UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación y Validación del Algoritmo de Robótica de Enjambre *Ant Colony Optimization* en Sistemas Físicos

Protocolo de trabajo de graduación presentado por Walter Andres Sierra Azurdia, estudiante de Ingeniería Mecatrónica

Guatemala,

2021

Resumen

En el siguiente protocolo se presenta una propuesta para trabajo de graduación enfocado en inteligencia y robótica de enjambre. En una tesis anterior se implementaron los algoritmos *Ant Colony Optimization* y *Ant System* a nivel de simulación. En este trabajo se implementarán y validarán los algoritmos en sistemas físicos.

Para lograr el objetivo, se migrarán los algoritmos desarrollados a un lenguaje de programación adecuado para los microcontroladores que se usarán en los sistemas físicos. Se realizarán pruebas simples en una mesa de pruebas. Finalmente, se compararán los resultados obtenidos en los sistemas físicos con los obtenidos en las simulaciones.

Antecedentes

Implementación de *Bug Algorithm* en un robot E-Puck

Anteriormente se han implementado algoritmos de seguimiento de trayectoria con obstáculos Por ejemplo, en [1] se describe la implementación en un robot E-Puck de un algoritmo para que este avanzara en una trayectoria mientras evadía obstáculos en el camino, este trabajo fue desarrollado por Departamento de Ingeniería en sistemas y Control del Instituto Universitario de Automática e Informática Industrial, en la Universidad Politécnica de Valencia.

Se modeló y simuló la cinemática del robot para la implementación del algoritmo, también se realizaron pruebas con el robot físico. Tanto en la simulación utilizando WeBots como en el robot real se obtuvo un buen desempeño, a pesar de los buenos resultados analizaron los efectos al ser expuestos a distintos factores ambientales y como estos afectaban el desempeño de los robots. Uno de los factores que más afecto en el desempeño del robot fue el color de los obstáculos, ya que el robot detectaba mejor los obstáculos que eran de color rojo. También se pudo observar que el módulo de odometría del robot era menos preciso cuando la distancia recorrida aumentaba o cuando el robot realizaba una cantidad considerable de rotaciones. Finalmente se pudo demostrar no solo la implementación del *Bug Algorithm* sino también la implementación de un sistema de control para mantener la trayectoria de una manera muy precisa [1].

Implementación de PSO en Robots Diferenciales

En la segunda fase del proyecto Robotat de la Universidad del Valle de Guatemala desarrollado por Aldo Aguilar [2], se utilizo como base el algoritmo clásico *Particle Swarm Optimization* y se modificó con el fin de poder implementarlo en robots diferenciales reales ya que sin esta modificación, los movimientos con el algoritmo estándar realizado las partículas (robots) son irregulares y como consecuencia se tendrá una saturación en los actuadores del robot.

El algoritmo modificado emplea un planeador el cual antes de seguir la posición exacta

dada por el algoritmo, utiliza un sistema de control el cual retraerá la trayectoria determinada y generara velocidades suaves y continuas para los robots diferenciales. Se realizaron distintas pruebas en entorno de simulación de *WebBots* en las cuales se determinó que el controlador con mejor desempeño es el *TUC-LQI*.

Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre

La tercera fase el proyecto Robotat desarrollado por Eduardo Santizo [3] se proponen dos soluciones para realizar mejoras al algoritmo PSO mediante el uso de técnicas de aprendizaje reforzado y profundo. La primera propuesta es una mejora al algoritmo por medio de redes neuronales recurrentes y la segunda propuesta fue una alternativa al algoritmo de navegación alrededor de un ambiente conocido por medio de programación dinámica.

Para poder mejorar el desempeño del PSO se empleo el *PSO Tuner* que consiste de una red neuronal recurrente que toma diferentes métricas propias de las partículas del PSO y a través de su procesamiento por medio de una red LSTM, GRU o BiLSTM las torna en predicciones de los parámetros que deberá utilizar el algoritmo. Luego de obtener los parámetros adecuados, el PSO Tuner fue capaz de reducir el tiempo de convergencia en mínimos locales, la red utilizada fue la BiLSTM ya que esta resulto ser la mejor opción de las tres propuestas.

