

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**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**

Trabajo de graduación presentado por Rubén Alejandro Lima García
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**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**

Trabajo de graduación presentado por Rubén Alejandro Lima García
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2022

Vo.Bo.:

(f) _____
Ing. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f) _____
Ing. Luis Alberto Rivera Estrada

(f) _____
[1in] (f) _____
(f) _____

Fecha de aprobación: Guatemala, de de .

Prefacio

La elaboración de este trabajo fue posible gracias a los conocimientos adquiridos a lo largo de la carrera como Programación en Matlab y C, sistemas de control, uso de raspberries entre otros y algunos adquiridos en este año como la programación en C y robótica. Este trabajo es una parte importante en el desarrollo de un entorno físico para aprender y entender la robótica swarm dentro del laboratorio de robótica de la Universidad Del Valle de Guatemala.

Agradecimientos primeramente a Dios, a mi familia Por el apoyo dado a mi en todo este recorrido tanto económico como personal. A mis compañeros con quienes luche para salir adelante en todo y a toda persona que me ayudara en el camino hacia esta meta.

Índice

Prefacio	v
Lista de figuras	x
Lista de cuadros	xi
Resumen	xiii
Abstract	xv
1. Introducción	1
2. Antecedentes	3
2.1. Robótica de enjambre	3
2.2. Programmable Robot Swarms	3
2.3. Bluebots	4
2.4. Termite-inspired robots	5
2.5. OFFensive Swarm-Enabled Tactics (OFFSET)	5
2.6. Megaproyecto Robotat	6
2.7. Bytebot	9
3. Justificación	11
4. Objetivos	13
5. Alcance	15
6. Marco teórico	17
6.1. Robótica de enjambre	17
6.1.1. Taxonomía	23
6.1.2. Desventajas de la robótica swarm	29
6.1.3. Futuras Aplicaciones	29
6.2. Robots Móviles	31
6.3. AlphaBot2-Pi	31

6.4.	Raspberry Pi	31
6.4.1.	Software	33
6.4.2.	Hardware	33
6.5.	Protocolo de transmisión TCP	33
6.6.	Libreria ZeroMQ	34
6.7.	Sockets	35
6.8.	Hilos	35
6.9.	Particle Swarm Optimization	36
6.10.	Ant Colony Optimization	37
7.	Plataforma OptiTrack	39
7.1.	Hardware	39
7.1.1.	Cámaras	39
7.2.	Marcadores	40
7.2.1.	Montaje	41
7.3.	Software	42
7.4.	Comparación	43
7.5.	Nuevo servidor para agentes móviles	44
8.	Plataformas móviles	45
8.1.	Diseño	45
8.1.1.	Bytebot	45
8.1.2.	Bytebot3B	46
8.2.	Funcionamiento	49
9.	Pruebas de algoritmo en plataformas móviles	51
9.1.	Comunicación de Optitrack con agentes	51
9.2.	Procesamiento de información	53
9.3.	Pruebas y Ejecución	55
10.	Conclusiones	59
11.	Recomendaciones	61
12.	Bibliografía	63

Lista de figuras

1.	Kilobots [3].	4
2.	Prototipo Bluebot[4].	4
3.	Robots termita junto a los bloques de construcción [5].	5
4.	Robot terrestre UGS [7].	6
5.	Trayectoria generada por marcadores PSO y controlador TUC [8].	6
6.	Partículas en las esquinas de la región de búsqueda luego de la ejecución de la primera prueba del PSO tuner con neuronas [9].	7
7.	Partículas en posiciones iniciales [10].	7
8.	Partículas en posiciones finales [10].	8
9.	Ejemplo de Comparación de rutas entre matlab y C++ a un nodo específico [9].	8
10.	Vista Inferior del modelo impreso [12].	9
11.	Vista Superior del modelo impreso [12].	9
12.	Diagrama de características.	17
13.	Representación de centralización	18
14.	Representación de descentralización	18
15.	Representación de escalabilidad	19
16.	Representación de robustez	19
17.	Representación de flexibilidad	20
18.	Representación de comunicación	20
19.	Representación de stigmergy	21
20.	Representación sensores	21
21.	representación de comunicación a través de wifi	22
22.	Enjambre	22
23.	Comportamiento de enjambres robóticos [20].	23
24.	Representación de comportamiento	24
25.	Representación de Patrones	24
26.	Representación de ensamblaje	25
27.	Representación de ensamblaje	25
28.	Representación de navegación	26
29.	Swarms Explorando	26
30.	Representación de transportación	27

31.	Flocking	27
32.	Toma de decisiones	28
33.	Alphabot 2 [35].	32
34.	Raspberry Pi 4 Model B [37].	32
35.	Ejemplo conexión TCP.	34
36.	Logo Libreria zeroMQ.	34
37.	Representación de hilos.	36
38.	Representación de convergencia de partículas a su global best [43].	36
39.	Funcionamiento del Algoritmo Ant Colony [44].	37
40.	Camara ex41	40
41.	Marcadores 2,9 y 7	40
43.	Montaje final alrededor de la mesa de pruebas.	41
46.	Diagrama de funcionamiento.	43
47.	Recepción de coordenadas.	43
48.	Detección de coordenadas anteriormente[10].	44
50.	Conexión para Bytebot3B	47
51.	Diseño para corte Mdf	47
52.	Diseño del PCB	48
53.	Diseño caster ball	48
55.	Recepción agente 1	51
56.	Recepción agente 2	52
57.	Muestra código de recepción y envío	52
58.	Ejemplo de simulación código C	53
59.	Diagrama de flujo 1	54
60.	Diagrama de flujo 2	54

Lista de cuadros

Resumen

Enfocado en la utilización de los algoritmos de inteligencia de enjambre, el principal objetivo de este proyecto es la implementación de estos en plataformas móviles. Con esto se logran algoritmos que puedan ser utilizados independientemente de la plataforma y el lenguaje a disposición, con el fin de crear en un futuro un enjambre de robots útiles para la práctica y la enseñanza.

Para lograr esto se utilizaron los algoritmos anteriormente validados a nivel de simulación y a nivel físico con plataformas físicas estáticas y las plataformas para robótica de enjambre desarrolladas en trabajos de años anteriores. Al verificar los resultados y el funcionamiento se procedió a hacer la migración del código del servidor que recibe los datos del sistema de captura de movimiento OptiTrack a otro lenguaje.

Luego de tener una comunicación con las plataformas y el sistema de captura de movimiento, se procedió a modificar el código de movimiento del Bytebot. Por último se realizaron pruebas y validaciones de los resultados en comparación a las hechas en trabajos anteriores y entre ellas.

Abstract

Focused on the use of swarm intelligence algorithms, the main objective of this project is the implementation of these on mobile platforms. With these algorithms are achieved that can be used regardless of the platform and the language available, in order to create in the future a swarm of useful robots for practice and teaching.

To achieve this, previously validated algorithms were used at the simulation level and at the physical level with static physical platforms and the platforms for swarm robotics developed in works from previous years. When verifying the results and the operation, the server code that receives the data from the Optitrack motion capture system was migrated to another language.

After communicating with the platforms and the motion capture system, the Bytebot's motion code was modified. Finally, tests and validations of the results were carried out in comparison to those made in previous works and between them.

CAPÍTULO 1

Introducción

Conformado por una gran cantidad de agentes computacionales simples, inspirados por la auto organización y el control descentralizado de los enjambres naturales como lo son las hormigas, las abejas, los cardúmenes de peces, las terminas y las aves entre otros, la robótica de enjambre es una rama de la robótica que utiliza estos agentes para poder encontrar los mejores caminos computacionales de acuerdo a una meta común previamente programada. Estos agentes son capaces de percibir y modificar su entorno basándose en las percepciones y modificaciones de sus otros compañeros logrando así encontrar el mejor camino hacia la meta programada.

Para que un grupo de robots pueda entrar en la categoría de enjambre necesita tener ciertas características como los son simplicidad, autonomía, control de sistema, comunicación, inspiración biológica y taxonomía.

Para la utilización de estos agentes se necesitan diferentes algoritmos que permiten la predicción del comportamiento del enjambre, usando como punto de partida las características de cada agente individual y su entorno. Algunos de estos algoritmos son *Ant Colony Optimization* (ACO), *Particle Swarm Optimization* (PSO), *Bacteria Swarm Foraging Optimization* (BSFO), *Stochastic Diffusion Search* (SDS), *Artificial Bee Hive Algorithm* (ABHA). En este trabajo se validaron los algoritmos, PSO y ACO.

Se presenta la recepción de los datos provenientes del sistema de captura de movimiento OptiTrack puesto en la mesa de pruebas del laboratorio de robótica de la UVG, así como la validación de las diferentes partes de los algoritmos utilizando estos datos.

Se implementó una comunicación por medio de un servicio de mensajería utilizando la el protocolo TCP/IP para el envío y recepción de información entre los procesos lo cual mejora la eficiencia y el tiempo de respuesta de la plataforma móvil al realizar la migración.

Gracias a esto el proceso de ciclo de trabajo, es decir, ciclo completo entre recibir la coordenada hasta el movimiento del robot ya en posición para recibir la siguiente coordenada es 1 solo y no depende de un temporizador sino que cada acción depende de la otra por lo que

no se saltan pasos. Luego de replicar una gran mayoría de las pruebas hechas anteriormente, se pasó a la ejecución de los algoritmos y las pruebas utilizando el Bytebot sobre la mesa de pruebas del laboratorio de robótica de la UVG.

Posterior a la evaluación y pruebas hechas, se hicieron comparaciones con los resultados obtenidos dentro de la simulación.

CAPÍTULO 2

Antecedentes

2.1. Robótica de enjambre

La robótica de enjambre (*swarm robotics*) es una rama de la robótica que enfoca sus estudios en desarrollar robots simples, escalables y con gran tolerancia a las fallas para realizar tareas complicadas, en lugar de robots robustos y complejos. Las claves de la robótica de enjambre son la reducción de costos y miniaturización ya que estas son unas de las mas grandes complicaciones que tienen los robots grandes y complicados [1].

2.2. Programmable Robot Swarms

Los científicos han estudiado los comportamientos de los animales durante mucho tiempo. Entre los mas interesantes están los de las hormigas y las abejas quienes tienen comportamientos colectivos que distribuyen las tareas de sus colonias en cada integrante de estas. Este comportamiento inspiró a los ingenieros del instituto Wyss a crear simples robots móviles que puedan trabajar colaborativamente tratando de lograr una meta general. Los “Kilobots” son un conjunto de 1024 robots con los que el usuario puede interactuar para realizar tareas complejas. Un ejemplo de uso sería utilizarlos de base para poder desarrollar un enjambre de robots-voladores con instrucciones para polinizar un sector en específico [2].



Figura 1: Kilobots [3].

2.3. Bluebots

En la vida marina se puede ver comportamientos muy sincronizados y complejos, uno de estos casos son los cardúmenes. El comportamiento sincronizado de estos los ayuda a conseguir comida, migrar e incluso escapar y protegerse de depredadores. Sin embargo, dentro de estos no hay un pez líder o una comunicación entre peces. Este tipo de comportamiento es llamado implícito, este denota que los peces individuales toman decisiones dependiendo de lo que ven que sus vecinos hacen. Este comportamiento ha fascinado a los ingenieros y científicos. En 2021 los investigadores del instituto Wyss y la escuela de ingeniería y ciencias aplicadas John A. Paulson (SEAS) desarrollaron un robot con forma de pez que puede sincronizar sus movimientos junto a otros robots en un Cardumen sin ayuda ni control externo. Los llamados “Bluebots” funcionan a base de un sistema guiado por visión de leds azules. Cada uno esta equipado con 2 cámaras y 3 luces LED, las cámaras detectan las luces LED de los Bluebots vecinos y usan un algoritmo para determinar la distancia, dirección y hacia donde se dirigen sus contrapartes. Bajo este sistema los investigadores lograron representar diferentes comportamientos como concentración, dispersión y formaciones circulares [4].



Figura 2: Prototipo Bluebot[4].

2.4. Termite-inspired robots

Los mas grandes constructores de la naturaleza sin duda, son las termitas. Estos animales hacen estructuras de tamaño colosal siendo ellas diminutas, lo hacen en conjunto pero no hay ningún capataz, maestro de obras o comandante. Estos animales reaccionan a su alrededor para tomar la mejor decisión que ayude a la construcción de su termitero (hogar). Bajo esta idea el instituto Wyss desarrollo unos robots bastante simples que levantan pequeñas estructuras (bloques) y las utilizan para poder armar estructuras pedidas por algún humano. Estos robots pueden escalar las mismas estructuras que están creando para así poder hacer una estructura de un mayor tamaño que el del mismo robot. Estos robots actúan independientemente y no tienen conocimiento de qué está pasando fuera de su área inmediata. Esto hace que reaccionen a su ambiente tomando las mejores decisiones dependiendo de lo que puedan observar y entender [5].

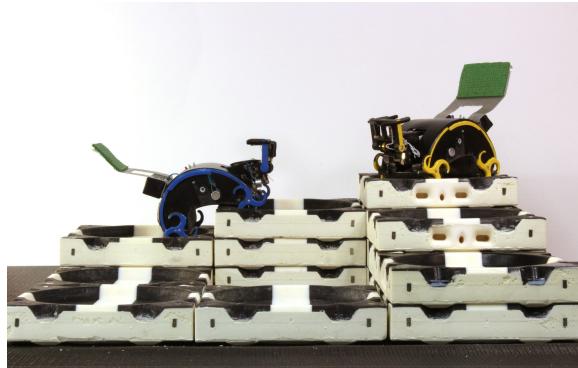


Figura 3: Robots termita junto a los bloques de construcción [5].

2.5. OFFensive Swarm-Enabled Tactics (OFFSET)

Los robots swarm también se ven incluidos en los planes de los ejércitos alrededor del mundo, tal es el caso del americano bajo la dirección del departamento avanzado de investigación para la defensa (DARPA, por sus siglas en inglés). El programa, llamado OFFSET está compuesto por una serie de robots en enjambre de más de 250 unidades. Estos se dividen en sistemas de aeronaves no tripuladas (UAS) y sistemas terrestres no tripulados (UGS) ya que se utilizan para distintas misiones en entornos urbanos muy complejos. El sistema utiliza esta tecnología para el desarrollo de tácticas para operaciones de campo con enfoque en la interacción humano máquina [6].



Figura 4: Robot terrestre UGS [7].

2.6. Megaproyecto Robotat

En la Universidad Del Valle de Guatemala se ha estado trabajando desde la perspectiva de robótica de enjambre desde hace ya un tiempo, uno de estos proyectos es el Robotat. Este constó de cuatro fases, en la primera fase se diseño y construyó una plataforma para el Robotat en donde implementaron un algoritmo de visión por computadora que regresa la posición y orientación del robot desde una perspectiva de dos dimensiones.

En la segunda fase, desarrollada por Aldo Aguilar [8], se utilizo el Particle Swarm Optimization, un algoritmo clásico de control. Este se modificó con la intención de implementarlo en robots diferenciales reales.

