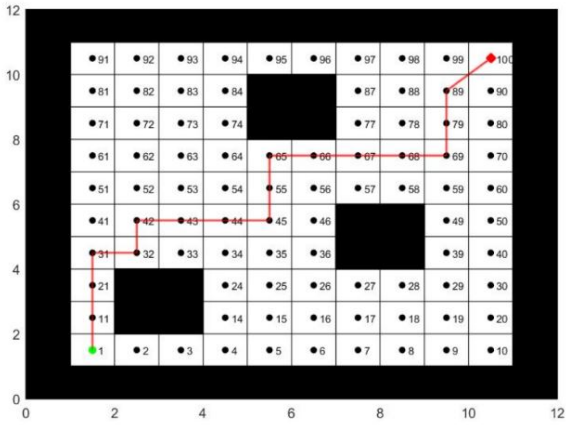

Figura 8: Trayectoria del camino encontrado con feromona [13].

Algoritmo de sincronización y control de sistemas de robots multi-agente para misiones de búsqueda

Este algoritmo fue desarrollado e implementado por Andrea Maybell Peña [15], centrado en manejar el sistema de robots multi-agente para misiones de búsqueda, apoyándose de teoría de grafos y control moderno para mantener a los agentes en formaciones específicas, generando una especie de cuerpo con los robots diferenciales como sus partes, con la capacidad de "fluir" a través de obstáculos. Se crearon subalgoritmos para mantener la formación y un control para la evasión de obstáculos.

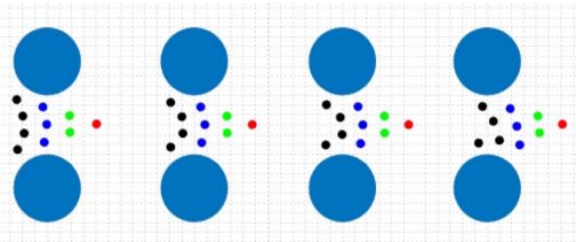

Figura 9: Movimiento en formación mínimamente rígida a través de obstáculos [16].

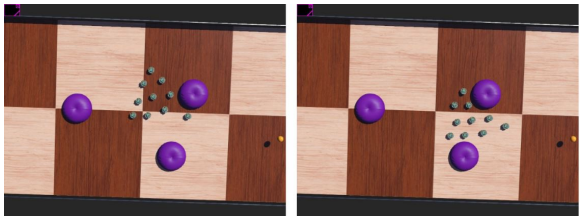

Figura 10: Simulación en Webots de la formación moviéndose a través de obstáculos [16].

Se avanzó hasta simulaciones en Matlab modelando al agente como partícula, para luego realizar simulaciones en Webots, adaptando al agente a un modelo de un robot diferencial. La plataforma robótica elegida en ese momento para la tarea fue un BitBot, mas nunca se llego a probar en dicho robot.

En 2022, se volvió a estudiar este algoritmo de formaciones con un enfoque 3D, en la tésis

de Kenneth Aldana[17]. Se implementó el algoritmo en simulaciones de Matlab y Webots solamente, no se llegó a implementar en físico tampoco.

Plataformas Robóticas Móviles en UVG

A lo largo de los últimos años se ha buscado desarrollar plataformas móviles adecuadas para la robótica de enjambre en el ambiente Robotat de la UVG. Entre los primeros prospectos se mencionaron los Bitbots [15], y también se mostró interés en los Epucks. En los años más recientes se han desarrollado varios robots diferenciales como lo son:

- ByteBot: Implementado por Julio Rodríguez [18], consiste de una Rasberry Pi Zero con Servomotor, sensores de proximidad y cámara; cuenta con comunicación inalámbrica TCP IP y UDP.
- Alphabot: Es un Alphabot con modificaciones que usa una Raspberry Pi, se implementó en el Robotat con sistema OptiTrack. Rediseño por Luis Nij [19].
- ByteBot3D: Es una continuación del trabajo realizado en el ByteBot, modificado por Rubén Lima [11], utiliza el mismo controlador y comunicación que la versión anterior, usa el sistema de captura de movimiento del Robotat.

Justificación

La inteligencia de enjambre es un campo relativamente nuevo en el área de inteligencia artificial, con potencial en diversas aplicaciones que van desde el contexto de la vida cotidiana hasta un ambiente industrial. La mayor ventaja que presenta es que se compone de múltiples agentes para llevar a cabo tareas complicadas y se forma de unidades simples de bajo costo. Con este atributo, los sistemas robóticos basados en la inteligencia de enjambre no son vulnerables en caso de que uno de los agentes falle o se pierda, lo que les da robustez. Esta misma ventaja resulta útil en misiones de búsqueda y rescate, así como la movilización de distintos tipos de agentes (entre ellos terrestres, acuáticos y aéreos) según la aplicación lo requiera.

Se ha elegido estudiar la robótica de enjambre debido a que esta resulta ideal para poner en práctica los conocimientos adquiridos a lo largo de la carrera de Ingeniería Mecatrónica, principalmente los conceptos avanzados sobre robótica y control adquiridos en los últimos años de estudio.

Desde 2019 en la Universidad del Valle de Guatemala se comenzó con el estudio y mejora de algoritmos de inteligencia de enjambre, de entre los cuales se destaca el algoritmo de sincronización y control de formaciones en sistemas robóticos multiagente, cuya alta coordinación y flexibilidad representa una alta eficiencia al momento recorrer áreas estratégicamente. Esto se puede aplicar a la búsqueda de sobrevivientes para catástrofes naturales, dada la versatilidad del algoritmo de control de los robots.

Habiendo mencionado los avances previos en la UVG, cabe resaltar que todos los avances que se han hecho han sido a nivel de simulación en software como Matlab y Webots, por lo

que realmente no se ha tenido una validación formal física de la mayoría de los algoritmos en robots móviles funcionales, entre ellos el algoritmo de sincronización y control de formaciones desarrollado por Maybell Peña [15]. Esta falta de validación se ha debido a que no había infraestructura adecuada, pero ahora sí existe y el algoritmo está listo para realizar pruebas en el Robotat, con el sistema OptiTrack y los robots diferenciales con ESP32. Con la oportunidad que estos recursos presentan, se puede validar el algoritmo en un ambiente físico, lo que sería un gran paso para la investigación de la inteligencia de enjambre. Finalmente, esto dejaría un precedente con buena base para futuro estudio del tema, haciendo cada vez más factible su uso en el mundo moderno.