Para la alternativa al algoritmo se utilizo como base la programación dinámica de *Grid-world*, sin embargo este fue modificado para lograr ajustar problemas con robots diferenciales. En primer lugar, se incremento el numero de acciones que puede tomar el robot, es decir que ahora el robot se podrá mover en una diagonal a 45^o. Luego se divide el espacio de trabajo en celdas y es escaneado para determinar las celdas obstáculo y meta. Por ultimo, mediante *Policy Iteration* se genera una acción óptima por estado, para generar una trayectoria a seguir por el controlador, tanto el PSO Tuner como el planificador de trayectorias fueron probados a nivel de simulación utilizando Matlab.

Implementación del ACO en robots diferenciales

En el trabajo desarrollado por Gabriela Iriarte [4], en el cual se implemento el algoritmo de inteligencia de enjambre *Ant System* o *Ant Colony* como planificador de trayectorias y poder tener otra alternativa al MPSO. Para poder comparar el desempeño de este algoritmo se utilizaron varios controladores y se determino cual de todos resulta ser mas efectivo, luego de realizar las pruebas, el controlador con la respuesta mas suave son el controlador de pose y el controlador de pose de Lyapunov. Tambien se realizaron pruebas con algoritmos genéticos para explorar otras alternativas al MPSO y el AC. El planificador de trayectorias fue realizado en Matlab y para realizar pruebas se realizo una simulación en Webots utilizando como robot el E-Puck.

El alcance tanto de este trabajo como el realizado por Eduardo Santizo [3], fue realizar la implementación de los algoritmos y realizar pruebas con dicho algoritmo todo a nivel de simulación utilizando Matlab y Webots como herramientas para hacer las pruebas, a pesar que las pruebas realizadas fueron exitosas y se demostró un buen desempeño de los

algoritmos, no se llegó a una implementación física en ninguno de los trabajos.

Justificación

Dentro de la rama de inteligencia computacional de enjambre, el *Ant Colony Optimization* (ACO) es uno de los algoritmos más utilizados. Este consiste en un método de optimización basado en el comportamiento de insectos para determinar el camino mas rápido hacia la meta.

En la fase anterior, se planteó el uso del algoritmo *Ant Colony* como un método alternativo al PSO en el proyecto Robotat de la Universidad del Valle de Guatemala. Este es utilizado para planificar la trayectoria utilizada por el robot para lograr llegar a la meta de la forma mas rápida. También se implementaron distintos controladores para tener una métrica de comparación entre dichos controladores y poder comparar el desempeño con el algoritmo PSO. Los resultados de este planificador de trayectorias fueron probados en un entorno de simulación en Matlab y Webots.

Dado el alcance de la fase anterior, en el cual se llegó a implementar el algoritmo exitosamente a nivel se simulación. En este trabajo se busca implementar el planificador de trayectorias en robots diferenciales físicos, para de esta forma poder someter al sistema a factores reales. También se busca validar el desempeño del algoritmo en los robots diferenciales móviles físicos al ser puestos a prueba en ambientes controlados utilizando el algoritmo *Ant Colony*.

Objetivos

Objetivo General

Implementar y validar los algoritmos de robótica en enjambre *Ant Colony Optimization (ACO)* y algunas variantes desarrollados en años anteriores a nivel de simulación, en sistemas físicos.

Objetivos Específicos

- Evaluar distintas opciones de microcontroladores, sistemas embebidos, lenguajes de programación y entornos de desarrollo, y seleccionar los más adecuados para su uso en aplicaciones de robótica de enjambre utilizando el algoritmo ACO.
- Migrar los algoritmos desarrollados anteriormente a los microcontroladores de los sistemas físicos seleccionados.
- Validar la migración de los algoritmos y verificar el desempeño de los sistemas físicos mediante pruebas simples en ambientes controlados.

Marco teórico

Ant Colony Optimization (ACO)

Inspiración

La inspiración de este algoritmo viene del comportamiento natural en una colonia de hormigas, al rededor de los cuarentas y cincuentas del siglo 20, el entomólogo Pierre-Paul Graseé observo en una colonia de termitas lo que el denomino como "Estímulos Significativos". En sus observaciones pudo ver los efectos de las reacciones tanto en el insecto que produjo como en la colonia y utilizó el termino de "estigmergía" para describir este tipo de comunicación en la cual los trabajadores son estimulados por el desempeño que logran. En estas observaciones se determinó que las hormigas poseen la habilidad de encontrar el camino más corto entre la fuente de alimento y su colonia mediante el uso de feromonas.

A partir de estas observaciones en 1992 Marco Dorigo desarrolló un algoritmo con base al comportamiento de las hormigas denominado *Ant System (AS)*, a partir de este algoritmo se desarrollaron muchos híbridos hasta que en 1999 Dorigo junto a Di Caro and Gambardella basandose en un método metaheurístico, en donde se ven las hormigas como base del algoritmo denominado ACO, para resolver problemas discretos de optimización [5].