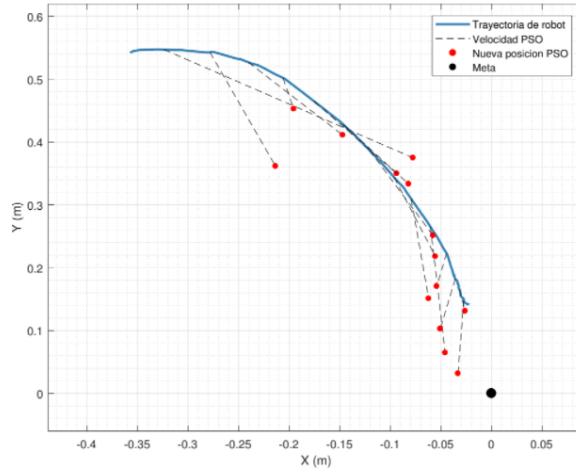


Figura 5: Trayectoria generada por marcadores PSO y controlador TUC [8].

La tercera fase, desarrollada por Eduardo Santizo [9] se enfoca en realizar mejoras al

algoritmo PSO utilizando técnicas de aprendizaje reforzado y profundo. Para las mejoras se empleó el PSO tuner el cual consiste de una red neuronal recurrente la cual utiliza diferentes métricas propias de las partículas del PSO y las convierte, utilizando el procesamiento de una red LSTM, FRU o BiLstm en predicciones de los parámetros que el algoritmo debe utilizar. El resultado fue que la red BiLstm fue la mejor opción de las propuestas ya que redujo el tiempo de convergencia a sus mínimos locales. La fase IV fue desarrollada por distintos alumnos durante el 2021.

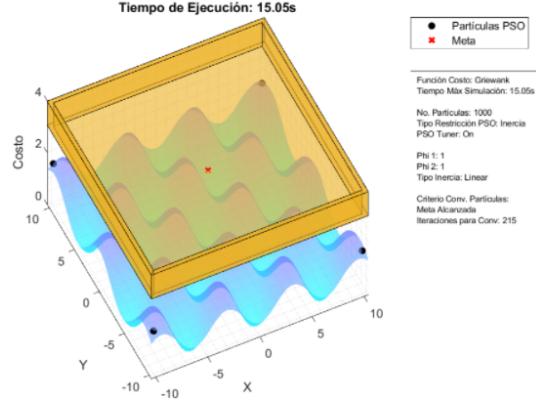


Figura 6: Partículas en las esquinas de la región de búsqueda luego de la ejecución de la primera prueba del PSO tuner con neuronas [9].

En la tesis desarrollada por Alex Maas [10] se implementó el algoritmo Modified Particle Swarm Optimization en sistemas físicos. Se validaron las pruebas hechas anteriormente a nivel de simulación utilizando métodos físicos un poco más rústicos ya que no se contaban con los componentes necesarios. Se replicaron los resultados obtenidos bajo las mismas condiciones iniciales y se validaron estos en la ejecución física. Para lograr esto se desarrollaron dos sistemas de comunicación, el primero usaba un algoritmo de visión por computadora para detectar en tiempo real la pose de cada agente, el segundo permitía a los agentes intercambiar información que era requerida para que el algoritmo siguiera su correcto funcionamiento. Estos algoritmos fueron implementados usando programación multihilos ya que se necesitaba tanto recibir como mandar información paralelamente.

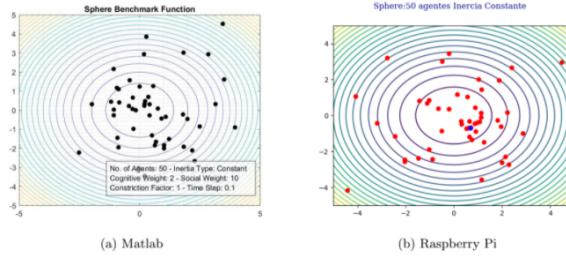


Figura 7: Partículas en posiciones iniciales [10].

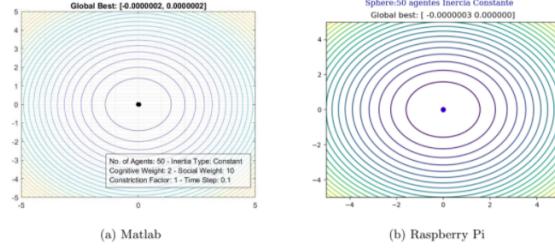


Figura 8: Partículas en posiciones finales [10].

Walter Sierra en su tesis [11], utilizo otro algoritmo para tener una referencia distinta, el *Ant Colony Optimization* (ACO). El migró el algoritmo a la plataforma Raspberry Pi utilizando el lenguaje de programación C. Las pruebas se hicieron usando métodos simples ya que no se contaba con una plataforma móvil y para validarlas se utilizo una plataforma de visión por computadora la cual permitía obtener la pose del robot. Las pruebas presentadas mostraron resultados en escenarios distintos tales como lo son, con y sin obstáculos en el mapa. Uno de los resultados mas importantes fue el lograr implementar un sistema dinámico en donde el algoritmo es capaz de recalcular la ruta al encontrar nueva información dentro de su área cercana, obstáculos o robots.

RUTA AL NODO 56	
Matlab	C++
1	1
12	12
23	23
34	34
45	45
56	56

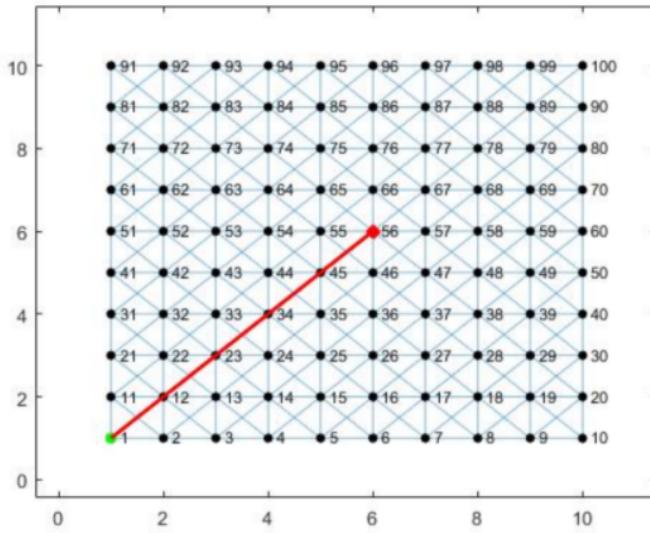


Figura 9: Ejemplo de Comparación de rutas entre matlab y C++ a un nodo específico [9].

2.7. Bytebot

Finalmente Julio Rodríguez [12], en su tesis esta realizando la tercera fase de una plataforma que se ha ideado desde cero llamada Bytebot. La plataforma tiene capacidad de movilizarse, comunicarse por Wifi, recibir información sobre el desplazamiento y modificar su dirección, actualmente se esta implementando la detección y movilización de objetos. Esta plataforma es utilizada como opción para las pruebas físicas a realizar.

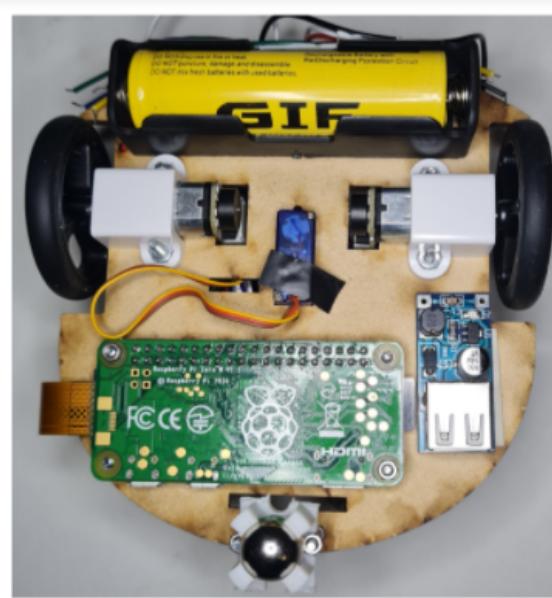


Figura 10: Vista Inferior del modelo impreso [12].

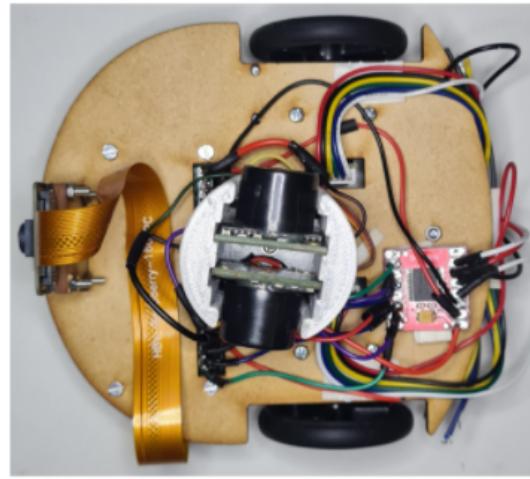


Figura 11: Vista Superior del modelo impreso [12].

CAPÍTULO 3

Justificación

Los algoritmos de inteligencia de enjambre han inspirado distintas investigaciones y tesis dentro de la Universidad Del Valle de Guatemala y su Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica. Para lograr poner a prueba estos algoritmos y su eficacia en el mundo real se necesitan plataformas móviles las cuales ya están disponibles.

En la fase anterior, a pesar de la migración de los algoritmos a plataformas físicas, estas no eran plataformas móviles autónomas por lo cual se debían mover manualmente hacia las posiciones deseadas. Debido a esto la meta es avanzar hacia la realización de un laboratorio de enseñanza e investigación que cuente con distintas plataformas móviles que ya hayan sido probadas y evaluadas sobre los algoritmos previamente diseñados para que puedan aplicarse en ambientes reales.

En esta tesis se busca utilizar las diferentes plataformas móviles a disposición para poner a prueba los algoritmos de inteligencia de enjambre desarrollados por alumnos en tesis previas. Se migrarán los algoritmos a estas distintas plataformas para poder recrear pruebas y comparar resultados.

CAPÍTULO 4

Objetivos

Objetivo General

Implementar y validar los algoritmos de robótica de enjambre desarrollados en fases previas usando plataformas móviles, en la mesa de pruebas de laboratorio de robótica de la UVG.

Objetivos Específicos

- Migrar los algoritmos desarrollados previamente para su ejecución en plataformas móviles con las que se cuenta actualmente.
- Replicar experimentos realizados en fases previas a nivel de simulación y con plataformas no móviles, esta vez usando plataformas móviles en la nueva mesa de pruebas de laboratorio de robótica de la UVG.
- Evaluar y comparar el desempeño de las distintas plataformas, así como los resultados de los algoritmos implementados.

CAPÍTULO 5

Alcance

El alcance de este trabajo de graduación consiste en la implementación dentro de diferentes plataformas físicas como lo es el Bytebot, los algoritmos de robótica de enjambre desarrollados en tesis anteriores. El algoritmo sobre el que se trabajó fue el *Particle Swarm Optimization* desarrollado por Alex Maas [10]. La plataforma utilizada fue Bytebot que fue desarrollada de igual manera en un trabajo de graduación el año pasado por Julio Rodriguez [12] y que nos permite tener movimiento en gran cantidad de direcciones con un pequeño grupo de comandos.

Las pruebas utilizadas fueron las mismas que las evaluadas en los trabajos de graduación anteriores, por lo tanto las comparaciones de los resultados fueron con los resultados evaluados en la mesa de pruebas anterior y no están actualizados a la nueva mesa de pruebas del laboratorio de robótica de la UVG.

Debido a la falta de un número óptimo de plataformas móviles no se probaron los algoritmos de la mejor manera posible por lo que para un otro trabajo de graduación la creación de un mayor numero de plataformas sería de gran utilidad para validar el algoritmo con comunicación entre agentes y obstáculos.

CAPÍTULO 6

Marco teórico

6.1. Robótica de enjambre

La robótica de enjambre es un nuevo acercamiento en robótica para lograr realizar la coordinación de un largo grupo de robots sin tener un control centralizado o ser robots de un alto nivel de entendimiento. Estos están inspirados en los grupos naturales de animales como lo son las hormigas o las abejas.

Un gran numero de pequeños y simples robots pueden realizar tareas complejas en un nivel de eficiencia muchísimo mayor a robots complejos pero en menor cantidad [13]. Esto añade robustez y flexibilidad al grupo y sus usos. Este estilo de robótica tiene ciertas características importantes que actúan como los pilares para los agentes y así poder ser considerados enjambres [14].

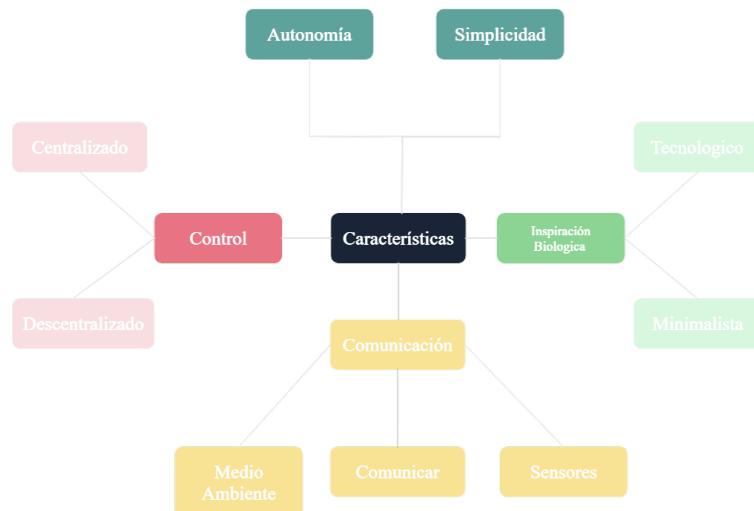
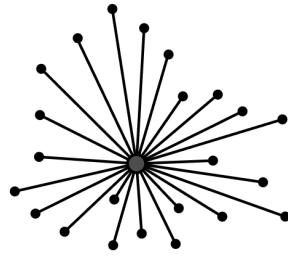


Figura 12: Diagrama de características.

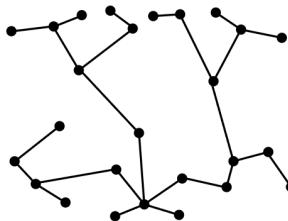
- Simplicidad: Los robots pertenecientes al enjambre deben ser simples, baratos y con partes fácilmente reemplazables, esto debido a que el costo de crear muchos de estos debe ser menos al de crear 1 robot mas caro que pueda realizar la operación por si solo.
- Autonomía: los robots deben ser autónomos, es decir que no deben ser controlados remotamente por un servidor o un humano así como tampoco tienen un agente central o una central a la que envían y reciben la información, sin embargo si el enjambre es muy grande se pueden dejar agentes líderes para secciones cuya función va a ser dirigir a su equipo a cargo.
- Control de sistema: todos los sistemas deben tener un modelo de toma de decisiones que escoge el mejor camino para lograr la tarea dada, hay 2 tipos.
 - Centralizado: Este sistema tiene un robot líder o central que se encarga del control y la organización de los demás agentes dependientes de él.