Objetivos

Objetivo General

Validar el algoritmo de inteligencia de enjambre enfocado en sincronización y control de formaciones de sistemas robóticos multi-agente previamente desarrollado a nivel de simulación, en sistemas físicos, y evaluar su funcionamiento en el ecosistema Robotat.

Objetivos Específicos

- Adaptar el software desarrollado anteriormente para su aplicación en la mesa de pruebas del Robotat y los robots diferenciales Pololu 3Pi+.
- Evaluar la generación de trayectorias usando marcadores en el Robotat y el sistema de captura de movimiento OptiTrack.
- Examinar el comportamiento de los robots diferenciales al ejecutar las trayectorias generadas por el algoritmo de sincronización y control, en distintos escenarios.
- Comparar los resultados obtenidos en simulaciones con los obtenidos con los agentes físicos en ambientes controlados.

Marco teórico

Es importante entender el contexto del algoritmo de sincronización y control de formaciones multi-agente, para lograr aplicarlo correctamente a un entorno real. Es por esto que en las siguientes secciones se desarrollará un poco más sobre los conceptos claves detrás del algoritmo que se compone de teoría de grafos principalmente, funciones y herramientas matemáticas de interés, sistemas de control, funcionamiento, resultados previos de simulación, el software y hardware relevante, la infraestructura disponible, el sistema de captura de movimiento, etc.

Definiciones importantes de robótica de enjambre

Agente

Un agente es un individuo simple que forma parte del enjambre, en este caso un robot de dimensiones relativamente pequeñas que posee la capacidad de realizar acciones simples para cumplir un objetivo más complejo de forma cooperativa; recibir y transmitir información, así como instrucciones [3].

Formaciones

En los enjambres de la naturaleza comúnmente se puede notar la emergencia de patrones entre los agentes, creando formaciones y comportamientos aparentemente coordinados, ya sea a propósito o no. Esto no es diferente en la robótica de enjambre, pues también se pueden realizar formaciones simples o complejas según la aplicación lo requiera. Se pueden crear movimiento coordinados mediante la generación de trayectorias con evasión de obstáculos y control de velocidad, con restricciones de distancia entre agentes, se elaborará sobre esto más adelante. Existen dos tipos principales para la coordinación de las formaciones, los cuales son control centralizado y descentralizado [3].

Control centralizado

Este enfoque no esta construido alrededor de comunicación entre agentes por medio de sensores, sino que alrededor de un sistema de transmisión de datos a un mismo receptor (un CPU que se encarga de coordinación). Esto se debe a que agentes robóticos por lo general solo reportan su posición a un sistema de cómputo principal y este le devuelve instrucciones al robot, sin necesidad de que este se tenga que comunicar con los demás agentes de forma directa. La formación de patrones descentralizada por lo general es más costosa debido a la necesidad de un CPU de control y procesamiento, también es menos escalable y con menor robustez a fallas [20]. Cabe resaltar que la adaptación a nivel grupal solo se puede obtener mediante un control centralizado o permitiendo que los agentes comuniquen sus información del exterior entre sí [3].

Control descentralizado

Este tipo de control consiste en que los agentes realicen mediciones individuales de su entorno y se comuniquen entre sí, sin necesidad de un CPU externo que les otorgue instrucciones. Esto implica que el programa y procesamiento debe ocurrir completamente en los agentes, según la información que obtengan de sus agentes cercanos. Este enfoque, al no poseer una unidad central de procesamiento, resulta ser más barato y menos propenso a fallas o pérdidas, pues cada uno de los robots tiene el mismo peso de información para la formación [3].

Teoría de grafos

Es una rama de la matemática que se centra en el estudio de los grafos, los cuales son una especie de mapas de ruta, que se dibujan con puntos y líneas. Por lo general los grafos son herramientas útiles en el modelado de problemas, para representar las relaciones de los componentes importantes y sirven para resolver problemas de búsqueda de caminos eficientes, entre otros. Esta teoría emplea estructuras matriciales para representar a los grafos y realizar operaciones con los mismos [21].

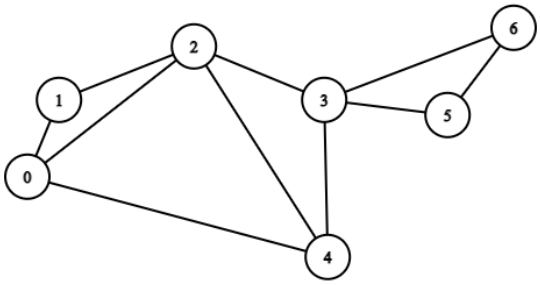

Figura 11: Ejemplo de un grafo.

Conceptos básicos en teoría de grafos

Vértices: También llamados nodos, son los puntos que conforman al grafo. Cada vértice tiene una valencia asociada según la cantidad de aristas que confluye en él.

Aristas: También llamados arcos, son las líneas que conectan a los vértices, y tienen una longitud de la conexión asociada. Cuando dos aristas se cruzan se le llama cruce [21]. Se pueden clasificar en:

- **Adyacentes:** Estas convergen en un mismo vértice.
- **Paralelas:** Se caracterizan por compartir tanto el vértice inicial como el final.
- **Cíclicas:** Su vértice final es el mismo que el inicial.

Camino: Conjunto de vértices interconectados por aristas.

Tipos de grafo:

Existen diferentes tipos de grafos según sus características, configuración, utilidad y complejidad. A continuación se muestran las distintas clases de grafos principales clasificados según las cualidades de las aristas, vértices, su peso y formas de conexión [21] [22]:

- **Simple:** Definición estándar de un grafo, que indica que este solo acepta una arista para unir dos vértices.
- **Multigrafo:** Acepta más de una arista.
- **Dígrafo:** Poseen una orientación en las aristas, representada por una flecha (ver Figura 12).
- **Etiquetado:** Los vértices tienen etiqueta y las aristas un peso.
- **Aleatorio:** Sus aristas están asociadas a una probabilidad.
- **Hipergrafo:** Las aristas son inicidentes a 3 o más vértices.
- **Infinito:** El cardinal de los vértices y aristas es infinito.
- **Plano:** Este se puede representar sin ninguna intersección entre vértices y aristas.
- **Regular:** Todos sus vértices tienen el mismo nivel de valencia.