Funcionamiento

Cuando una hormiga encuentra alimento, esta deja un rastro de feromonas en el camino a la colonia, el resto de compañeras detectan este rastro y lo siguen ya que estas saben que encontraran alimento al final del camino, mientras más hormigas pasan por el camino el rastro se hace cada vez más fuerte. Este es un método eficiente ya que aunque existan dos caminos diferentes, mientras más corto sea el camino, las hormigas que vienen detrás pueden detectar la feromonoas dejadas por sus compañeras más rápido, aumentando las posibilidades que escogan el camino mas corto. Este comportamiento se ve reflejado en el experimento de "doble puente" realizado por Deneuborg, donde se colocan 2 caminos para llegar a una fuente de alimento, en un principio los dos caminos tenían la misma distancia, al empezar el experimento el comportamiento de las hormigas fue aleatorio pero al final todas seleccionaron un solo camino, luego uno de los caminos se hizo mas corto que el otro, al igual que en el experimento anterior, el comportamiento de la colonia fue aleatorio, pero mientras pasaba el tiempo, los insectos escogieron el camino mas corto, a partir de este experimento se desarrolló un modelo dado por la Ecuación 8 [5].

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \tag{1}$$

Donde, m_1 y m_2 representan la cantidad de hormigas que han pasado por el camino 1 y el camino 2 respectivamente, k y h son parámetros que se ajustan mediante experimentación y p_1 representa la probabilidad que la hormiga escoga el camino 1.

Algoritmos desarrollados

Existen diversas variantes para el algoritmo ACO, entre las mas exitosas son el *Ant System*, cuya principal característica es que en cada iteración los valores de feromonas son actualizados mediante la cantidad de hormigas m que han construido una solución. El modelo que define la cantidad de feromoas τ_{ij} es el siguiente:

$$\tau_i j = (1-\rho) * \tau_i j + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

Donde, ρ es la taza de evaporación de la feromona, m es el número de hormigas y $\Delta\tau_{ij}^{k}$ es la cantidad de feromona que hay en las aristas.

$$\Delta\tau_{ij}^{k} = \left\{ \begin{array}{r} \dfrac{Q}{L_k},\ \text{k utilizo aristas (i,j)} \\ 0,\ \text{otro caso} \end{array} \right\} \tag{3}$$

Donde, Q es una constante y L_k es la longitud del recorrido construido por la hormiga k.

$$P_{i,j}^{k} = \left\{ \begin{array}{r} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum \tau_{il}^{\alpha}\eta_{il}^{\beta}}, \text{sic}_{ij} \in N(s^p) \\ 0,\ \text{otro caso} \end{array} \right\} \tag{4}$$

Donde, $N(s^p)$, es el set de componentes factibles, es decir las aristas (i,l) donde l son las aristas que no ha sido visitado por la hormiga k. Los parámetros α y β controlan la importancia de la feromona dada por la información heurística η_{ij} que esta dada por:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{5}$$

Donde, d_{ij} es la distancia entre la arista i y j [5]. .

Raspberry Pi

La Raspberry Pi es un ordenador pequeño y de bajo coste, al cual se puede conectar un monitor y un teclado para interactuar con ella como cualquier otra computadora. Fue desarrollado como una organización caritativa de la Fundación Raspberry Pi en 2009 cuyo objetivo era animar a los niños a aprender informática en las escuelas.

Con Raspberry Pi esto es mucho más sencillo y abre las puertas de la experimentación y el aprendizaje en distinetos ambitos. Puede usarlo para aprender a programar, crear proyectos de electrónica, y para muchas de las cosas que hace cualquier PC de escritorio, como hojas de cálculo, procesamiento de texto, navegar por Internet y jugar ciertos videojuegos.

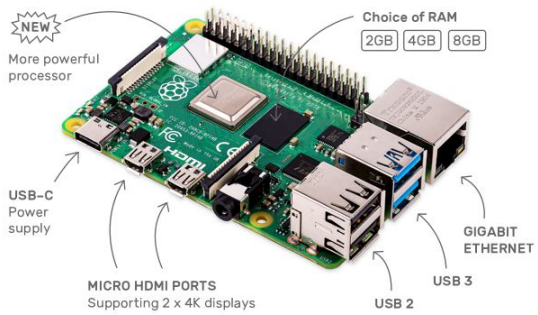


Figura 1: Raspberry Pi [6].