CENTRALIZED

Figura 13: Representación de centralización

- Descentralizado: Este es modelo más usado y aceptado debido a sus beneficios, se trata de que cada agente es independiente, el algoritmo toma decisiones propias y únicas basadas en su posición y la percepción del medio que lo rodea, como se menciona en [14], este modelo es el de mejores propiedades, a continuación algunas mencionadas tanto en [15].



DECENTRALIZED

Figura 14: Representación de descentralización

- Escalabilidad: El sistema debe ser escalable para poder funcionar con grupos de diferentes sin importar el tamaño. Independientemente del numero de agentes dentro del enjambre completar la tarea de manera eficiente debe ser posible ya que el tamaño del grupo no debería afectar el funcionamiento del sistema. Por lo tanto los enjambres deben funcionar bien si hay pocos agentes o muchos agentes, cuando el grupo es pequeño entonces el sistema debe funcionar perfectamente y cuando el sistema es grande, por el bien del sistema debería utilizarse una cooperación y coordinación aún mejor.

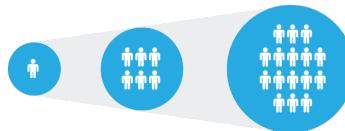


Figura 15: Representación de escalabilidad

- Robustez: La robustez de un sistema se ve al ser afectado por disturbios externos o errores internos. Las perturbaciones atmosféricas incluyen cambios en los alrededores, un incremento en el numero de obstáculos presentes, cambios en la temperatura y el clima dentro de otros factores. Algunos de los componentes del sistema pueden presentar problemas de funcionamiento o no funcionar al 100 por ciento. Estas situaciones y mas deben entrar dentro de los parámetros manejables para los sistemas swarm. Debido a la simplicidad del sistema Incluso si algún agente comete un error el performance total no se ve afectado ya que no son dependientes uno del otro y pueden completar la tarea sin modificar el nivel de eficiencia.



Figura 16: Representación de robustez

- Flexibilidad: La flexibilidad es la característica clave de los enjambres ya que necesitan realizar bastantes tareas al mismo tiempo. El sistema debe ser capaz de poder usar la coordinación y cooperación entre robots para lograr emerger con una variedad de soluciones a las tareas pedidas. Los robots deben ser capaces de trabajar juntos y adaptarse a los cambios en sus alrededores para realizar lo pedido.



Figura 17: Representación de flexibilidad

- Comunicación: Sería imposible realizar las tareas sin una buena comunicación, ya que esta es la base para una buena organización del sistema en pos del cumplimiento de la tarea dada. Las comunicaciones pueden ser globales, lo que aumentaría exponencialmente el costo a medida que el numero de agentes incremente, esto consigo disminuiría la escalabilidad del sistema y por lo tanto, su flexibilidad. Es posible rescatar que este tipo de comunicación es beneficioso cuando se actualizan las estrategias de control ya que aunque cada agente tiene una posición diferente y atado a esto un control diferente, la señal se manda desde una central lo cual hace mas rápida la recepción. Sin embargo a pesar de ser o no ser global, hay 3 tipos de comunicación entre los agentes de cada enjambre. [16].

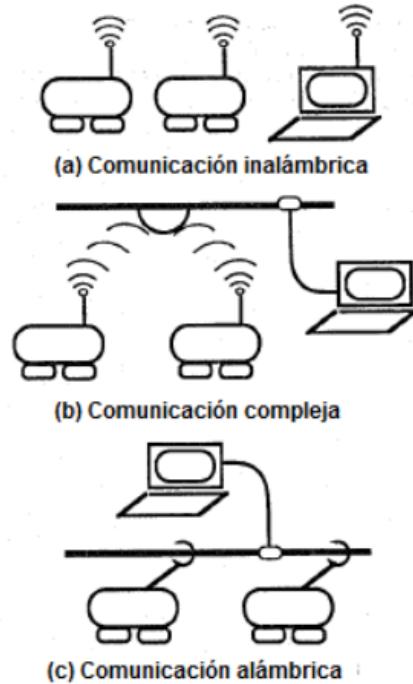


Figura 18: Representación de comunicación

- Interacción a través del medio ambiente: La mas simple de las comunicaciones dentro de la robótica swarm es la inspirada biológicamente, la comunicación a través del ambiente, llamada stigmergy, fue propuesta por primera vez por Grasse en 1995 al estar estudiando las termitas. Este es el nombre que se le dio a la comunicación indirecta entre individuos de la misma colonia que al concluir una tarea, inducía a realizar la siguiente en la lista, En la robótica de enjambre esto se traduce a que las tareas dadas a los agentes del enjambre se ven modificadas dependiendo de su entorno y los elementos que se encuentren en el. Es posible dejar ciertas marcas o señales para estimular a otros agentes a continuar la tarea o a continuar con otra ya que esta está terminada, gracias a las marcas la comunicación con los otros agentes es totalmente indirecta. Sin embargo luego de ciertas inconsistencias, pruebas y criticas a este método, se determino que la comunicación indirecta por medio del ambiente no tiene suficiente validez para su uso en la robótica swarm. [17].

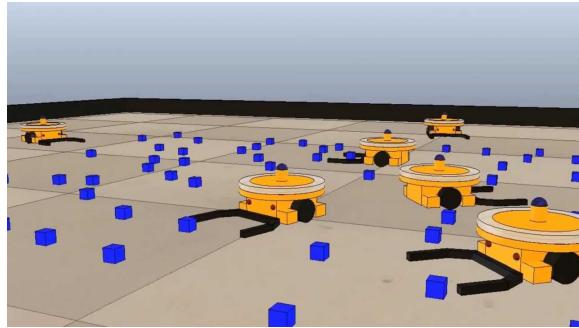


Figura 19: Representación de stigmergy

- Interacción a través de sensores: Esta interacción permite la distinción de cada agente y objeto dentro del enjambre aunque no se comuniquen explícitamente con estos. Esto se hace para poder interactuar de cierta manera con ellos (evasión de obstáculos, búsqueda de coordenadas etc.), lo que genera una comunicación mas simple y eficaz. [18].



Figura 20: Representación sensores

- Interacción a través de comunicaciones: Esta interacción le permite a los robots tener información de las tareas que los otros agentes del enjambre están llevando a cabo y generar un panorama inspirados en esta información. Esta comunicación se asimila a una red WiFi y se compone con 2 tipos, peer-to-peer y broadcast, sin embargo esta opción no está inspirada biológicamente y tiene muchos cuellos de botella, problemas de escalamiento, costo y flexibilidad. [19]

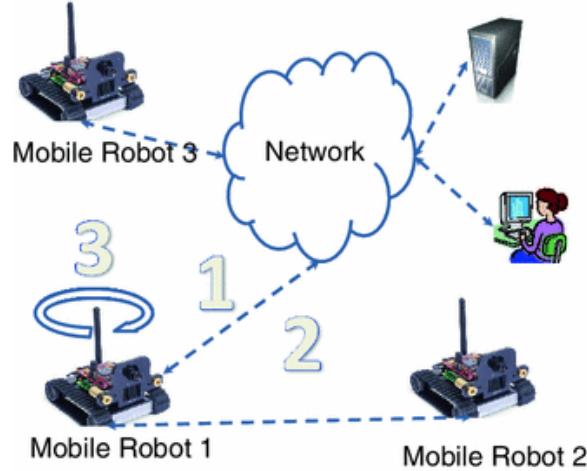


Figura 21: representación de comunicación a través de wifi

- Inspiración Biológica: La robótica swarm tiene su mayor inspiración en la naturaleza y su forma de actuar sin embargo hay 2 enfoques de pensamiento que debaten que tan inspirada en la naturaleza debe ser para poder seguir siendo útil. Cada uno posee ventajas y desventajas únicas, debido a esto el enfoque es dependiente de la tarea y los objetivos a alcanzar con el enjambre. [14].



Figura 22: Enjambre

- Enfoque Minimalista: En este, las características físicas y de comunicación del robot deben ser mínimas, para poder imitar la simplicidad de la naturaleza.
- Enfoque Tecnológico: En este, se trata de utilizar mayor cantidad de componentes electrónicos avanzados, lo cual le da mayor robustez al enjambre pero pierde la simplicidad de la naturaleza.

6.1.1. Taxonomía

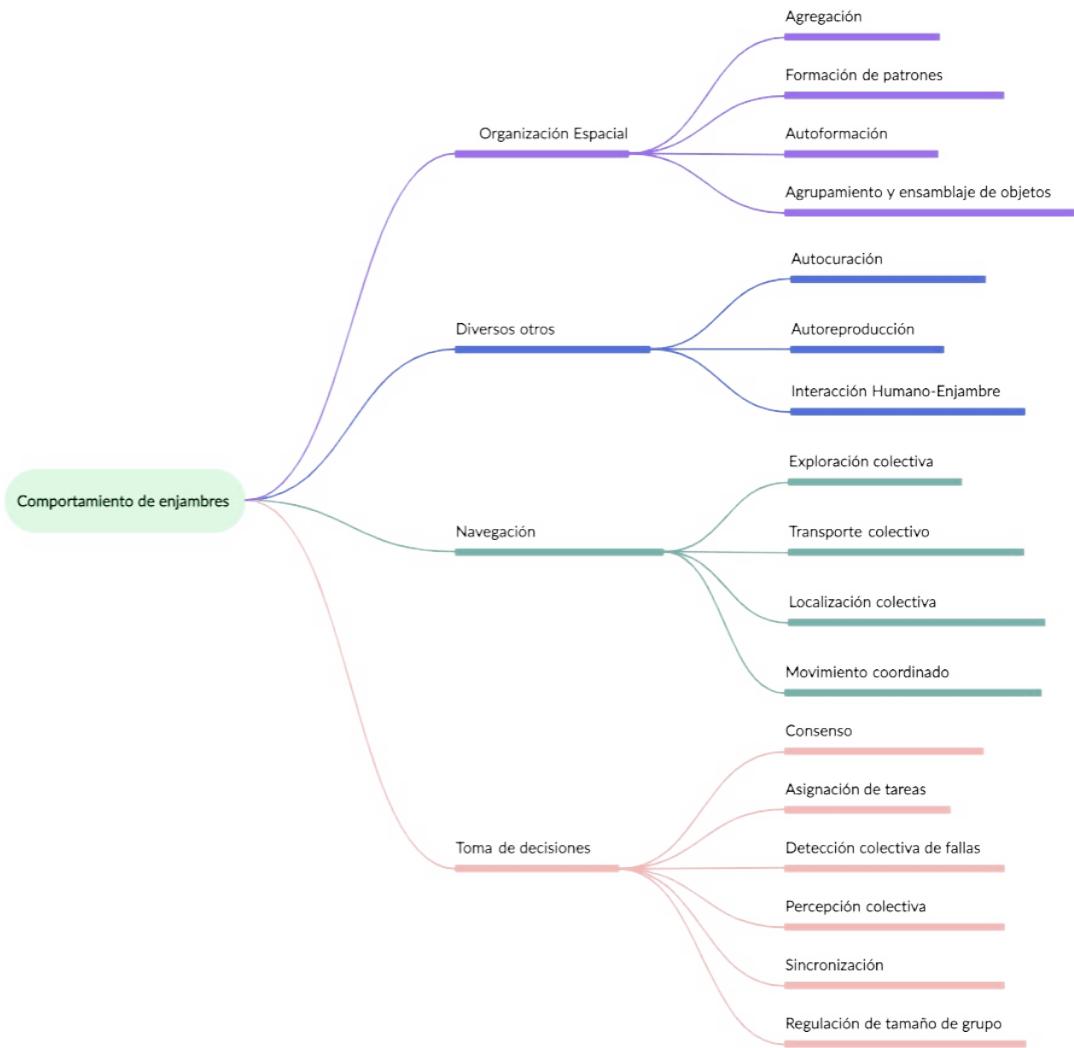


Figura 23: Comportamiento de enjambres robóticos [20].

- Organización Espacial: Los comportamientos descritos a continuación hacen posible el movimiento de los enjambres a través del ambiente para lograr ordenarse a si mismos u objetos en el espacio.
 - Agregación: Mover los robots individualmente para converger a un punto específico en el ambiente. Esto permite acercar a los agentes dentro del enjambre para una futura interacción.

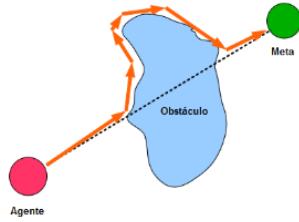


Figura 24: Representación de comportamiento

- Formación de patrones: Ordena a todos los agentes del enjambre en una forma específica. Un caso especial de esto es cuando los agentes se forman todos en linea recta, normalmente para poder establecer comunicación multisalto.

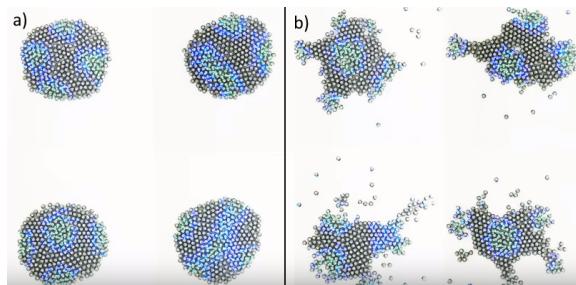


Figura 25: Representación de Patrones

- Auto formación/armado: Conecta los robots en orden para establecer estructuras, pueden ser conectados físicamente o virtualmente con enlaces de comunicación. Un caso especial de esto es la morfogénesis, cuando el enjambre evoluciona a una forma predeterminada.

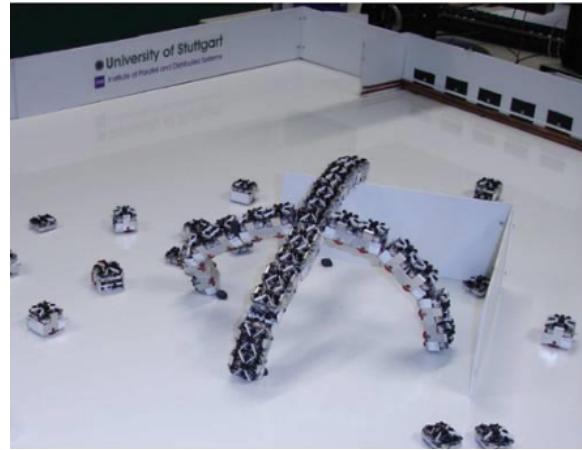


Figura 26: Representación de ensamblaje

- Agrupamiento y ensamblaje de objetos: El enjambre de robots, utiliza los objetos en su ambiente para construir estructuras usando el montaje y ensamblaje.