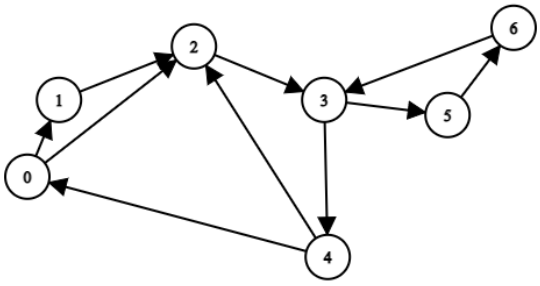

Figura 12: Ejemplo de dígrafo.

Matrices de interés

Entre las matrices que serán de principal utilidad para el algoritmo de sincronización y control de formaciones se puede mencionar las siguientes [21] [22]:

- **Matriz de Adyacencia:**

  El grafo se representa con una matriz cuadrada A de tamaño n^2, siendo n el número de vértices. Si existe una arista entre el vértice x y el y, el elemento $a_{\mathrm{x,y}}$ es 1, si no, es 0.

- **Matriz de Incidencia:**

  El grafo se representa por una matriz de $A \times V$, es decir aristas por vértices. La matriz según v,m brinda información sobre la arista, con 1 siendo conectado y 0 no conectado.

- **Matriz de grados:**

  Matriz diagonal D que contiene información del grado de cada vértice, lo que indica cuantas aristas están conectadas al vértice. Combinando esta y la matriz de adyacencia se consigue una matriz laplaciana [23][24].

- **Matriz Laplaciana:**

  Esta matriz L se obtiene al restar la matriz de adyacencia A a la matriz de grados D de un grafo. Es decir $L = D - A$ [25].

- **Matriz de Rigidez:**

  Esta matriz sirve para representar grafos rígidos, y cuyo enfoque es relacionar los desplazamientos de un conjunto de vértices de una estructura, con las fuerzas exteriores que se necesitan para poder lograr un desplazamiento. Ya que sus distancias entre vértices son constantes, el grafo se mueve en su totalidad como una estructura rígida [22].

  Existe otra variación de este grafo, que forma parte de la familia de gráficos de Lamam, llamado grafo mínimamente rígido, o gráfico de Laman. Se describe como un grafo $G = (V, E)$ con n vértices $V = \{1,2,...,\mathrm{n}\}$, $m = |E|$ aristas que cumple con $m = 2n - 3$ y cada subconjunto con $k \geq 2$ vértices abarca hasta $2k - 3$ aristas [26].

  Para construir un grafo mínimamente rígido es necesario usar la variación de la matriz de rigidez y ejecutar el algoritmo de inserción de Henneburg, que sirve para la generación de este tipo de grafo "flexible", ya que cada vértice se queda con dos grados de libertad. El algoritmo consiste en los siguientes pasos [27]:

  - Numerar todos los vértices.
  - Agregar una arista entre el vértice 1 y el vértice 2.
  - Los vértices restantes se van agregando en orden al componente conectado del grafo, conectando cada uno a la estructura de grafo con 2 aristas.
  - Mantener el número de conexiones necesarias para n vértices por debajo de $(n^2 - n)/2$.

Teoría de control

Para lograr implementar un manejo de las formaciones de agentes robusto, y mantenerlos en las posiciones adecuadas de un grafo mínimamente rígido es necesario tomar en cuenta el elemento de control. Luego de haber construido un grafo mínimamente rígido, se debe tomar

en cuenta la información devuelta por los agentes. Esto desemboca en la necesidad de un grafo etiquetado, es decir sus aristas tienen una ponderación o peso, según la dinámica de un lazo cerrado del sistema de control multi-agente. El hecho de que sea dinámico implica que la longitud de las aristas varía en el tiempo [15] [27].

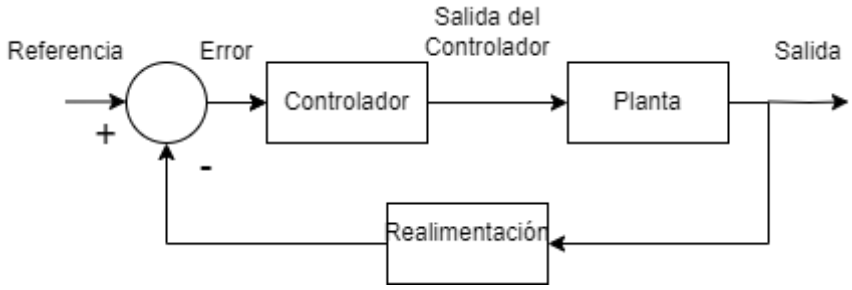

Figura 13: Lazo cerrado de control.

Control de formaciones

El control de formaciones se basa en dos niveles de control, uno superior y uno inferior. El superior se encarga del comportamiento de los robots como agentes, para mantener sus posiciones relativas entre así, así como la posición global de los mismos [15]. El control de capa inferior se encarga de controlar la velocidad de las ruedas para que los motores de las ruedas izquierda y derecha coincidan con la magnitud esperada.

Modelo del robot diferencial

Es importante adaptar el movimiento de formaciones de enjambre al contexto donde se implementará el algoritmo, pues en una simulación de partículas el movimiento es mucho más simple, pues carecen de dimensiones, volumen y masa. La adaptación corresponde en este caso al modelo de un robot diferencial, pues estos son los agentes con la tarea de ejecutar las trayectorias encontradas[15] [16].

Este modelo contempla las dimensiones físicas del robot, y las distancias l del motor hasta su centro, ϕ es el ángulo de orientación del uniciclo en el plano XY, v es la velocidad lineal, ω es la velocidad angular del robot y r es el radio de las ruedas del robot. El subíndice $ctrl$ indica control.