También reproduce videos de alta definición. La Raspberry Pi está siendo utilizada para aprender de programación y creación digital [6].

La Raspberry Pi es la placa de un ordenador simple compuesto por un SoC, CPU, memoria RAM, puertos de entrada y salida de audio y vídeo, conectividad de red, ranura SD para almacenamiento, reloj, una toma para la alimentación, conexiones para periféricos de bajo nivel, reloj. Se tiene que conectar conectar periféricos de entrada y salida para poder interactuar con la Raspberry, instrumentos como un monitor, un ratón y un teclado y grabar un sistema operativo para Raspberry Pi en la tarjeta SD.

Robots Móviles

Los robots móviles son una clase de robots con la capacidad de moverse en el entorno. Existe una variedad de robots móviles que pueden moverse en el suelo, sobre el agua, a través del aire o bajo del agua. Con estos ejemplos se destaca el diversidad de lo que se conoce como plataforma robótica.

A pesar de los distintos tipos, estos robots móviles son muy similares en términos de lo que hacen y cómo lo hacen. Una de las funciones más importantes de un robot móvil es moverse a de un punto inicial hasta un punto final. La meta podría especificarse en términos de alguna característica del entorno, por ejemplo se mueven hacia la luz, o en términos de alguna coordenada geométrica o referencia de mapa. En cualquier caso, el robot tomará algún camino para llegar a su destino y se enfrentará desafíos como obstáculos que pueden bloquear su camino [7].

E-Puck

El E-Puck es un robot diferencial móvil con llantas desarrollado por Dr. Fancesco Mondada y Michel Bonani en el 2006 en *EPFL, (Federal Swiss Institute of Technology in Lausanne.* El robot consiste de dos llantas con actuadores que permiten girar en ambas direcciones para

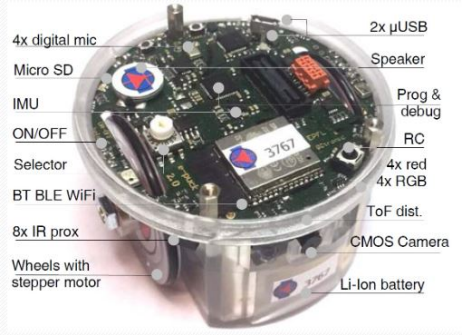


Figura 2: Sensores del robot E-Puck [8].

Figura 3: Robot Pi-puck [9]

cambiar la dirección del robot. Además tiene diversos sensores que permiten hacer mediciones muy buenas de distancia con objetos a su alrededor. El robot utiliza un procesador dsPIC que funciona como un microcontrolador [1].

Pi-puck

El Pi-puck es la plataforma del robot E-puck con la adaptación de la interfaz de la Raspberry desarrollado en la Universidad de York por (YRL) *(York Robotics Laboratory)* en conjunto con GCtronic. Es una extensión del robot E-puck y el E-puck 2 que permite montar una computadora de placa única Raspberry Pi Zero en el robot, agregando soporte para Linux, periféricos adicionales y mayores posibilidades de expansión [9].

Metodología

Para evaluar las distintas opciones de sistemas embebidos, el microcontrolador y lenguaje de programación a utilizar para la implementación del algoritmo, se propone:

- Investigar plataformas y lenguajes de programación para la implementación, esto para facilitarnos la tarea de conseguir las herramientas necesarias.
- Luego de investigar las distintas opciones que se encuentre, determinar el microcontrolador a utilizar.
- Con el microcontrolador escogido, determinar el lenguaje de programación a utilizar.
- Al escoger las herramientas a utilizar, se procederá a familiarizarse con el entorno de desarrollo escogido, realizando diversas pruebas para aprender a programar en la plataforma.