Figura 27: Representación de ensamblaje

- Navegación: La navegación permite el movimiento coordinado de los enjambres de robots a través de un ambiente. Dependiendo del tipo de ambiente, es el tipo de implementos y habilidades que los enjambres deben tener. En la imagen de abajo vemos robots que están haciendo un ejercicio de exploración en aguas abiertas.



Figura 28: Representación de navegación

- Exploración colectiva: El enjambre de robots navega a través del espacio/ambiente cooperando entre ellos para poder explorarlo. Ejemplos de uso pueden ser, un resumen situacional, búsqueda de objetos, monitoreo de ambiente, establecimiento de una red de comunicación.

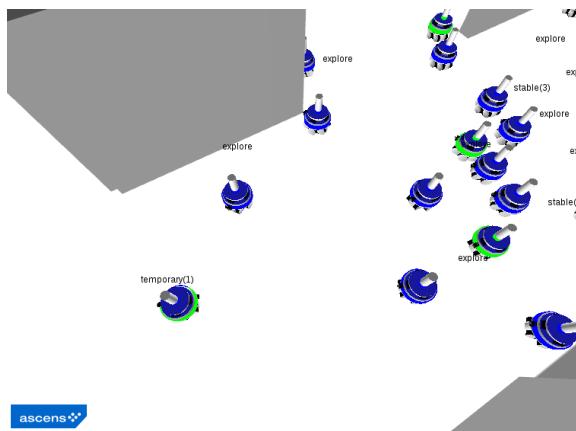


Figura 29: Swarms Explorando

- Transporte colectivo: Cuando hay objetos muy pesados o grandes que se necesitan mover, el enjambre se dispone a moverlo usando un gran porcentaje de agentes.

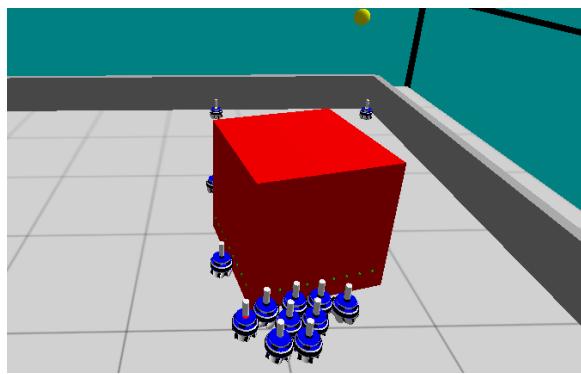


Figura 30: Representación de transportación

- Movimiento coordinado: Mueve los robots del enjambre dentro de una formación, puede ser con una forma predeterminada o una forma arbitraria como se hace en el flocking.

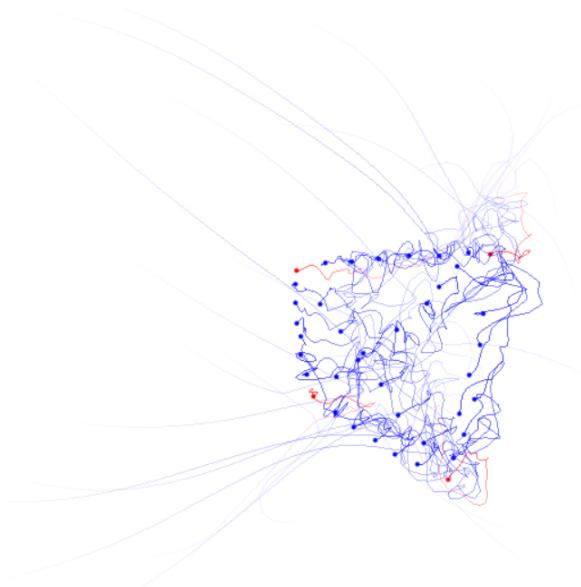


Figura 31: Flocking

- Toma de decisiones: Este comportamiento permite a los robots del enjambre tomar una decisión común sobre un problema o situación presentada.



Figura 32: Toma de decisiones

- Consenso: Permite a los robots individuales dentro del enjambre llegar a un acuerdo sobre una elección dentro de múltiples opciones.
- Asignación de tareas: Asigna las tareas dinámicamente a los robots individuales pertenecientes al enjambre, el objetivo principal es maximizar el rendimiento del enjambre. Si los robots desarrollados son heterogéneos las tareas se distribuyen en consecuencia a sus capacidades para hacer una mejor distribución y aumentar el rendimiento.
- Detección colectiva de fallas: Esta detección determina las deficiencias individuales de los robots pertenecientes al enjambre. Esto permite encontrar robots que no cumplen las funciones dadas por diversos problemas y desvían el comportamiento del enjambre del predefinido.
- Percepción colectiva: Esta combina la información obtenida por los robots individualmente en una big picture information. Permite al enjambre hacer decisiones colectivas de una manera informada, por ejemplo en clasificación de objetos en el espacio, dividir a los robots en un determinado sitio para futuras tareas, determinar la mejor solución para un problema global.
- Sincronización: Para los robots que cuentan con osciladores, alinea la frecuencia y fase de los mismos para que los robots tengan un entendimiento del tiempo y puedan realizar acciones sincronizadas.
- Regulación de tamaño de grupo: Permite a los robots del enjambre formar grupos de determinado numero.
- Diversos Otros: Hay algunos comportamientos que no entran en las otras categorías.
 - Auto curación: Permite al enjambre recuperarse de fallos propiciados por los fallos en los individuos, la meta es poder disminuir el impacto de esta falla al resto de robots lo que conlleva a unas de las ventajas antes mencionada como lo es la robustez.

- Auto reproducción: Permite al enjambre poder crear nuevos robots o replicar algunos planos dados. Esto permite eliminar aún mas la mano humana en los enjambres y hacerlos mas independientes.
- Interacción Humano-Enjambre: Permite a los humanos a cargo del enjambre poder controlar a los robots o recibir información de ellos, usualmente a través de una terminal.

6.1.2. Desventajas de la robótica swarm

Anteriormente se vieron las muchas ventajas de la robótica swarm, sin embargo no todo es perfecto y también tiene algunas falencias. Debido a que el diseño y fabricación de estos robots no se hace a una escala industrial, las aplicaciones dependen totalmente de los robots que la misma empresa tenga o de los que una universidad o instituto de investigación les proporcionen. Algunas otras son:

- Fiabilidad: La robótica de enjambre esta diseñada con base en la naturaleza, sin embargo en la naturaleza el enjambre funciona suponiendo que los miembros van a morir, perderse o fallar. Esto a ellos no les genera un problema debido a que sus integrantes son un numero muy alto, en los enjambres robóticos aunque su diseño permita perder integrantes, esto supone un problema de funcionamiento y económico que las empresas no están dispuestas a afrontar en este momento[21].
- Comportamiento: Aunque es novedoso que los robots autónomos puedan tener control de como se van realizando los proyectos, la seguridad brindada por el control completamente centralizado no ha sido alcanzada. Por otro lado, la seguridad ofrecida por la planta centralizada contra los hackeos tampoco ha sido lograda, ya que al ser una plataforma simple no dispone de memoria y hardware necesario para evitar este tipo de ataques.[22]

6.1.3. Futuras Aplicaciones

Aunque en este punto, muchos países, universidades e instituciones ha desarrollado enjambres de robots, solamente se han utilizado los robots mas simples y minimalistas, equipados con sensores básicos e intercambiando una pequeña cantidad de información. Debido a esto, lo que se esta probando es la teoría y el funcionamiento pero los robots actuales no van a ser escalables a futuras aplicaciones. Si lo comparamos con otros robots de investigación como lo son los hechos por Boston Dynamics, Hanson robotics entre otros los swarms hechos hasta el momento parecen juguetes, sin embargo aunque los swarms deben ser en teoría mucho mas simples que estos, deben pasar de ser objeto de investigación de comportamiento a ser viable para hacer cosas útiles en el mundo. Luego de hacer a los swarms mas viables, nos toparíamos con otros inconvenientes como lo serían el lograr controlarlos de manera eficiente, el mantenimiento y las actualizaciones de hardware [23].

- Agricultura: Para la robótica swarm la agricultura representa un área muy importante para desarrollar soluciones robóticas. Debido a que los problemas en la agricultura están caracterizados por ambientes sin estructura, distribuciones espaciales de gran tamaño y tareas redundantes, la aplicación de una solución flexible, escalable es completamente necesaria [24]. Algunos de los problemas que pueden ser resueltos en la agricultura son:
 - Control automático de hierbas, para poder reducir el trabajo y los costos operativos entras se aumenta la producción y se reduce el trazo químico.
 - Detección automática de hierbas.
 - Mecanismo de extracción para hierbas individuales.
 - mapeo y navegación eficiente alrededor de los cultivos.
- militar-seguridad: Los países en la búsqueda por contar con una mejor fuerza armamental,y de seguridad han incluido arsenales de robots swarm dentro de sus filas tanto terrestres, aéreas y marítimas [25].
- medicinal: En la parte medicinal se unieron los nanobots con la robótica swarm para intentar crear nuevas soluciones para poder tratar el cáncer o enfermedades para las que al momento no se tiene cura. Un ejemplo reciente son los microbots swarm hechos para embolización selectiva en[26] o los milibots desarrollados por el instituto de sistemas inteligentes Max Planck que en un futuro podrían incluso realizar una cirugía no invasiva [27].
- Astronomía: Dentro de este campo, la investigación usando swarm no es mucha pero el uso de estos puede ser de gran ayuda en el futuro, como por ejemplo en [28] se usan los swarms para tratar de detectar materia oscura de 5000 galaxias diferentes o en [29] en donde se introduce la idea de hacer un telescopio swarm, ya que se dividiría en robots o sistemas independientes que luego pueden colaborar juntos.
- Zonas peligrosas y de rescate: Así como los militares mandan robots a desactivar bombas para eliminar el riesgo de fatalidad, en las zonas peligrosas se está haciendo uso de enjambres para poder tratar con los residuos químicos de algún derrame, o búsqueda de supervivientes en desastres naturales. Aplicado a esto, en un futuro pueden incluso transportar personas u objetos fuera del área del desastre sin intervención humana. Estos robots no necesariamente tienen que ser terrestres, también pueden ser UAVs como lo comentan en [30].
- Hogar: En el hogar podrían utilizarse swarms para hacer las tareas mas repetitivas del día al día.
- Manejo autónomo y Smart Traffic: En este caso los carros, semi autónomos o autónomos tienen ya de por sí la infraestructura para ser una red swarm ya que cooperan para encontrar el mejor camino hacia un lugar o intercambian información sobre accidentes o desvíos. Al agregar a estos modelos autónomos mas comportamientos "swarm"podríamos hacer que al compartir información la carga vial sea menor ya que toman la mejor decisión dependiendo del destino final, en una emergencia pueden hacer fila automáticamente para dejar pasar una ambulancia. El modelar la motorización autónoma como un enjambre swarm tiene ciertas ventajas como lo sería que

cada agente pueda evaluar con información local, siguiendo reglas locales, en el caso de una flota mas grande, se intercambiaría información para lograr una red global. Si estas flotas de carros autónomos son manejados por una central no tendrá la habilidad de coordinar y manejar cada auto en el trafico a través de la mejor ruta y tomando las mejores decisiones como lo haría un agente individual cooperando con otros agentes individuales intercambiando información. En la parte motora aún quedan muchos retos por afrontar pero depara un futuro prometedor. [31].

Algunas otras opciones no discutidas acá serían, construcción, moderación de trafico aéreo, redes celulares aéreas, monitorización de clima, seguridad, transporte de cargamentos, entre otros [32].

6.2. Robots Móviles

Un robot móvil es una maquina capaz de moverse a través de su entorno y no esta asegurado a una locación fija.

Esta es una rama de la industria bastante utilizada y en constante cambio e investigación ya que estos robots ayudar a empresas de almacenaje, logística, agricultura y muchas otras [33]. Entre los robots móviles están los autónomos (AMR) guiados autónomos (AVG) [34].

6.3. AlphaBot2-Pi

El Alphabot 2 es un kit robótico para la Raspberry Pi. Este kit incluye muchas funciones bases como lo son, seguimiento de linea, detección de obstáculos, conexiones inalámbricas (Wifi/bluetooth/control remoto/infrarrojo). Esta diseñado para no necesitar soldadura o un armado complicado y una programación de fácil comprensión.

En sus especificaciones incluye:

- Un regulador de voltaje LM2596 que provee a la Pi un voltaje estable de 5V
- Un chip TLC1543 que permite a la Pi usar sensores analógicos
- Un controlador servo PCA9685 para realizar rotaciones de manera mas rápida y sencilla
- Un convertidor UART CP2102 para controlar la Pi vía UART [35].

6.4. Raspberry Pi

La Raspberry Pi es una placa simple de bajo costo y muy alto rendimiento, básicamente una pequeña computadora. Fue desarrollada por la Raspberry Pi Fundación[36].



Figura 33: Alphabot 2 [35].

A medida que la Raspberry Pi fue ganando adeptos se fueron creando varios modelos diferentes, cada uno para tareas o necesidades diferentes. Algunos de los modelos son:

- Raspberry Pi 1 Modelo A (descontinuada)
- Raspberry Pi 1 modelo B (descontinuada)
- Raspberry Pi 2 modelo B
- Raspberry Pi 3 modelo A+
- Raspberry Pi 3 modelo B+
- Raspberry Pi 4 modelo B [37].



Figura 34: Raspberry Pi 4 Model B [37].

6.4.1. Software

Raspberry Pi OS es el software utilizado por la Raspberry Pi. Anteriormente llamado raspbian este sistema operativo tiene mas de 35,000 paquetes diferentes, previamente compilados para darle un mayor alcance al uso de la Raspberry Pi.

Este sistema operativo se puede descargar desde la pagina oficial de Raspberry Pi y tiene una guía completa de instalación [38].

6.4.2. Hardware

La Raspberry Pi cuenta con un procesador ARM con arquitectura RISC (Reduced Instruction Set Computer). Esta arquitectura utiliza instrucciones bastante simples por lo que en la ejecución el consumo de energía es mínimo [39].

6.5. Protocolo de transmisión TCP

El Transmission Control Protocol por sus siglas en inglés es un protocolo desarrollado a mediados de los años 70 por Vint Cerf y Robert Khan debido a la necesidad de conocer al receptor de la información y hacer las comunicaciones bidireccionales

El protocolo TCP funciona de manera sencilla dividida en 3 fases. El primero es generar una conexión emisor - receptor utilizando sockets y siendo autorizada por ambas partes, luego se realiza la negociación de 3 pasos que consta en intercambiar 3 tramas con mensajes de sincronismo y reconocimiento (ACK), los datos pasados son: Espacio disponible en el bufer, cantidad máxima de datos en un segmento y Número de secuencia inicial que se utilizará para enumerar los datos de salida, luego se notifica que la conexión está abierta y lista para comenzar a transmitir/recibir.

Este protocolo permite un intercambio de información fiable, lo que quiere decir que los datos enviados se van a entregar en el orden enviado y sin errores como lo serían tramas incompletas o tramas dobles [40].