Se tienen las siguientes ecuaciones principales para analizar la cinemática del robot [28]:

$$v = \frac{r(\dot{\Phi}_R + \dot{\Phi}_L)}{2} \tag{1}$$

$$\omega = \frac{r(\dot{\Phi}_R + \dot{\Phi}_L)}{2l} \tag{2}$$

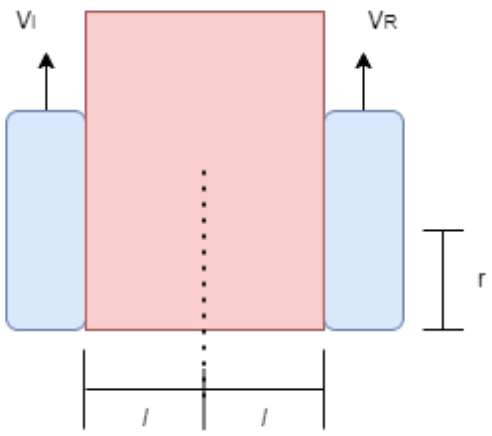

Figura 14: Modelo de un uniciclo en 2D.

De donde se puede llegar a las siguientes expresiones de la velocidad angular controlada de tanto la rueda izquierda como la derecha, como se muestra a continuación:

$$\dot{\phi}_{L,ctrl} = \frac{v_{ctrl} + l\omega_{ctrl}}{r} \tag{3}$$

$$\dot{\phi}_{R,ctrl} = \frac{v_{ctrl} + l\omega_{ctrl}}{r} \tag{4}$$

De forma resumida, para pasar de un modelo de uniciclo no holónomico, al robot móvil, se deben realizar tres pasos:

- Se prueba que el modelo del agente robótico pueda mapearse al uniciclo.
- Se aplica control al uniciclo para calcular las velocidades de control v_{ctrl} y ω_{ctrl}.
- Se realiza un mapeo de las velocidades de control hacia el robot real.

Herramientas matemáticas relevantes

Para lograr efectuar un control de múltiples capas con un funcionamiento correcto y adecuado para mantener la formación del enjambre de forma óptima se recurre a funciones, ecuaciones y otras herramientas matemáticas.

Ecuación de consenso

Para mantener la formación de agentes en los lugares asignados es necesario aplicar el concepto de la ecuación de consenso. Esta herramienta toma en cuenta el centro de masa de la formación y se obtiene la velocidad para cada agente individual, lo que induce a mantener la forma del grafo. Se describe como:

$$v_i = \sum_{i \in N(j)} (x_i - x_j) \tag{5}$$

Donde N es el número de vecinos j, es decir agentes en los vértices adyacentes/conectados a la unidad de interés i.

Al añadir pesos se obtiene la ecuación derivada de velocidad para la formación con tensiones de aristas entre agentes (vértices del grafo):

$$\frac{\partial e_{ij}}{\partial x_i} = \omega_{ij}(\|x_i - x_j\|)(x_i - x_j) \tag{6}$$

Con esta ecuación ya es posible despejar el peso para construir el control de formación tomando en cuenta tensiones, así como el mantenimiento de la conectividad [16] [29].

Función de tensión

Las funciones de tensión definen como se comportarán los agentes al intentar mantenerse en sus posiciones asignadas. Según la función de tensión así son las características de reunión de los agentes y cuanta holgura pueden tener estos entre su posición actual y la objetivo. Las principales funciones de tensión usadas para el control de formaciones en la ecuación de consenso son [15]:

- Evasión de colisiones.

$$\epsilon(x) = \frac{x^2}{x - r} \tag{7}$$

- Control de formación: Donde d es la distancia entre agentes.

$$\epsilon(x) = \frac{(x - d)^2}{2} \tag{8}$$

- Combinación de las previamente mencionadas con mantenimiento de conectividad.

$$\epsilon(x) = \frac{(x - d)^2}{(x - r)(x - R)} \tag{9}$$

- Otras combinaciones.
- Coseno Hiperbólico.

$$\epsilon(x) = \frac{(cosh(1.8x - 8.4)}{10} \tag{10}$$

Infraestructura en Robotat

El Robotat (Robot Habitat), como se explicó previamente, es un ecosistema de desarrollo de robots, consistente de varias herramientas útiles para experimentar con la robótica de enjambre, y a continuación se describirán las herramientas generales con las que cuenta el Robotat.

Mesa de pruebas

Es una plataforma plana de 3.8×4.8 m, con bordes alrededor de ella para servir como barreras delimitadoras para que los robots se mantengan dentro del ecosistema. Además cuenta con 6 cámaras de captura de movimiento alrededor de la plataforma que forman parte del sistema OptiTrack.

OptiTrack

Es un sistema de captura de movimiento con cámaras ultra precisas fabricado por la empresa OptiTrack, cuyas principales aplicaciones en el mundo son: producción virtual para películas, ciencias del movimiento, realidad virtual, robótica, animaciones, etc. A continuación se muestran los distintos dispositivos que lo componen y un ejemplo de montaje [30].

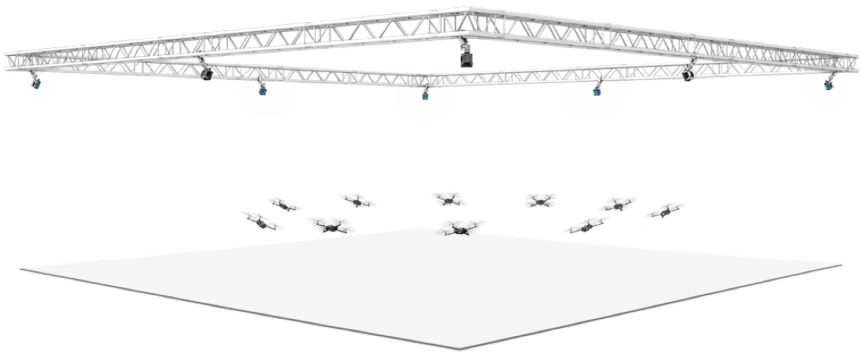

Figura 15: Ejemplo de sistema de captura de movimiento OptiTrack [30].