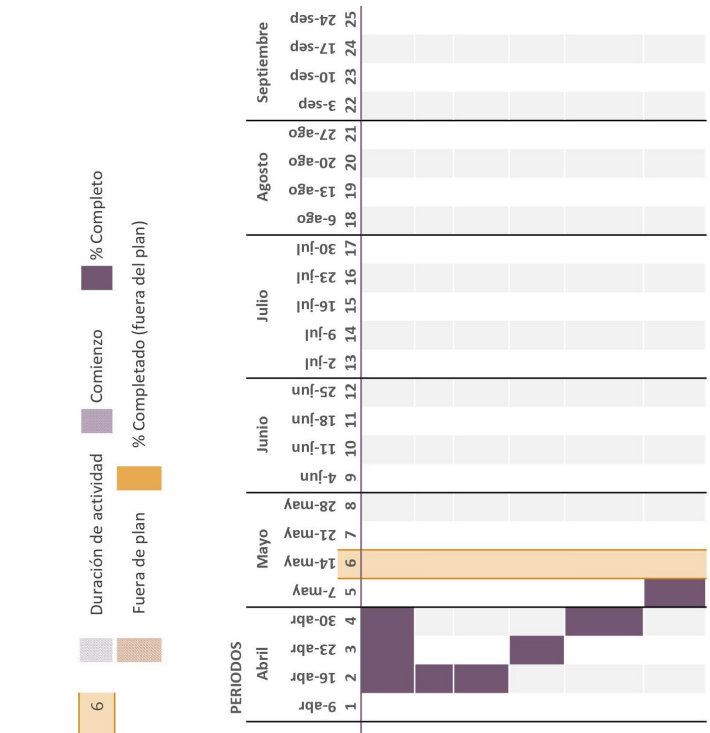
Para migrar el algoritmo ACO desarrollado anteriormente por Gabriela Iriarte [4], se propone:

- Sacar medidas de la mesa de pruebas para la implementación y ajustar el código ya hecho a las nuevas medidas.
- Correr las simulaciones con los ajustes hechos y verificar el funcionamiento.
- Comenzar implementando un algoritmo mas sencillo como lo es el *Simple Ant Colony Optimization*.
- Una vez se logre la migración del algoritmo simple, implementar el algoritmo *Ant System*.
- Al lograr migrar el algoritmo, se implementaran los diferentes métodos de control a utilizar ya que se planea utilizar varios para lograr un mejor rendimiento.

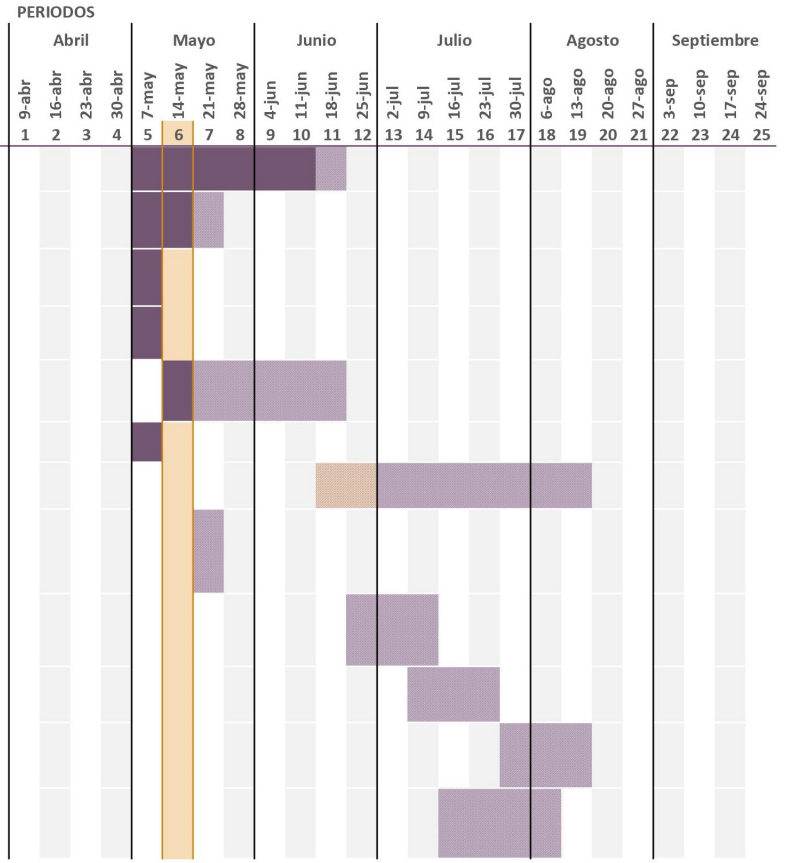
Para validar la migración de los algoritmos al controlador, se propone:

- Simular un método de comunicación entre los agentes u "hormigas" para empezar a probar el funcionamiento.
- Comenzar a realzar distintos experimentos en mesa de pruebas, como lograr llegar a una meta especifica o recibir la información de las "feromonas" y procesarla.
- Al tener todo funcionando, implementar comunicación entre dos o mas controladores y que estos envíen y reciban información exitosamente.