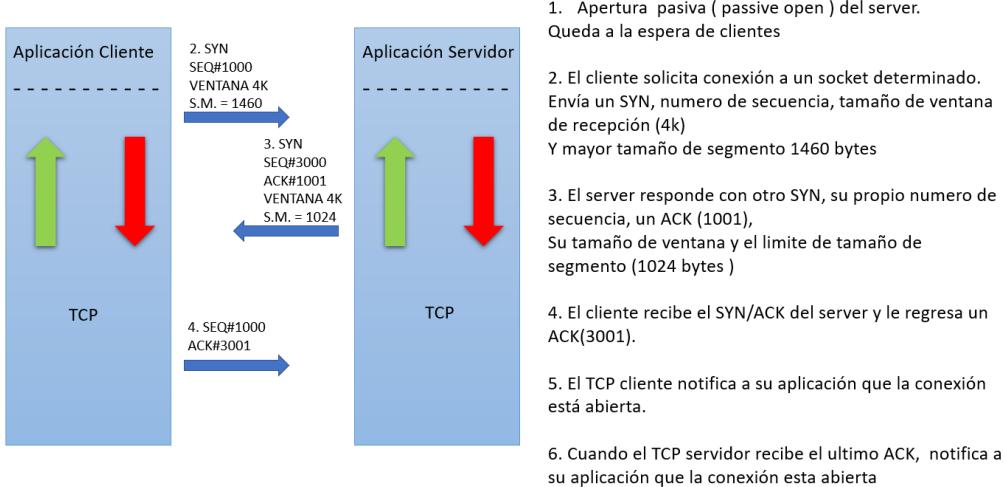


Figura 35: Ejemplo conexión TCP.

6.6. Libreria ZeroMQ

La librería ZeroMQ permite diseñar sistemas de mensajería y comunicación de manera muy simple. Se basa en cola de mensajes y a diferencia de otros agentes intermedios, ZeroMQ no necesita un broker. Esta librería cuenta con diferentes patrones de mensajería como lo son



Figura 36: Logo Libreria zeroMQ.

- **Pair:** Es un socket clásico, bidireccional entre cada cliente y 1 solo servidor.
- **Request-Replay:** Esta se utiliza mayormente para los procedimientos remotos y genera conexiones entre varios servidores y varios clientes.
- **Publish-Subscribe:** Se conectan publicadores contra subscriptores, su uso es mayormente para la distribución de datos por medio de mensajes categorizados.
- **Pipeline:** Conecta los nodos organizados en pasos secuenciales, es necesario para organizar los flujos de procesamiento de mensajes en paralelo o por etapas.

6.7. Sockets

Los sockets son una herramienta que nos ayuda a establecer un canal de comunicación entre 2 programas o rutinas dentro de la misma red. Estos trabajan con sistemas operativos tipo Unix, los cuales tienen servicios de entradas/salida de sistemas [41].

No importando si los programas corren en la misma o diferente computadora, mientras estén conectados a la misma red se pueden utilizar sockets para establecer comunicación.

Existen 2 tipos de sockets

- Orientados a conexión (TCP): Al tener 2 programas que deben intercambiar información, se utilizan sockets para poder transmitirla por medio del protocolo, esos garantizan que la información no va a perder ningún dato y si un proceso está ocupado, el socket espera a que se libere para entregar los datos y continuar la comunicación.
- Sin Conexión (UDP): Se utilizan para transmitir los datos pero no se asegura que los datos lleguen completos o que si el otro proceso esta ocupado el socket con la información va a esperar para entregarla. [42].

6.8. Hilos

Los hilos son tareas que piden ser ejecutadas al mismo tiempo, es decir de forma paralela. Esta es definida dentro del sistema operativo como un subprocesso o una unidad compacta de pequeño impacto que tiene la tarea de planificar los tiempos de procesamiento entre los diferentes procesos. Cada hilo es único por su contador de programa, la pila de ejecución y el estado de la CPU.

Para que los hilos puedan funcionar se comparten recursos, datos y espacios de direcciones por lo cual es mas fácil cambiar de un hilo a otro que la otra manera, la cual es de un proceso a otro. Los hilos poseen un estado de ejecución y al sincronizarse entre ellos tienen una tarea específica y determinada para lograr aumentar la eficiencia del uso del procesador.

los usos mas comunes de los hilos son:

- Trabajo interactivo y en segundo plano
- Procesamiento asíncrono
- Aceleración de la ejecución
- Estructuración modular de los programas

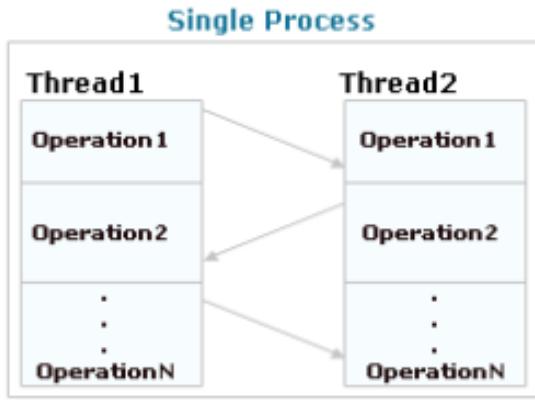


Figura 37: Representación de hilos.

6.9. Particle Swarm Optimization

Este algoritmo pertenece a una categoría de técnicas llamada optimización inteligente y es clasificado como un algoritmo estocástico de optimización basado en población. Este algoritmo simula un grupo de partículas con posiciones y velocidades iniciales asignadas, estas toman el nombre de *soluciones*. El algoritmo obtiene los 2 mejores resultados y la partícula al recibir esta información es actualizada. Uno de estos es llamado *local best*, que como su nombre indica es la mejor solución alcanzada por una partícula hasta el momento. El segundo es rastreado por el mismo algoritmo para ser el *second best*. Este proceso de obtención de resultados se repite hasta un numero específico de iteraciones o se logre la convergencia. Esta convergencia es lograda al momento que todas las partículas son atraídas hacia la partícula con mejor solución global, llamada *global best*. Con estos datos es posible calcular los 2 principales factores a utilizar en la ecuación del PSO, llamados factor Cognitivo y Factor Social [43].

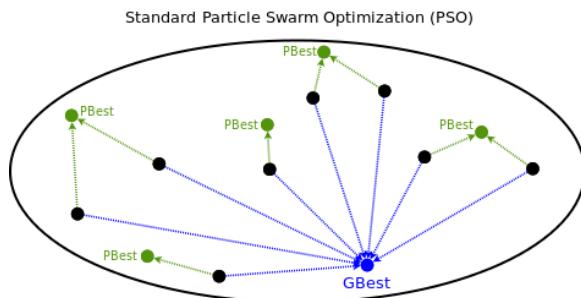


Figura 38: Representación de convergencia de partículas a su global best [43].

6.10. Ant Colony Optimization

El algoritmo de Ant Colony Optimization podemos resumirlo como un algoritmo con una red de agentes asíncronos que se mueve sobre diferentes estados o soluciones que llevan hacia un mismo punto y regresan la información a la central (El algoritmo) para que este determine la mejor ruta y así complete la solución al problema.

Ant Colony Optimization no es un algoritmo en sí, sino que es una clase de algoritmos. El primero fue el Ant System que fue propuesto por Colorni, Doeigo y Maniezzo. La idea principal, inspirada en las hormigas, es una búsqueda paralela sobre varias bifurcaciones dentro del sistema que llevan a un mismo punto. El sistema cuenta con una memoria dinámica que, a medida que los agentes manden información está pueda estructurarla y compararla con información de otros agentes obteniendo así la mejor ruta a la resolución del problema. Entre las variaciones está el Ant System (AS), Ant Colony System (ACS), Max-Min Ant System (MMAS), Ant-Q, Fast Ant System (FAS), Antabu, AS-rank y ANTS [44].

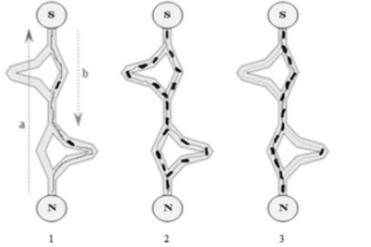


Figura 3: Proceso del algoritmo *Ant Colony Optimization* [14].

Figura 39: Funcionamiento del Algoritmo Ant Colony [44].

CAPÍTULO 7

Plataforma OptiTrack

El sistema de captura de movimiento OptiTrack es fundamental para las pruebas futuras ya que nos retro alimenta con las coordenadas actuales de los agentes dentro de la mesa de trabajo, las cuales son esenciales para el uso de los algoritmos de enjambre. En este capítulo se explicará mas a detalle de que consta este sistema, el software y la configuración para poder utilizarlo, una pequeña comparación de este sistema con el utilizado anteriormente y el funcionamiento del servidor para esta tesis.

7.1. Hardware

Esta sección presenta las partes físicas de las que consta el sistema Optitrack, uso y montaje de las mismas.

7.1.1. Cámaras

Las camaras que son utilizadas por el sistema tienen por nombre Primex41, estas son camaras de alta gama utilizadas por lideres en la industria del entretenimiento y la investigación como lo son Google, Disney, Activision, MIT entre otros, estas camaras funcionan utilizando Ethernet, conectadas a un switch principal [45].



(a) Vista frontal



(b) Vista trasera

Figura 40: Camara ex41

7.2. Marcadores

A pesar de no ser una parte del sistema como tal, son de vital importancia ya que estos son los dispositivos que las camaras del OptiTrack detectan. Los markers utilizados llevan por nombre Hand Rigid Bodies Marker Set y crean cuerpos rígidos únicos para poder obtener las coordenadas de estos. Los marcadores están enumerados para poder identificarlos al momento de hacer las peticiones al servidor.

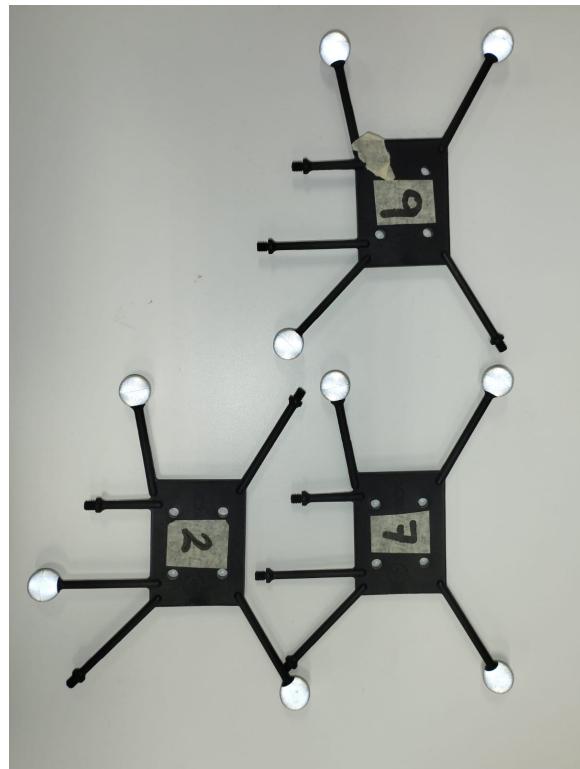
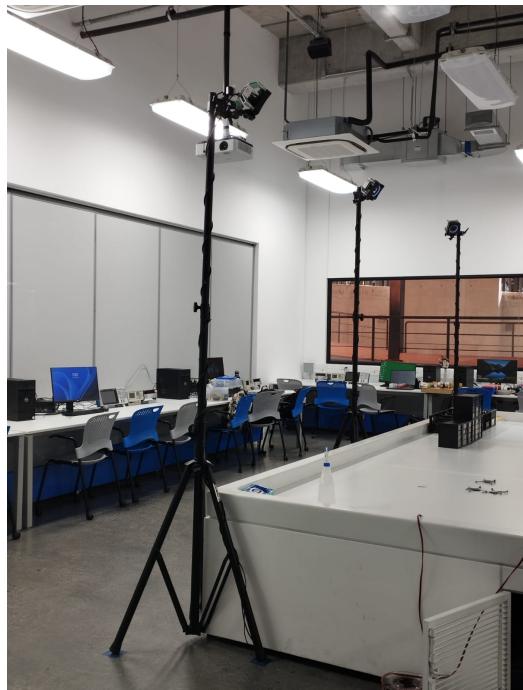


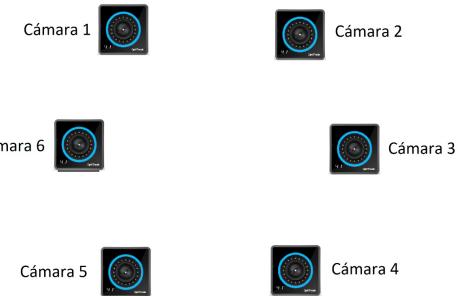
Figura 41: Marcadores 2,9 y 7

7.2.1. Montaje

Para poder montar el sistema, las camaras están colocadas en abrazaderas tipo Manfrotto direccionadas a 30 grados de inclinación y estas asimismo conectadas a trípodes cuya altura es de metros, posicionados en forma rectangular alrededor de la mesa de pruebas del laboratorio de robótica de la UVG el sistema esta completo.



(a) Trípode



(b) Posicionamiento camaras

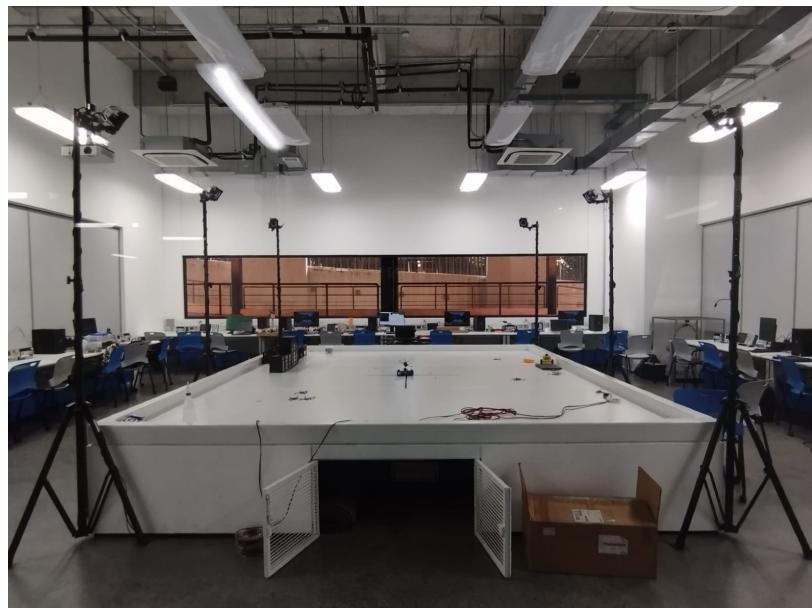
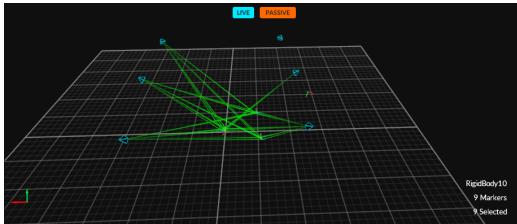


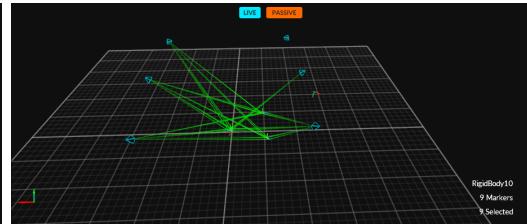
Figura 43: Montaje final alrededor de la mesa de pruebas.