El modelo disponible en la universidad es el Prime^x 41 (Figura 16, cuyas características más importantes son:

- Rango de captura: 100 pies, 290 pies^3/por cámara para marcadores pasivos y 1000000 pies^3/por cámara para marcadores activos.
- Infrarrojos discretos.
- Captura de imágenes.
- Incertidumbre de ± de 0.1 mm y errores rotacionales menores a 0.5 grados.
- Lentes de 12 mm.

Figura 16: Cámara de Captura de Movimiento Prime^x 41 [30].

- Retrocompatibilidad.
- Captura de Datos 2D/3D y cuerpos rígidos.
- Calibración y sincronización de cámaras.
- Precio: $6499

Comunicación del Robotat

El Robotat tiene la capacidad de realizar mediciones por medio de las cámaras OptiTrack, transmitirlas usando un switch de por medio, a un servidor principal en una computadora de laboratorio por medio del protocolo UDP. Este servidor de Python luego transmite los datos por medio de Wi-Fi usando un Router. Las computadoras que se encuentre en el rango del inalámbrico del Router pueden conectarse al servidor y extraer datos específico según se necesiten, como coordenadas, ángulos de Euler, pose, etc.

Por último, el Robotat cuenta con plataformas robóticas diferenciales, los Pololus 3Pi+, y se les puede enviar comandos a estos por medio de Wi-Fi también.

Software

Para lograr manejar y controlar los agentes de robótica de enjambre es necesario involucrar varios tipos de software, entre ellos lenguajes de programación especializados y simuladores de precisión con enfoque en entornos físicos. A continuación se menciona el software principal.

Matlab

Matlab es una plataforma de programación y cómputo numérico especializada en temas de ingeniería y ciencias, para analizar datos, desarrollar algoritmos y crear modelos; desarrollada por la compañía MathWorks [31].

Para este trabajo de validación del algoritmo de sincronización y control de formaciones, se utiliza como el centro de procesamiento de datos y generación de trayectorias, lo que le da un enfoque centralizado a este caso de robótica de enjambre.

Webots

Es una plataforma de código abierto creada por la compañía Cyberbotics [32] centrada en la simulación de robots, provee una ambiente completo de desarrollo para programar, simular y modelar distintos tipos de robots. Cuenta con un GUI moderno, un *physics engine*, un *rendering engine* y un editor de texto. La plataforma permite crear y añadir distintos tipos de objetos dentro de un mundo .wbt, modificar los parámetros de varios componentes, usar modelos ya preexistentes, sensores, implementar controladores (en lenguajes C, C++, Python, Java, Matlab, ROS, o un API).

En esta plataforma se realizan las simulaciones con el modelo adaptado a un uniciclo para los robots diferenciales, en lugar de solo partículas. Los códigos desarrollados en la fase previa trabajada por Maybell Peña [15] están programados en Python para los controladores de Webots.

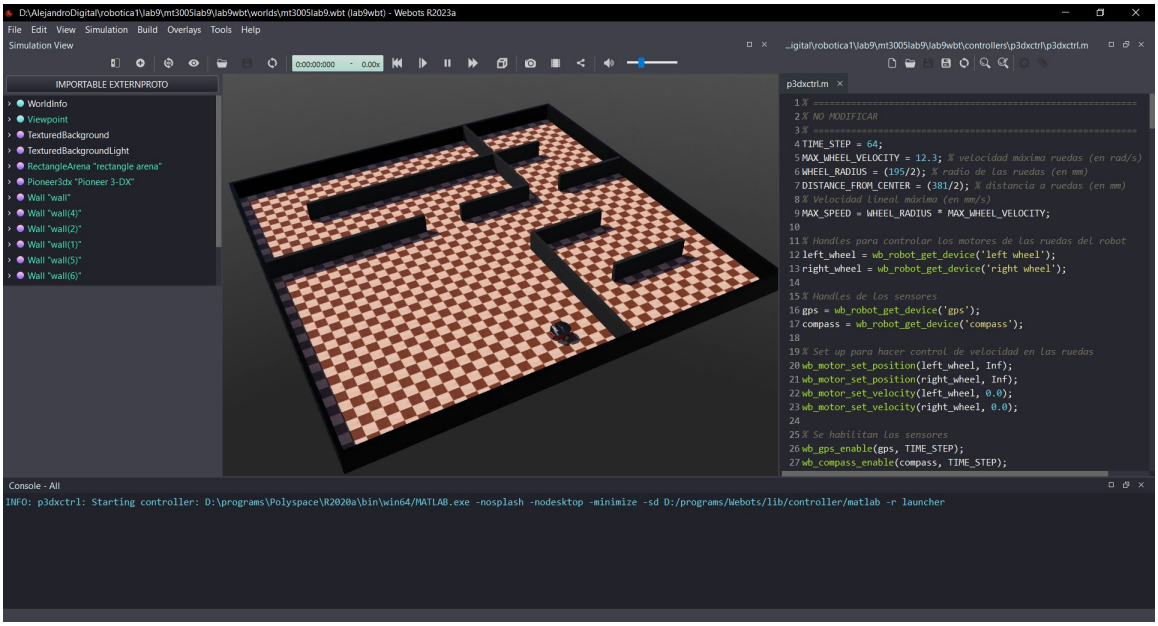

Figura 17: Entorno de desarrollo en Webots.

Hardware

Como último tema a profundizar, se aborda el hardware que se utilizará con el agente para ejecutar las trayectorias generadas por el algoritmo de sincronización y control de formaciones. El robot a utilizar como agente es un robot diferencial Pololu 3Pi+.

Plataforma Móvil: Robot Diferencial Pololu 3Pi+ modificado

La plataforma móvil elegida para realizar la validación del algoritmo es un Pololu 3Pi+ modificado, pues el original tiene una capacidad de procesamiento menor, ya que utiliza como cerebro del robot a un Arduino (ATmega32U4 MCU). Ya que el robot necesita una mayor capacidad de procesamiento para ejecutar las trayectorias, se decidió incluir un microcontrolador ESP32, para realizar el control de capa superior para la ejecución de trayectorias, mientras que el Arduino se encarga de realizar el control de capa baja, es decir se enfoca en controlar la velocidad de los motores.

Figura 18: Pololu 3Pi+ [33].