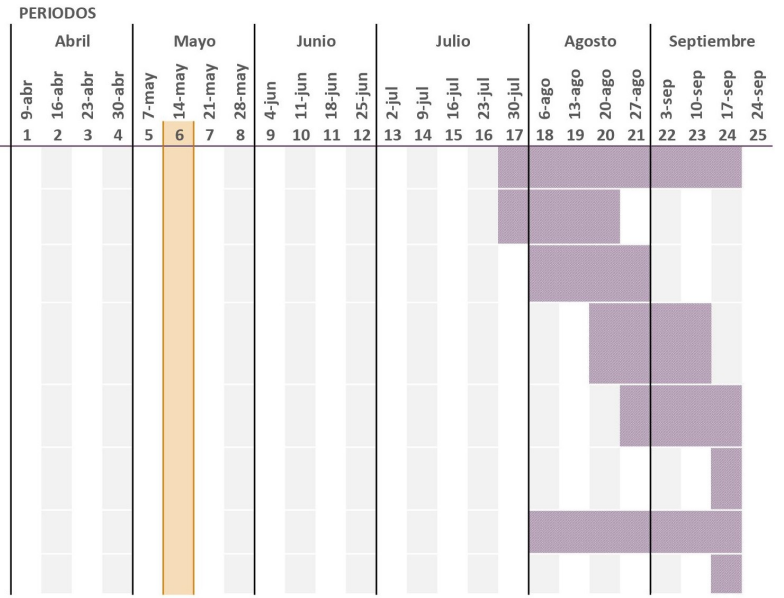
Cronograma de actividades



	ACTIVIDAD	INICIO	DURACIÓN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
2	Herramienta a utilizar	5	7	5	7	90%
2.2	Investigar plataformas y lenguajes accesibles	5	3	5	3	80%
2.2	Determinar el controlador a utilizar	5	1	5	1	100%
2.3	Determinar el lenguaje de programación	5	1	5	1	100%
2.4	Familiarizarse con el entorno escogido	6	6	6	6	20%
2.5	Realizar una cotización.	5	1	5	1	100%
3	Migrar el algoritmo ACO	13	7	11	9	0%
3.1	Sacar medidas de la mesa de pruebas para la implementación	7	1	7	1	0%
3.2	Implementar el Simple Ant Colony Optimization	12	3	12	3	0%
3.3	Implementar el Ant System	14	3	14	3	0%
3.4	Perfeccionar los algoritmos	17	3	17	3	0%
3.5	Implementar los diferentes métodos de control	15	4	15	4	0%



		ACTIVIDAD	INICIO	DURACIÓN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
4		**Validar los resultados**	17	8	17	8	0%
	4.1	**Simular el método de comunicación**	17	4	17	4	0%
	4.2	**Implementar en mesa de pruebas.**	18	4	18	4	0%
	4.3	**Implementar comunicación entre dos controladores**	20	4	20	4	0%
	4.4	**Determinar los mejores métodos de control**	21	4	21	4	0%
5		**Evaluar implementación de algoritmos genéticos**	24	1	24	1	0%
6		**Redacción de tesis**	18	7	18	7	0%
7		**Entrega de tesis**	24	1	24	1	0%



Índice preliminar

Referencias

[1] L. Marín, M. Vallés, Á. Valera y P. Albertos, "Implementation of a bug algorithm in the e-puck from a hybrid control viewpoint," *2010 15th International Conference on Methods and Models in Automation and Robotics, Miedzyzdroje, Poland*, págs. 174-179, 2010.

[2] A. Aguilar, "Algoritmo Modificado de Optimización de Enjambre de Particulas (MPSO)," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[3] E. Santizo, "Aprendizaje Reforzado y Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2020.

[4] G. Iriarte, "Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2020.

[5] M. Dorigo, M. Birattari y T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, n.º 4, doi: 10.1109/MCI.2006.329691. Págs. 28-39, 2006.

[6] Raspberry Pi Foundation, *Raspberry Pi documentation*, `https://www.raspberrypi.org/documentation/faqs/#introduction`, 2020.

[7] P. Corke, *Robotics, Vision and Control*. Berlin, Heidelberg: Springer, ISBN: 978-3-642-20144-8, 2017.

[8] GCtronic, *E-Puck Education Robot*, `http://www.e-puck.org/`, Accessed: 2018-02-23, 2018.

[9] University of York, *Pi-puck documentation*, `https://pi-puck.readthedocs.io/en/latest/`, 2020.