7.3. Software

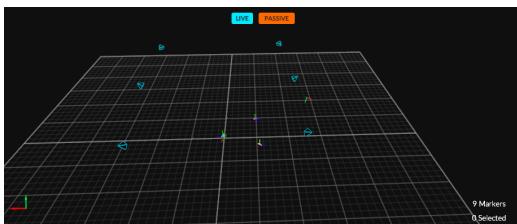
El software que utiliza el sistema OptiTrack es llamado Motive. Se encarga de interpretar la información obtenida con las camaras, este software viene en conjunto con un kit de desarrollo llamado NatNet. Este kit permite utilizar la información obtenida con el OptiTrack en otra aplicación como lo sería una red Wifi. El kit actúa como un cliente - servidor, en donde el servidor es el programa Motive y el cliente la aplicación creada para interpretar la información y comunicarla a donde se necesite [46]. A continuación se presentan unas imágenes que representan algunas de las opciones y forma de visualización dentro del sistema Motive.



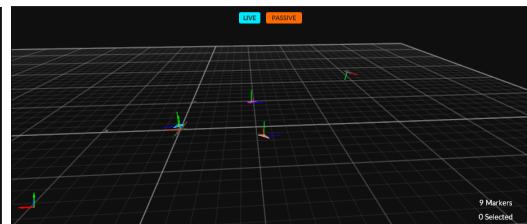
(a) Motive Iniciando



(b) Lineas de visualización OptiTrack.



(a) Cámaras del sistema OptiTrack.



(b) Visualización de Markers.

Utilizando este paquete de desarrollo y los códigos de ejemplo del mismo el MSc. Miguel Zea desarrollo un servidor en python para poder obtener los datos del OptiTrack, transformarlos y enviar los datos ya identificados a través de una comunicación TCP hacia Matlab, las peticiones posibles se pueden hacer en cuaternarios, ángulos de euler u otros, el código tiene un funcionamiento simple, descrito en el diagrama siguiente.

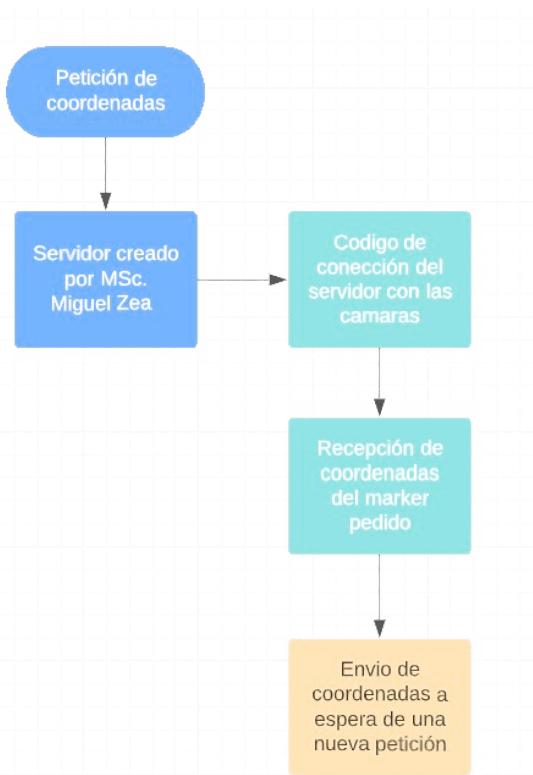


Figura 46: Diagrama de funcionamiento.

En la imagen siguiente hay un ejemplo de la recepción con R7 recibiendo las coordenadas en Euler y R10 recibiéndolas en cuaternarios.

```

R7 =
-0.6139  0.0025  0.6852 -0.1707 -0.7271 -0.5137

R10 =
-1.0825  0.0474 -0.3738  0.7076 -0.0080 -0.7063  0.0195

```

Figura 47: Recepción de coordenadas.

7.4. Comparación

El OptiTrack como nuevo dispositivo de rastreo de movimiento vino a mejorar la forma de obtener las coordenadas de objetos dentro de la mesa de pruebas del laboratorio de robótica de la UVG. Debido a esto la detección de coordenadas previamente hecha para otro trabajo de graduación se volvió obsoleta, por lo que se debían desarrollar nuevos códigos de obtención de coordenadas y actualizar algunos otros códigos que utilizaban este sistema y se usan en otros trabajos de graduación.

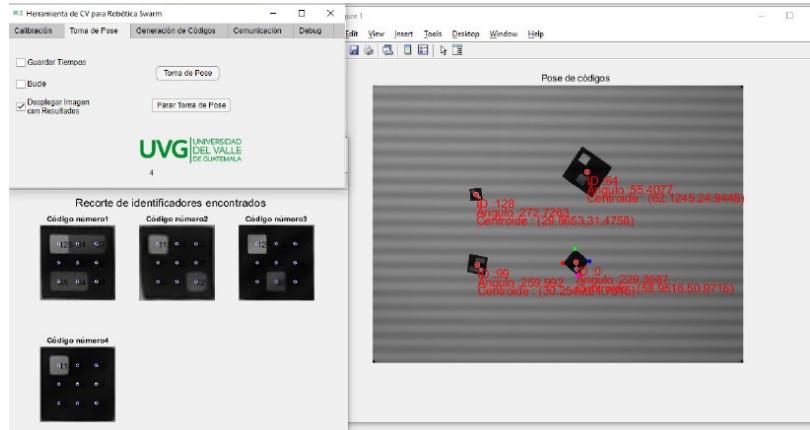


Figura 48: Detección de coordenadas anteriores[10].

El sistema Optitrack cuenta con una diferencia de +/- 0.2 mm en cuanto a posicionamiento y una diferencia de +/- 0.1 grados en direccionamiento por lo que tiene mejores características que el método anterior. Por otro lado este sistema es modular por lo que si en un futuro se amplía la mesa de pruebas, este sistema solo debe actualizar su posicionamiento y de ser necesario comprar mas camaras para continuar funcionando de manera eficiente.

7.5. Nuevo servidor para agentes móviles

El servidor creado por el Msc.Miguel Zea constaba de 2 partes, el servidor que obtiene las coordenadas proporcionadas por las camaras y una segunda parte que envía las coordenadas a la IP que haga la petición, este ultimo es un código desarrollado en matlab, lamentablemente la recepción de datos en Matlab nos dejaba con la necesidad de mandarlos de regreso a la plataforma móvil en donde se aloja el código PSO, por lo cual se tomó la decisión de optimizar el código de recepción utilizando Python ya que el movimiento de la plataforma se hace en este lenguaje y es mas fácil y rápido el traslado de información. Para verificar esto, se hicieron pruebas de velocidad con diferentes cantidades de markers, presentados en la siguiente tabla.

Data	Matlab		Python	
	Euler (s)	Cuat (s)	Euler (s)	Cuat (s)
100	11.295	13.30	8.65	9.44
1000	126.90	101.97	59.74	74.91

El algoritmo de movimiento del Bytebot están desarrollado en python, contrario a ello el algoritmo PSO esta desarrollado en C. Ya que estos 2 deben poder comunicarse entre ellos y con el nuevo servidor sin perder información, se opto por usar sockets conjunto al protocolo TCP. En el caso del Bytebot se utilizaron sockets normales, mientras que en el Bytebot3B fue una librería de intercambio de información llamada ZeroMQ.

CAPÍTULO 8

Plataformas móviles

En este capítulo se va a mostrar el diseño, modificaciones y verificación de funcionamiento de las plataformas móviles utilizadas.

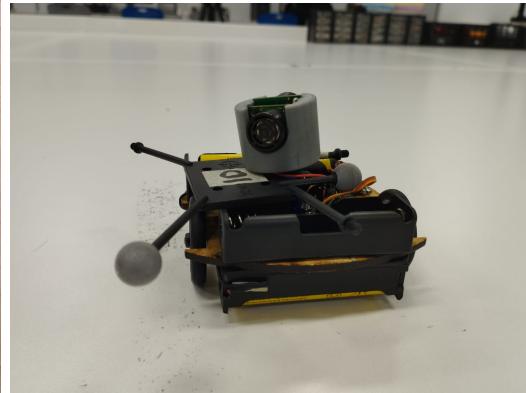
8.1. Diseño

8.1.1. Bytebot

Esta plataforma, como fue presentada en los antecedentes, es el resultado del trabajo de graduación de Julio Rodriguez [12] en ese trabajo se encuentran las especificaciones de esta plataforma. La plataforma contaba con distintos componentes útiles para este trabajo, sin embargo también tenía otros que no eran necesarios, por lo cual se quitaron para facilitar su uso, esta fue la única modificación que se hizo al diseño previo.



(a) Vista de arriba Bytebot sin módulos



(b) Vista de arriba Bytebot con módulos

8.1.2. Bytebot3B

Debido a que se necesitaba otra plataforma para poder hacer una comunicación de enjambre y una comparación tanto en la ejecución del algoritmo como en el movimiento y seguimiento de coordenadas per se, se desarrollo el Bytebot3B.

El Bytebot3B es una plataforma montada sobre una RaspberryPi 3B+ utilizando la misma idea de diseño que se creó para el Bytebot.

Para la parte del diseño físico se utilizaron las llantas y los motores de un kit de desarrollo que fue dado por el Departamento de electrónica de la universidad llamado MiniQ. El puente H utilizado en esta plataforma es el mismo que con el Bytebot debido a que la RaspberryPi 3B+ funciona con el mismo voltaje que la Raspberry Zero W y se utilizo la misma distribución del voltaje por medio de baterías. una ventaja de que el pinout de las diferentes versiones de una RaspberryPi no cambia, es que este quedo igual y no hubo que hacer modificaciones a esa parte. La conexión entre la RaspberryPi, el puente H y los motores quedo como se muestra en la siguiente imagen.

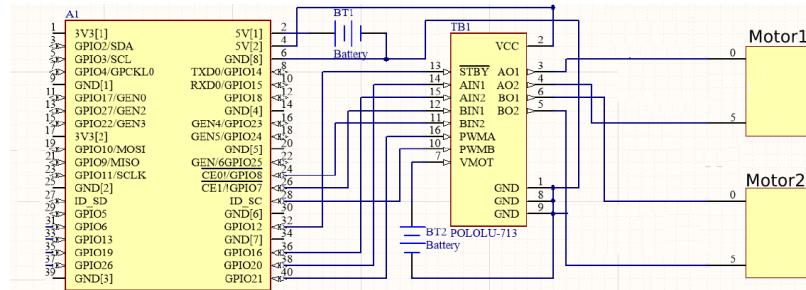


Figura 50: Conexión para Bytebot3B

En este caso, el Bytebot3B no está conectado a encoders como su predecesor sino que esta conectado directamente al puente H por lo que para las pruebas se moverá de forma mecánica, se espera luego modificar esta parte y utilizar encoders.

El diseño de la base de la plataforma fue hecho utilizando un diseño que julio dejó, este se modificó para que casara de manera correcta con la RaspberryPi 3B+ y sus módulos usb.

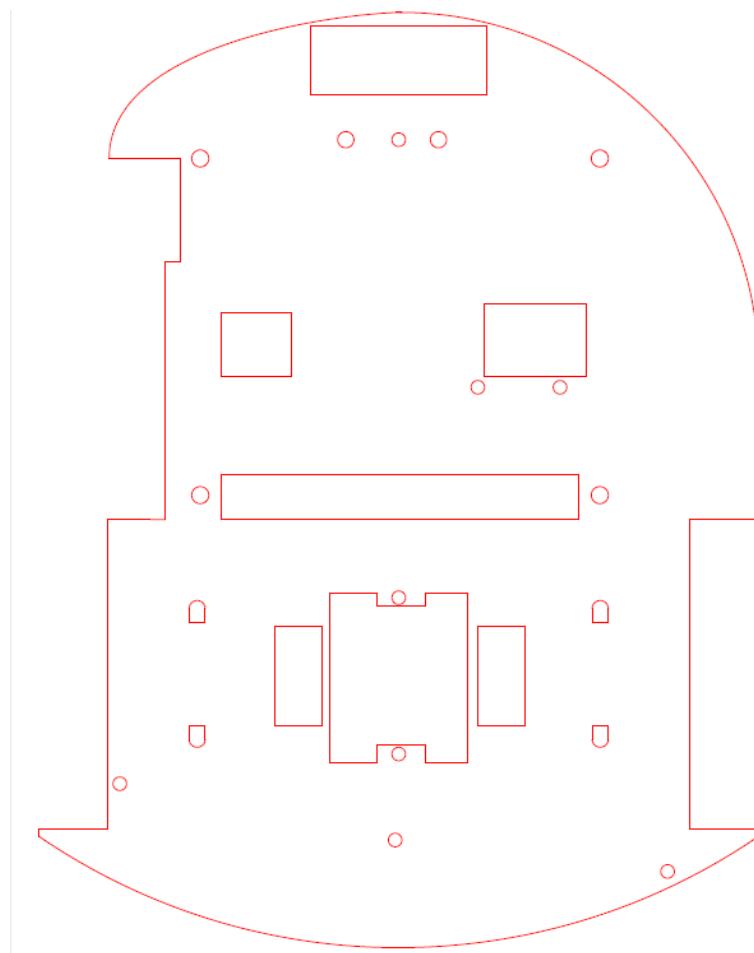


Figura 51: Diseño para corte Mdf

Finalmente el diseño del PCB que iba a conectar todo, fue modificado y actualizado para los nuevos requerimientos de la nueva plataforma.

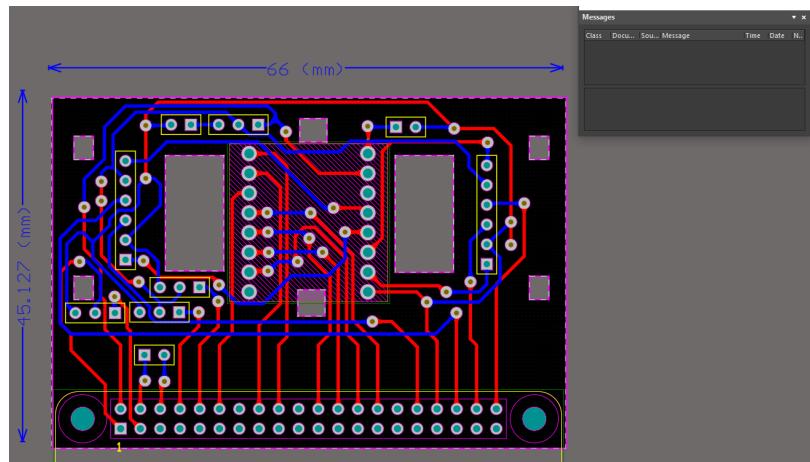


Figura 52: Diseño del PCB

Al terminar el diseño se observó que se necesitaba una caster ball pero que no era posible ponerla en la plataforma de mdf por lo cual se imprimió una pieza en 3D para este uso y con esta pieza, el robot quedó finalizado y listo para poder usar.