Las características principales del Pololu 3Pi+ son [33]:

- Modelo: modelo 3pi+ 32U4 OLED Robot
- Encoders de cuadratura dual para control de lazo cerrado de posición o velocidad
- Sensores de línea
- Sensores de choque frontales
- IMU: Acelerómetro de 3 ejes, magnetómetro, giroscopio
- Motores reductores micrometálicos de 30:1 MP de 6V
- Precio: $159.95
- Alimentación: 4 baterías AAA

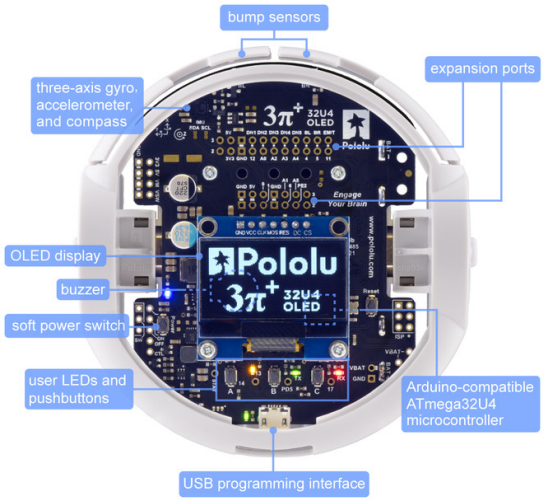

Figura 19: Especificaciones de sensores del Pololu 3Pi+ [33].

- cable USB A a Micro-B cable para programarlos y debuguearlos
- Velocidad máxima: 1.5 m/s
- Peso: 100 gramos (sin ESP32)
- Dimensiones: 97×96×36 mm

Microcontrolador del Agente: ESP32

El microcontrolador ESP32 fue el elegido para adaptarse al Pololu 3Pi+ debido a sus características ventajosas de procesamiento y conectividad. El ESP32 posee un módulo Bluetooth y Wi-Fi integrado, por lo que esto facilita la comunicación a distancia con el control centralizado de los agentes, para recibir la trayectoria a ejecutar. Además cuenta con las siguientes características principales [34]:

- Procesador: microprocesador de 32-bit Xtensa LX6 de doble núcleo (o de un solo núcleo), operando a 160 o 240 MHz y rindiendo hasta 600 DMIPS. Co-procesador de ultra baja energía (ULP).
- Memoria: 540 KiB SRAM
- Conectividad inalámbrica por Wi-Fi y Bluetooth
- Periféricos:
  - 12 bit ADC de hasta 18 canales
  - 2 DACs de 8 bits
  - 4 SPI
  - 2 I2C

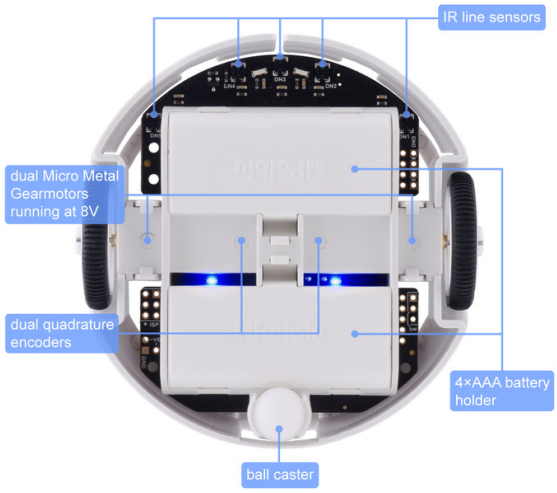

Figura 20: Especificaciones del Pololu 3Pi+ [33].

- 2 I2S
- 3 UART
- PWM
- LED PWM
- Cifrado flash

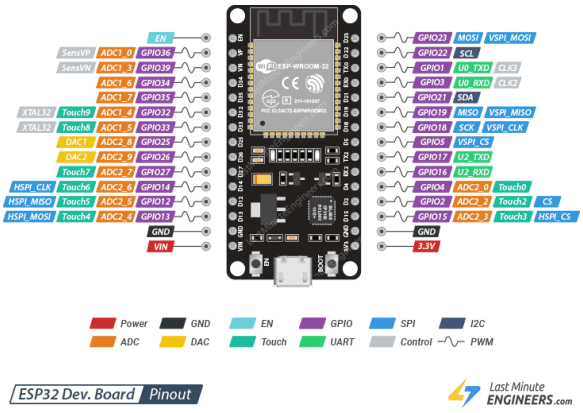

Figura 21: ESP32 *Pinout* [35].

Metodología

Adaptación de software para aplicación en Robotat y Pololu 3Pi+

1. Ejecutar los programas realizados en la fase previa de simulación realizados por Maybell Peña, para comprobar que estos funcionen correctamente, específicamente correr las simulaciones existentes en Matlab y luego en Webots.

2. Modificar los programas realizados en la fase previa para entender como funcionan los programas al hacer variantes de los escenarios de simulación, y variar parámetros, llegando así a tener una mayor familiarización con los mismos.

3. Identificar si existen bugs o errores y corregirlos de ser necesario, para que cuando empiece con la fase propia de trabajo se tenga seguridad de que la parte de simulación funcione correctamente, ya que es sobre este cimiento se desarrollará la fase de validación en físico.

4. Recabar información sobre el estado actual de comunicación entre la computadora, el Robotat y los Pololu 3Pi. También averiguar el protocolo de comunicación usado, las funciones existentes en Matlab para comunicarse con el robot, las limitaciones presentes y la forma de transmisión de datos de trayectoria y control para los agentes físicos.

Evaluación de generación de trayectorias usando marcadores en el Robotat con datos del sistema de captura de movimiento OptiTrack

1. Entender el funcionamiento del sistema de captura de movimiento OptiTrack, datos que se pueden obtener del mismo y características principales.

2. Realizar pruebas con el OptiTrack usando marcadores, para familiarizarme con el sistema. Extraer los datos relevantes del Robotat, como posiciones de agentes (comenzando con solo marcadores) y obstáculos del OptiTrack, para generar las trayectorias por medio de software en la computadora.

3. Ver si existe la posibilidad de replicar las dimensiones del Robotat en un mundo de Webots y simular a los agentes en Webots ejecutando las trayectorias.