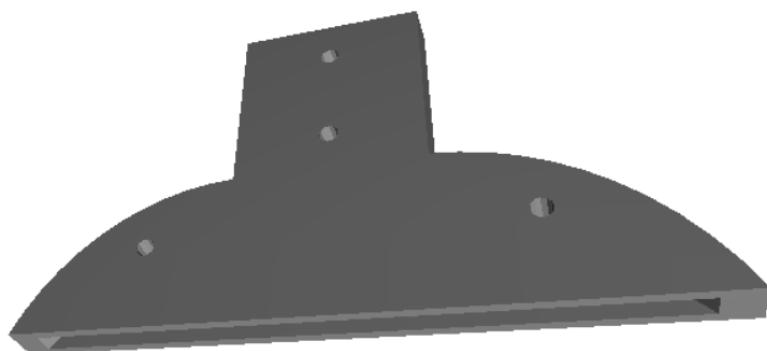
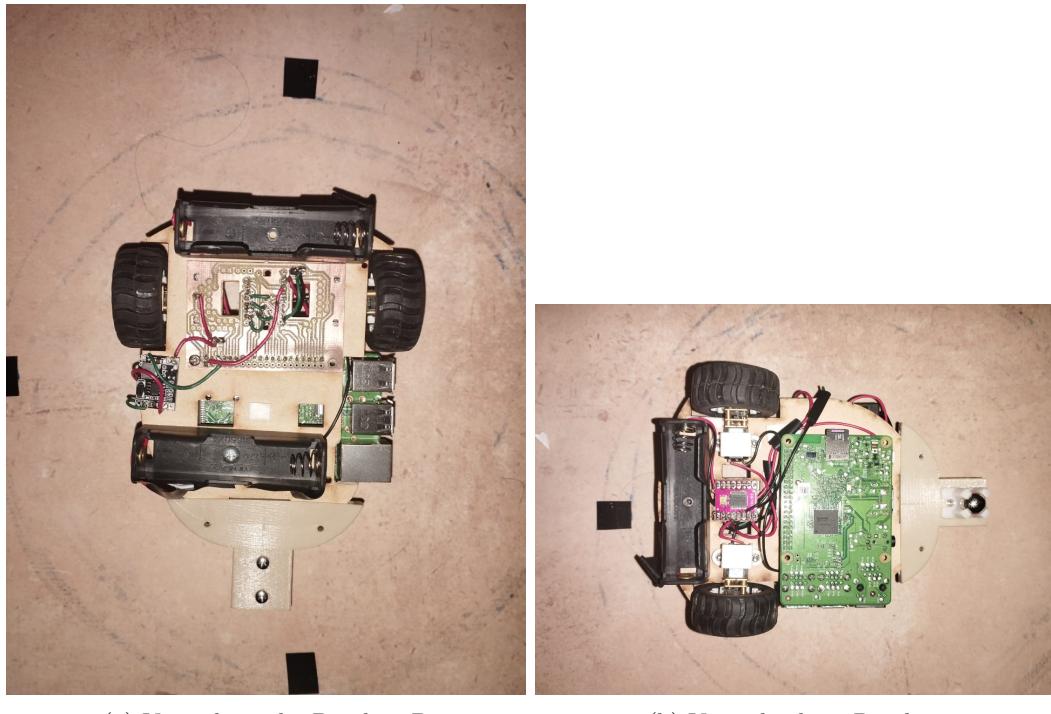


Figura 53: Diseño caster ball



(a) Vista de arriba Bytebot3B

(b) Vista de abajo Bytebot

8.2. Funcionamiento

Debido a que las 2 plataformas móviles funcionan con una RaspberryPi y están conectadas con los mismos puertos GPIO, es posible utilizar el mismo código de movimiento provisto por Julio. Este código fue creado en base a una librería llamada GPIOzero, tiene distintas modalidades como lo son

- Adelante
- Atras
- Giro Derecha
- Giro Izquierda

y muchas otras derivadas de estas. El funcionamiento con este código es un poco diferente para las plataformas ya que el Bytebot tiene la posibilidad de retro alimentarse con la información de los encoders para ver si se hizo el movimiento de manera correcta, pero el Bytebot3B no posee esta cualidad por el momento.

CAPÍTULO 9

Pruebas de algoritmo en plataformas móviles

En este capítulo se mostrará el desarrollo de como se generó la comunicación de las plataformas móviles con el servidor optitrack, el desarrollo del algoritmo, los datos obtenidos de este y la comunicación entre plataformas.

9.1. Comunicación de Optitrack con agentes

Como se menciono anteriormente el servidor creado en matlab fue sustituido por uno creado en python sin embargo las primeras pruebas se hicieron en este servidor para poder comprobar la correcta recepción de información, las siguientes imágenes representan las coordenadas mandadas a diferentes agentes diferenciados por su label, T1 y T2.

```
pi@raspi:~/Desktop/Codigos Anteiores AM/Codigo/C $ ./matlabserver
Mensaje recibido: T1,-0.61405,0.68532,-0.16494
Mensaje recibido: T1,-0.61405,0.68532,-0.17043
Mensaje recibido: T1,-0.61405,0.68532,-0.16138
Mensaje recibido: T1,-0.61406,0.68532,-0.16315
Mensaje recibido: T1,-0.61405,0.68532,-0.1613
Mensaje recibido: T1,-0.61406,0.68533,-0.16759
Mensaje recibido: T1,-0.61406,0.68533,-0.16224
Mensaje recibido: T1,-0.61405,0.68532,-0.16833
Mensaje recibido: T1,-0.61406,0.68533,-0.16416
Mensaje recibido: T1,-0.61405,0.68532,-0.16419
Mensaje recibido: T1,-0.61404,0.68533,-0.16206
Mensaje recibido: T1,-0.61405,0.68533,-0.16289
Mensaje recibido: T1,-0.61406,0.68533,-0.16633
Mensaje recibido: T1,-0.61406,0.68533,-0.16435
```

Figura 55: Recepción agente 1

```

pir@raspi:~/Desktop/Inteligencia-Computacional-y-Robotica-Swarm-2021-main/AM/Cod
igo/C $ ./Matlabserver
Mensaje recibido: T2,-0.87618,0.53406,0.70263
Mensaje recibido: T2,-0.87618,0.53408,0.70266
Mensaje recibido: T2,-0.87619,0.53407,0.70263
Mensaje recibido: T2,-0.87618,0.53407,0.70268
Mensaje recibido: T2,-0.87619,0.53406,0.70266
Mensaje recibido: T2,-0.87619,0.53408,0.70265
Mensaje recibido: T2,-0.87619,0.53407,0.70264
Mensaje recibido: T2,-0.87619,0.53408,0.70265
Mensaje recibido: T2,-0.8762,0.53407,0.70267

```

Figura 56: Recepción agente 2

Luego de verificar esto, se procedió a desarrollar el servidor de python, y a realizar las mismas pruebas con el mismo, sin embargo en este servidor cuando se obtienen los datos y luego se mandan hacia las 2 plataformas móviles, se hace mediante sockets, y cada una usa uno diferente, mientras que el Bytebot utiliza los sockets normales, el Bytebot3B utiliza la librería zmq. Por lo cual se utilizan hilos para poder enviar la información a sus respectivos lugares.

```

def Conexion_Robotat3B(agenteId):
    global Num;
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as tcp_obj:
        tcp_obj.connect((HOST, PORT))
        #print('Conectado')
        Num=0;
    while(1):
        print("Continuar?")
        X=int(input())
        if X==1:
            tcp_obj.sendall(json.dumps(agenteId).encode())
            #time.sleep(3)
            coordinates = tcp_obj.recv(1024).decode("utf-8")

            if(coordinates[1:19] =='onnection to the Ro'):
                print('Primer mensaje, obviarlo')
            else:
                #print(coordinates)
                Coo = coordinates.split(sep=', ')
                X=Coo[0]
                XX=X[1:19]
                #print("Cordenada X", XXX)
                Z=(Coo[2])
                #print("Cordenada Z", Z)
                Msg=XX[0:6]+','+Z[0:6];
                print(Msg)
                zero(Msg)
                #Sokt(Msg)
                time.sleep(1)
        else:
            print("terminado")

```

Figura 57: Muestra código de recepción y envío

9.2. Procesamiento de información

Al poder obtener las pruebas del movimiento de las plataformas, se procedió a hacer pruebas con el algoritmo desarrollado en el trabajo de graduación de Alex Maas [10]. El algoritmo tiene como nombre PSO, y mediante las plataformas se buscaba lograr un resultado parecido a las pruebas anteriormente realizadas en este trabajo.

Para poder proceder con la ejecución del algoritmo se hicieron pruebas en un simulador de códigos en C para poder entender el funcionamiento del mismo, y luego se hicieron diferentes versiones del código para cada agente. Estas modificaciones se hicieron al algoritmo debido a lo anteriormente mencionado sobre la diferente comunicación con sockets, se instalaron las librerías correspondientes y se hicieron las modificaciones pertinentes para el correcto funcionamiento. En las primeras versiones no se utilizó la comunicación entre agentes para comunicar sus posiciones y ajustar su valor global.

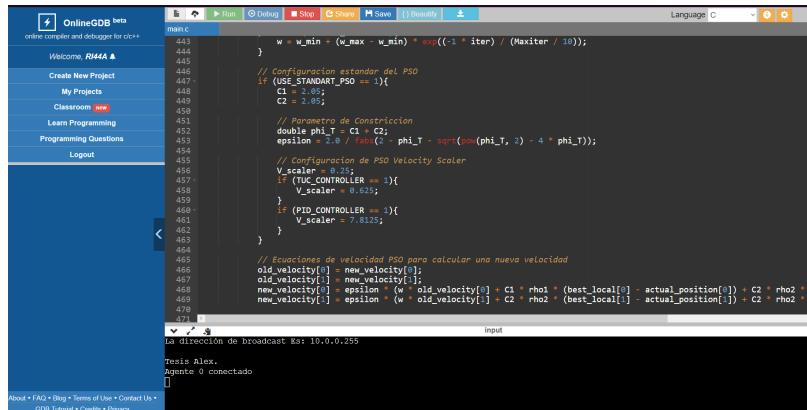


Figura 58: Ejemplo de simulación código C

Para poder proceder con la comunicación se probaron 2 flujos de envío y recepción de datos para decidir cual era la mejor manera de tratar con la información. Estos 2 flujos están descritos en los siguientes diagramas.

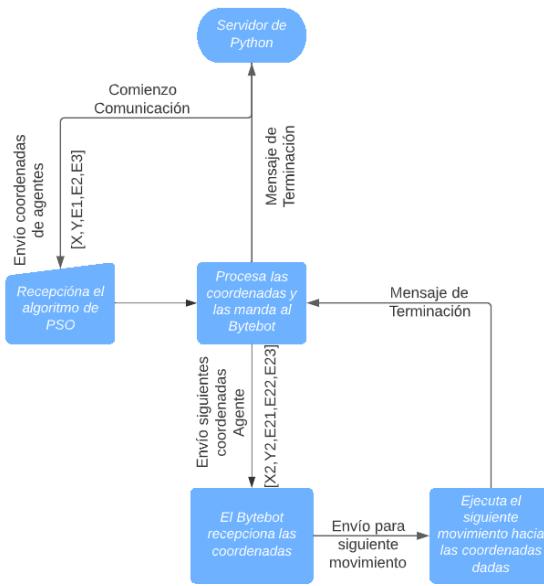


Figura 59: Diagrama de flujo 1

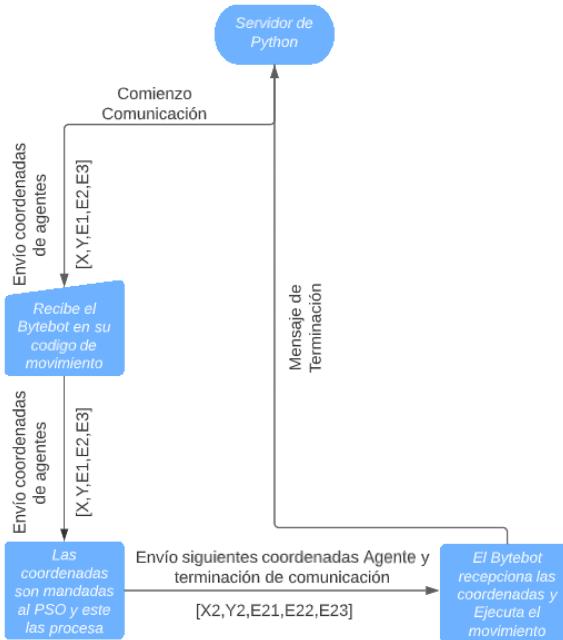
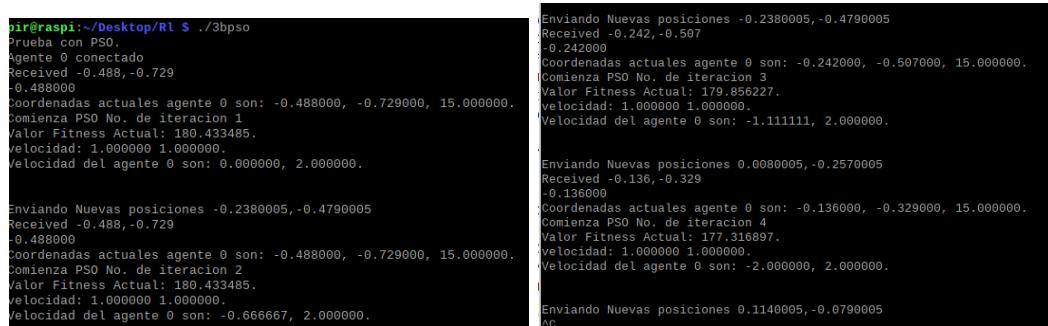


Figura 60: Diagrama de flujo 2

Debido a la facilidad de poder integrar diferentes funciones dentro de python, así como de enviar y recibir datos sin necesidad de comunicación por algún protocolo, solo intralenguaje, se decidió usar el diagrama 2.

9.3. Pruebas y Ejecución

Para el traslado de información se hicieron muchas pruebas hasta lograr exitosamente hacerlo de manera automática en ambas plataformas.



The image displays two terminal windows side-by-side. Both windows have a black background and white text. The left window, labeled '(a) Servidor ejecutando algoritmo PSO', shows the execution of a PSO algorithm. It starts with 'pir@raspi:~/Desktop/R1 \$./3bpso' followed by several lines of status output: 'Prueba con PSO.', 'Agente 0 conectado', 'Received -0.488,-0.729', 'coordenadas actuales agente 0 son: -0.488000, -0.729000, 15.000000.', 'Comienza PSO No. de iteracion 1', 'Valor Fitness Actual: 180.433485.', 'velocidad: 1.000000 1.000000.', 'Velocidad del agente 0 son: 0.000000, 2.000000.', 'Enviando Nuevas posiciones -0.2380005, -0.4790005', 'Received -0.242,-0.507', 'coordenadas actuales agente 0 son: -0.242000, -0.507000, 15.000000.', 'Comienza PSO No. de iteracion 3', 'Valor Fitness Actual: 179.856227.', 'velocidad: 1.000000 1.000000.', 'Velocidad del agente 0 son: -1.111111, 2.000000.', 'Enviando Nuevas posiciones 0.0080005, -0.2570005', 'Received -0.136,-0.329', 'coordenadas actuales agente 0 son: -0.136000, -0.329000, 15.000000.', 'Comienza PSO No. de iteracion 4', 'Valor Fitness Actual: 177.316897.', 'velocidad: 1.000000 1.000000.', 'Velocidad del agente 0 son: -2.000000, 2.000000.', 'Enviando Nuevas posiciones 0.1140005, -0.0790005', 'Received -0.114,-0.079'. The right window, labeled '(b) Servidor ejecutando movimiento', shows movement control commands. It includes 'Envmando Nuevas posiciones -0.2380005, -0.4790005', 'Received -0.242,-0.507', 'coordenadas actuales agente 0 son: -0.242000, -0.507000, 15.000000.', 'Comienza PSO No. de iteracion 3', 'Valor Fitness Actual: 179.856227.', 'velocidad: 1.000000 1.000000.', 'Velocidad del agente 0 son: -1.111111, 2.000000.', 'Enviando Nuevas posiciones 0.0080005, -0.2570005', 'Received -0.136,-0.329', 'coordenadas actuales agente 0 son: -0.136000, -0.329000, 15.000000.', 'Comienza PSO No. de iteracion 4', 'Valor Fitness Actual: 177.316897.', 'velocidad: 1.000000 1.000000.', 'Velocidad del agente 0 son: -2.000000, 2.000000.', 'Enviando Nuevas posiciones 0.1140005, -0.0790005', 'Received -0.114,-0.079'.

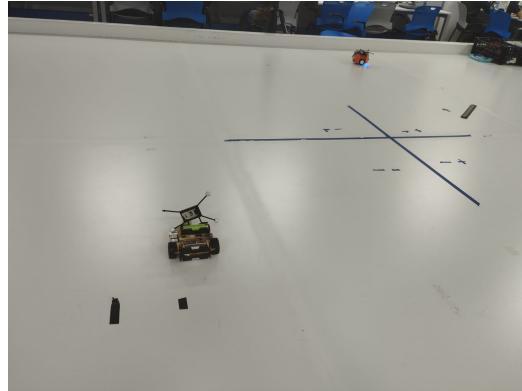
(a) Servidor ejecutando algoritmo PSO

(b) Servidor ejecutando movimiento

El Bytebot tenía un tiempo de procesamiento un poco menor al de su compañero y una mejor ejecución debido al uso de encoders. El movimiento por el que los 2 agentes se trasladan es definido por la posición actual y la posición siguiente en base a sumas y restas, dependiendo del resultado hace un movimiento hacia adelante o hacia atrás. Luego el robot hace un giro a la derecha de 45 grados y realiza la misma operación para el otro eje, luego hace un giro a la izquierda de 45 grados y regresa a su posición inicial viendo al eje X+. Para esta primera versión del movimiento el robot debe estar posicionado en dirección a X+. Se espera modificar este algoritmo para poder guardar la ultima posición y no hacer giros innecesarios. A continuación un ejemplo del movimiento de la plataforma Bytebot3B, con una distancia de movimiento frontal de 0.25m. Esta distancia se fue reduciendo en pruebas siguientes.



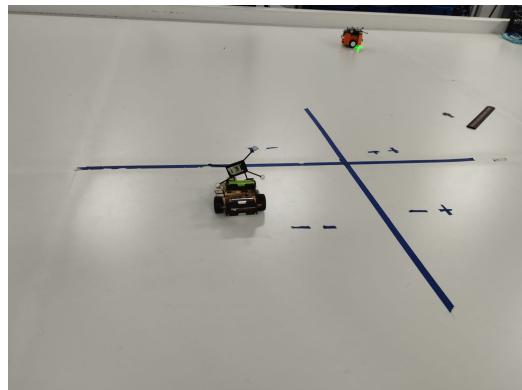
(a) Posición 1



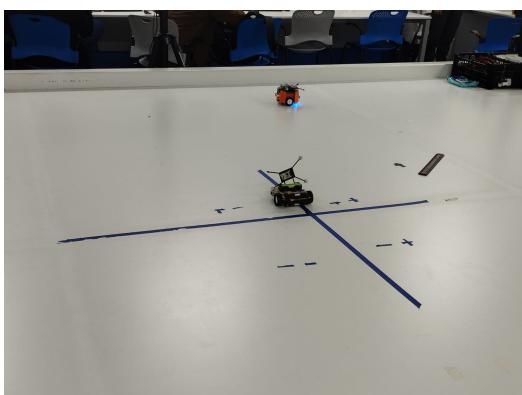
(b) Posición 2



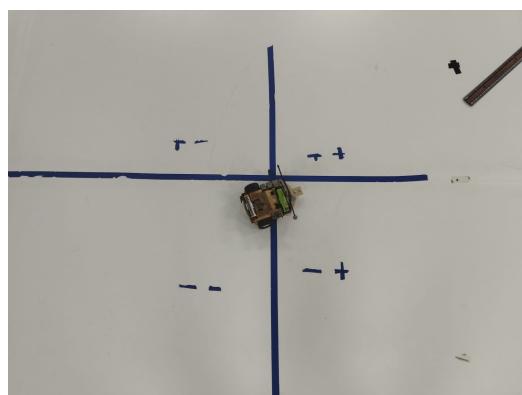
(a) Posición 3



(b) Posición 4



(a) Posición 5



(b) Posición 6

Esta prueba fue hecha con movimientos manuales utilizando el código de movimiento, con las direcciones dadas por el algoritmo debido a que a pesar de que el código automático de movimiento se ejecutaba, este presentaba ciertas diferencias con respecto al comportamiento dado mecánicamente y se atribuyó al no uso de encoders.

CAPÍTULO 10

Conclusiones

1. El algoritmo utilizado no estaba hecho para esta plataforma por lo que al obviar algunas opciones usadas anteriormente la viabilidad de comparación baja.
2. Gracias a las múltiples herramientas que ofrece la Raspberry la migración de algoritmos en leguajes como C y Python no supone ningún problema.
3. Las pruebas utilizando plataformas móviles fueron muy parecidas pero no iguales a las hechas anteriormente por lo que los resultados tienen cierto porcentaje de diferencia.
4. El uso de la caster ball es necesario para lograr un ángulo de 45 grados, sin este ángulo el movimiento se desequilibra y no se logra realizar el seguimiento de coordenadas a plenitud.
5. El diseño de un agente robotico mas largo y con un frente determinado mejoro la estabilidad y el giro para el correcto seguimiento de coordenadas.

CAPÍTULO 11

Recomendaciones

1. Agregar sensores a las plataformas móviles para poder visualizar los obstáculos en el perímetro
2. Hacer un código de comunicación intra agentes y central para poder observar la información en una base.
3. Realizar un mejor trazado de ángulos tanto en el OptiTrack como virtualmente en el robot para reducir el numero de iteraciones logrando hacer los movimientos como fueron pedidos exactamente.

CAPÍTULO 12

Bibliografía

- [1] M. Brambilla, «Swarm robotics: a review from the swarm engineering perspective,» *Swarm Intell.*, págs. 1-41, 2013.
- [2] W. Institute. «Programmable Robot Swarms.» (2014), dirección: <https://wyss.harvard.edu/technology/programmable-robot-swarms/>.
- [3] W. Institute. «A self-organizing thousand-robot swarm.» (2014), dirección: <https://wyss.harvard.edu/news/a-self-organizing-thousand-robot-swarm>.
- [4] L. B. / S. Communications. «Robotic swarm swims like a school of fish.» (2021), dirección: <https://wyss.harvard.edu/news/robotic-swarm-swims-like-a-school-of-fish>.
- [5] C. P. / S. Communications. «Robots to the rescue.» (2014), dirección: <https://news.harvard.edu/gazette/story/2014/02/robots-to-the-rescue-2/>.
- [6] M. J. Parsons. «OFFensive Swarm-Enabled Tactics (OFFSET).» (2016), dirección: <https://www.darpa.mil/program/offensive-swarm-enabled-tactics>.
- [7] O. Networks. «Defense and Intelligence Case Study – DARPA Offset Swarms for Raytheon BBN Technologies.» (2021), dirección: <https://www.oceusnetworks.com/defense-intelligence-case-study-darpa-offset-swarms-for-raytheon-bbn-technologies/>.
- [8] A. S. Aguilar, «Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO),» Tesis de licenciatura, Universidad del Valle de Guatemala, 2019.
- [9] E. Santizo, «Aprendizaje Reforzado y Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre,» Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.
- [10] A. D. Maas, «Implementación y Validación del Algoritmo de Robótica de Enjambre Particle Swarm Optimization en Sistemas Físicos,» Tesis de licenciatura, Universidad del Valle de Guatemala, ene. de 2022.
- [11] W. A. Sierra, «Implementación y Validación del Algoritmo de Robótica de Enjambre Ant Colony Optimization en Sistemas Físicos,» Tesis de licenciatura, Universidad del Valle de Guatemala, ene. de 2022.

- [12] J. A. Rodríguez, «Implementación y Validación del Algoritmo de Robótica de Enjambre Ant Colony Optimization en Sistemas Físicos,» Tesis de licenciatura, Universidad del Valle de Guatemala, 2022.
- [13] I. Navarro. «An Introduction to Swarm Robotics.» (2012), dirección: <https://www.hindawi.com/journals/isrn/2013/608164/>.
- [14] A. J. Sharkey y N. Sharkey, *The application of swarm intelligence to collective robots. Advances in applied artificial intelligence*, 1.^a ed. IGP, 2006.
- [15] E. Hunt, *Swarm robotics into the real world*, 2021.
- [16] Y. Tan y Z.-y. Z. Sharkey, *Research advance in swarm robotics*. 1.^a ed. DT, 2013.
- [17] S. M, *Swarm Robotic Behaviors and Current Applications*. 1.^a ed. Front Robot AI, 2020.
- [18] R. Solis-Ortega, *Informe técnico sobre enjambre de robots: Conceptualización, Simuladores y algoritmos de exploración*. jul. de 2018.
- [19] M. Kegeleirs, *Swarm SLAM: Challenges perspective*. 1.^a ed. Front Robot AI, 2021.
- [20] M. Brambilla, *Swarm Intell Swarm robotics: a review from the swarm engineering perspective*. 1.^a ed. HAL, 2013.
- [21] I. N. Oiza, *Swarm Robotics and formations of Robots*, 1.^a ed. Universidad Politécnica de Madrid, 2006.
- [22] C. Calderón-Arce, *Swarm Robotics: Simulators, Platforms and Applications Review*. 1.^a ed. MDPI, 2022.
- [23] A. R. Cheraghi y S. Shahzad, *Research advance in swarm robotics*. 1.^a ed. DT, 2013.
- [24] J. I. Vito Trianni y R. Haken, *The SAGA concept: Swarm Robotics for Agricultural Applications*. mayo de 2016.
- [25] M. Sangeetha y K. Srinivasan, «Swarm Robotics: A New Framework of Military Robots,» *Journal of Physics: Conference Series*, vol. 1717, n.^o 1, pág. 012017, ene. de 2021. DOI: 10.1088/1742-6596/1717/1/012017. dirección: <https://dx.doi.org/10.1088/1742-6596/1717/1/012017>.
- [26] J. LAW, X. WANG y M. L. XIN, «Microrobotic swarms for selective embolization,» *Science Advances Journal*, 2022.
- [27] M.-P.-I. für Intelligente Systeme, «Nature-inspired soft millirobot makes its way through enclosed spaces.,» *Physical Intelligence Journal*, 2018.
- [28] M. Chung y B. Lab. «5000 Robots merge to map de Universe in 3-D,» Berkeley Lab. (2018), dirección: <https://www.youtube.com/watch?v=g1LVMoxOKNc&t=9s>.
- [29] J. Dowell y G. B. Taylor, «The Swarm Telescope Concept,» *Journal of Astronomical Instrumentation*, 2018.
- [30] Y. H. Arnold R.D. y T. Tanaka, «Search and rescue with autonomous flying robots through behavior-based cooperative intelligence.,» *Int J Humanitarian Action*, 2018.
- [31] M. S., G. A. D. Caro. y T. S., «Swarm Intelligence and Cyber-Physical Systems: Concepts, Challenges and Future Trends,» *Journal Swarm and Evolutionary Computation*, 2021.
- [32] M. Abdelkader, S. Guler, H. Jalee y J. S. Shamma, «Aerial Swarms: Recent Applications and Challenges,» *Springer Naturel*, 2021.

- [33] P. Prakash. «Types of Mobile Robots – What to use where?» (2020), dirección: <https://addverb.com/types-of-mobile-robots-what-to-use-where/>.
- [34] Intel. «Tecnología y casos de uso de robots móviles autónomos.» (2020), dirección: <https://www.intel.es/content/www/es/es/robotics/autonomous-mobile-robots/overview.html>.
- [35] *Alphabot 2 User Manual*, 9 de abr. de 2022.
- [36] Adafruit. «Raspberry Pi.» (2013), dirección: <https://www.adafruit.com/category/105>.
- [37] ThePiHut. «Official Raspberry Pi Products.» (2016), dirección: <https://thepihut.com/collections/raspberry-pi>.
- [38] RaspberryPi. «Raspberry Pi OS.» (2018), dirección: <https://www.raspberrypi.com/software/>.
- [39] R. Pi. «Raspberry Pi Hardware.» (2019), dirección: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>.
- [40] J. Postel, «Transmission Control Protocol,» *Defense Advanced Research Projects Agency information Processing Techniques Office*, 1980.
- [41] K. A. Robbins y S. Robbins, «UNIX Systems Programming: Communication Concurrency and Threads,» *Sams*, 2001.
- [42] S. Walton, *Linux Socket Programming(1st edition.* mayo de 2003.
- [43] C. Duarte y C. J. Quiroga, *PSO Algorithm*, 1.^a ed. Ciudad Universitaria, Santander, Colombia, 2010.
- [44] A. P. Engelbrecht, *Computational Intelligence: an introduction*, 1.^a ed. Pretoria, Sudáfrica, 2008.
- [45] OptiTrack. «Primex 41.» (), dirección: <https://optitrack.com/cameras/primex-41/>.
- [46] OptiTrack. «Motive.» (), dirección: <https://optitrack.com/software/motive/>.