Observación del comportamiento de los robots diferenciales al ejecutar las trayectorias del algoritmo de sincronización y control

1. Según el estado actual de la comunicación y capacidad de movimiento del robot diferencial, analizar si es suficiente para la ejecución de trayectorias. De ser necesario realizar modificaciones o crear nuevas funciones para cumplir las necesidades.

2. Entender el funcionamiento del Pololu 3Pi+ tanto en comunicación, funcionalidad y procesamiento de instrucciones. También conocer sus características relevantes tales como vida de la batería, rango de velocidades, torque de los motores, etc.

3. Realizar pruebas de envío de instrucciones directamente a un solo robot diferencial para observar su comportamiento individual.

4. Ejecutar un trayectoria individual por parte de uno de los agentes.

5. Ejecutar un trayectoria individual por parte de uno de los agentes, con evasión de obstáculos, en diversos escenarios.

6. Migrar las trayectorias para la ejecución de los robots diferenciales, primero solo partiendo de posiciones arbitrarias/aleatorias para que se coloquen en las formaciones correspondientes.

7. Probar la formación inicial a través de distintos escenarios, con diferentes posiciones iniciales y configuraciones de obstáculos.

8. Luego de verificar que los robots son capaces de colocarse en formaciones, probar moverlos a través de un camino de un punto A a un punto B, manteniendo formación.

9. Después de confirmar que se pueden mover de A a B en formación, volver a probar, solo que ahora añadiendo obstáculos.

10. Realizar distintos experimentos creando diversos escenarios de interés, con otros puntos de inicio y final, diferentes configuraciones de obstáculos, formaciones, etc.

Comparación de resultados obtenidos en simulación con los agentes físicos en ambientes controlados

1. Documentar los resultados/datos obtenidos en las simulaciones de Webots/Matlab.

2. Documentar los resultados/datos obtenidos en las ejecuciones de la trayectoria en un ambiente controlado usando el OptiTrack.

3. Luego de generar los resultados tanto en Webots como en un ambiente controlado, comparar tanto los escenarios similares como los equivalentes.

4. Encontrar las diferencias principales del desempeño de los agentes en cuanto a la ejecución de trayectorias.

5. Analizar los datos obtenidos en las pruebas y reportarlos, con sus respectivas conclusiones y recomendaciones.

Cronograma de actividades

#	TAREA	INICIO	FIN
	0. Trabajo escrito /Misceláneo		
1	Lectura de tésis fase anterior y artículos científicos relacionados	5/06/2023	5/06/2023
2	Búsqueda y redacción de antecedentes	5/06/2023	5/06/2023
3	Definición de Título, objetivos y justificación	5/06/2023	5/06/2023
4	Investigación de marco teórico	5/06/2023	5/06/2023
5	Comprensión del tema principal	5/06/2023	5/06/2023
6	Recolectar programas de la fase anterior, ordenarlos, leer los archivos README	5/06/2023	5/06/2023
7	Redacción preliminar de resumen, metodología, índice y elaboración cronograma	5/06/2023	5/06/2023
8	Experimentar con programas en Webots/Matlab. Construcción de prototipo	5/06/2023	26/06/2023
9	Trabajo en documento de tésis	5/06/2023	25/09/2023
	1. Adaptación de software para aplicación en Robotat y Pololu 3Pi+		
1	Comprobación de funcionamiento de programas realizados en la fase previa en Matlab y Webots	5/06/2023	19/06/2023
2	Modificación de programas para comprenderlos mejor	5/06/2023	19/06/2023
3	Identificación de bugs/errores para su corrección, asegurar los cimientos para la fase de pruebas en Robotat.	5/06/2023	19/06/2023
4	Recabamiento de información sobre Robotat y Pololu 3Pi+, funciones existentes y comunicación.	19/06/2023	26/06/2023
	2. Evaluación de generación de trayectorias usando marcadores		
1	Comprensión del funcionamiento del OptiTrack, características y obtención de datos.	12/06/2023	19/06/2023
2	Realización de pruebas con OptiTrack usando marcadores para familiarizarse con estos. Extracción de datos de	19/06/2023	3/07/2023
3	Evaluar si existe posibilidad de simular las dimensiones del Robotat en mundo de Webots y simular los robots	12/06/2023	3/07/2023
	3. Observación de robots diferenciales ejecutando las trayectorias		
1	Analizar el estado de comunicación y capacidad de movimiento del robot diferencial. Realizar ajustes de ser ne	5/06/2023	19/06/2023
2	Comprender características principales de funcionamiento del Pololu 3Pi+.	5/06/2023	19/06/2023
3	Realizar pruebas de envío de instrucciones a agente individual.	3/07/2023	3/07/2023
4	Prueba de ejecución de trayectoria individual por un agente.	10/07/2023	10/07/2023
5	Ejecución de trayectoria individual para probar evasión de obstáculos con diversos escenarios.	10/07/2023	17/07/2023
6	Migrar trayectorias para juntar a múltiples agentes en formación partiendo de posiciones aleatorias.	17/07/2023	17/07/2023
7	Probar la formación inicial con distintos escenarios, posiciones iniciales y configuraciones de obstáculos.	17/07/2023	24/07/2023
8	Posterior a la verificación de formación inicial, realizar pruebas de movimiento de punto A a B, manteniendo fo	24/07/2023	31/07/2023
9	Realizar pruebas de movimiento de una formación de un punto A a un punto B con obstáculos.	31/07/2023	14/08/2023
10	Elaborar distintos experimentos con diversos escenarios: distintos puntos iniciales y finales, configuraciones,	14/08/2023	18/09/2023
	4. Comparación de resultados: simulación vs ambiente controlado		
1	Documentar los resultados/datos obtenidos en las simulaciones de Webots/Matlab.	5/06/2023	3/07/2023
2	Documentar los resultados/datos obtenidos con las ejecuciones de trayectoria en ambiente controlado.	3/07/2023	18/09/2023
3	Comparar los resultados de los escenarios similares o equivalentes entre simulación y entorno controlado.	31/07/2023	18/09/2023
4	Encontrar las diferencias principales de desempeño de los agentes en cuanto a la ejecución de trayectorias.	7/08/2023	18/09/2023
5	Analizar los datos obtenidos en las pruebas y reportarlos con sus respectivas conclusiones y recomendaciones	14/08/2023	18/09/2023

Índice preliminar

Referencias

[1] BostonDynamics, *BostonDynamics*, `https://www.bostondynamics.com/atlas`, Accessed: 2023-04-23.

[2] P. Muniganti y A. Pujol, "A Survey on Mathematical models of Swarm Robotics," ene. de 2010.

[3] L. Bayindir y E. Şahin, "A review of studies in swarm robotics," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, n.º 2, págs. 115-147, 2007.

[4] M. Rubenstein, A. Cornejo y R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, n.º 6198, págs. 795-799, 2014. DOI: `10.1126/science.1254295`. eprint: `https://www.science.org/doi/pdf/10.1126/science.1254295`. dirección: `https://www.science.org/doi/abs/10.1126/science.1254295`.

[5] W. Institute, *Kilobots: A Thousand-Robot Swarm*, `https://wyss.harvard.edu/media-post/kilobots-a-thousand-robot-swarm`, Accessed: 2023-04-23.

[6] M. Limeira, L. Piardi, V. C. Kalempa, A. S. de Oliveira y P. Leitão, "WsBot: A Tiny, Low-Cost Swarm Robot for Experimentation on Industry 4.0," *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, págs. 293-298, 2019.

[7] C. Perafán, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[8] A. Aguilar, "Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[9] A. H. Elsheikh y M. Abd Elaziz, "Review on applications of particle swarm optimization in solar energy systems," *International Journal of Environmental Science and Technology*, vol. 16, n.º 2, págs. 1159-1170, feb. de 2019, ISSN: 1735-2630. DOI: `10.1007/s13762-018-1970-x`. dirección: `https://doi.org/10.1007/s13762-018-1970-x`.

[10] A. Maas, "Implementación y Validación del Algoritmo de Robótica de Enjambre Particle Swarm Optimization en Sistemas Físicos," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[11] R. Lima, "Implementación y validación de algoritmos de robótica de enjambre en plataformas móviles en la nueva mesa de pruebas del laboratorio de robótica de la UVG," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[12] G. Iriarte, "Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[13] D. Baldizón, "Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[14] W. Sierra, "Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[15] A. M. Peña, "Algoritmo de sincronización y control de sistemas de robots multi-agente para misiones de búsqueda," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[16] A. M. Peña, M. Zea y L. A. Rivera, "Flat Tension Functions and Minimally Rigid Graphs for Tasks of Synchronization and Control of Multi-Agent Robotic Systems," en *2022 IEEE 40th Central America and Panama Convention (CONCAPAN)*, 2022, págs. 1-6. DOI: 10.1109/CONCAPAN48024.2022.9997593.

[17] K. Aldana, "Desarrollo e implementación de algoritmos de control para un enjambre de drones Crazyflie 2.0 mediante un sistema de visión de cámaras OptiTrack," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[18] J. Rodríguez, "Diseño e Implementación de una Plataforma Móvil Para Aplicaciones de Robótica de Enjambre - Fase III," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[19] L. Nij, "Evaluación y validación de plataformas móviles para aplicaciones prácticas de robótica," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[20] O. Soysal, E. Bahçeci y E. Şahin, "Aggregation in swarm robotic systems:Evolution and probabilistic control," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, n.º 2, 2007, ISSN: 1300-0632. dirección: http://search/yayin/detay/67384.

[21] A. M. T. Lucca, "Teoría de Grafos (Primera Parte)," *Revista de Educación Matemática*, vol. 15, n.º 1, ago. de 2021. dirección: https://revistas.unc.edu.ar/index.php/REM/article/view/10926.

[22] Wikipedia, *Teoría de grafos*, https://es.wikipedia.org/wiki/TeorÃŋa_de_grafos, Accessed: 2023-05-20.

[23] F. Chung, L. Lu y V. Vu, "Spectra of random graphs with given expected degrees," en, *Proc Natl Acad Sci U S A*, vol. 100, n.º 11, págs. 6313-6318, mayo de 2003.

[24] B. Mohar, "On the Laplacian coefficients of acyclic graphs," *Linear Algebra and its Applications*, vol. 422, n.º 2, págs. 736-741, 2007, ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2006.12.005. dirección: https://www.sciencedirect.com/science/article/pii/S0024379506005325.

[25] Hmong, *Matriz laplaciana*, https://hmong.es/wiki/Laplacian_matrix, Accessed: 2023-05-20.

[26] R. Haas, D. Orden, G. Rote, F. Santos, B. Servatius, H. Servatius, D. Souvaine, I. Streinu y W. Whiteley, "Planar minimally rigid graphs and pseudo-triangulations," *Computational Geometry*, vol. 31, n.º 1, págs. 31-61, 2005, Special Issue on the 19th Annual Symposium on Computational Geometry - SoCG 2003, ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2004.07.003. dirección: https://www.sciencedirect.com/science/article/pii/S0925772104001063.

[27] L. Krick, "Application of graph rigidity in formation control of multi-robot networks," Tesis doctoral, Universidad de Toronto, 2007.

[28] M. Zea, *MT3005 - Lecture 13 slides*, Notas de clase, Accessed: 2023-05-21.

[29] M. Rebollo, "Generalización de procesos de consenso en redes complejas," Tesis doctoral, Universidad Politécnica de Madrid, 2019.

[30] OptiTrack, *OptiTrack*, https://optitrack.com/, Accessed: 2023-05-21.

[31] MathWorks, *Matlab*, https://la.mathworks.com/products/matlab.html, Accessed: 2023-05-21.

[32] Cyberbotics, *Cyberbotics: Robotics simulation with Webots*, https://cyberbotics.com/, Accessed: 2023-05-21.

[33] Pololu, *3pi+ 32U4 OLED Robot*, https://www.pololu.com/category/280/3pi-plus-32u4-oled-robot, Accessed: 2023-05-21.

[34] *ESP32 Series*, ESP32, v4.2, Espressif Systems, mayo de 2023.

[35] L. M. Engineers, *ESP32 Pinout Reference*, https://lastminuteengineers.com/esp32-pinout-reference/, Accessed: 2023-05-21.