METODOLOGÍA PARA EL PLANEAMIENTO AUTOMÁTICO DE RUTAS EN VEHÍCULOS AÉREOS NO TRIPULADOS USANDO ALGORITMOS BIO-INSPIRADOS SOBRE SISTEMAS EMBEBIDOS

Germán David Góez Sánchez

INSTITUTO TECNOLÓGICO METROPOLITANO-ITM
MAESTRÍA EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL
FACULTAD DE INGENIERÍAS
MEDELLÍN, COLOMBIA
MAYO, 2016

Metodología para el planeamiento automático de rutas en vehículos aéreos no tripulados usando algoritmos bio-inspirados sobre sistemas embebidos

Germán David Góez Sánchez

Trabajo de investigación para optar al título de

Magíster en Automatización y Control Industrial

Directores

JORGE ALBERTO JARAMILLO GARZÓN, PhD
RICARDO ANDRÉS VELASQUEZ, PhD
RAMÓN FERNANDO COLMENARES, PhD

Línea de Investigación:

Sistemas de control y robótica

Grupo de Investigación:

Automática, Electrónica y Ciencias Computacionales

INSTITUTO TECNOLÓGICO METROPOLITANO-ITM
MAESTRÍA EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL
FACULTAD DE INGENIERÍAS
MEDELLÍN, COLOMBIA

MAYO, 2016

Dedico este gran logro a quienes con su confianza y apoyo sin escatimar esfuerzo alguno, me hanayudado a ser una persona de provecho,compartiendo tristezas, alegrías, éxitos y fracasos.

Agradecimientos

Al Dr. Jorge Alberto Jaramillo Garzón, profesor del Instituto Tecnológico Metropolitano, por su valioso aporte académico como director de esta tesis de maestría, en especial por su dedicación desinteresada en miras de llevar a buen término este trabajo de investigación.

Al Dr. Ricardo Andrés Velásquez Vélez, profesor de la Universidad de Antioquia, persona crítica, que por su valioso aporte académico como co-director de esta tesis de maestría contribuyo a alcanzar las metas propuestas en este trabajo de investigación.

Al Dr. Ramón Fernando Colmenares Quintero, profesor de la Universidad Cooperativa de Colombia, que como co-director de la tesis de maestría,puso todo su conocimiento y experiencia en Ingeniería Aeroespacial al servicio de este trabajo de investigación.

A los docentes de Maestría en Automatización y ControlIndustrial del ITM, por la experiencia y conocimientos transmitidos, así como por su compromiso y esfuerzoen miras de formar magister de calidad.

Al Instituto Tecnológico Metropolitano - (ITM), por brindarme la posibilidad de formarme como profesional y en esta ocasión, como magister en Automatización y ControlIndustrial. Sin la valiosa contribución de esta institución, no tendría las posibilidades a nivel profesional que tengo ahora.

Resumen

Este trabajo propone un método para el planeamiento de rutas on-line usando técnicas de optimización bio-inspirada con el fin de ser implementado sobre vehículos aéreos no tripulados, el cualplanee la ruta sin necesidad de intervención de un controlador en tierra. En el desarrollo del trabajo se evaluaron los optimizadores enjambre de partículas (PSO) y búsqueda Cuckoo (Ck) junto con el planificador de ruta propuesto. La evaluación del método consistió en ejecutar el planificador de ruta usando como base la técnica de optimización enjambre de partículas, comparándola con la ejecución del método usando la optimización por búsqueda Cuckoo. Los experimentos se realizaron sobre cinco rutas diferentes, cinco veces por ruta, lo que permitió comprobar la funcionalidad del método. Los resultados mostraron que el planificador de ruta tuvo mejores resultados al recorrer una menor distancia cuando su optimizador fue el PSO mostrando ser más estable, encontrando la ruta más fácilmente. En cuanto a tiempo de ejecución, los resultados mostraron que el planificador de ruta necesito un menor tiempo de ejecución cuando se usó el optimizador Ck, siendo de un 89% menor que la misma ejecución del planificador con PSO. Además se implementó el planificador de ruta sobre tres sistemas embebidos basados en microcontroladores, con arquitecturas ARM Cortex M0+, M4 y ColdFire V1de la familia Flexis de 32-bits. Los resultados comprueban que el método propuesto es lo suficientemente versátil y con baja demanda computacionalrealizando el planeamiento de ruta sobre los microcontroladores, y en el caso del ARM Cortex M4, entregando nuevos puntos de avance en la ruta en un tiempo inferior a seis segundos.Demostrando que la implementación de este tipo de metodologías es completamente viable, permitiendoque aeronaves no tripuladas naveguen de forma segura sobre zonas desconocidas, respetando las aerovías comerciales, obteniendo el máximo provecho al espacio aéreo.

Palabras clave: Meta heurística, Optimización por enjambre de partículas, Optimización por búsqueda, Cuckoo, Funciones de optimización, vehículo aéreo no tripulado, planeamiento en tiempo real.

Abstract

This work proposes a method to plan online routes using bio inspired optimization techniques to be implemented on unmanned aerial vehicles. The aim of this method is to plan the route without the need for a ground controller. Particle swarm optimization (PSO) and Cuckoo search (CK), together with the proposed route planner, were evaluated. The method executed the route planner using the PSO technique and compared it to the method using Cuckoo search optimization. Experiments were performed on 5 different routes, 5 times per route, allowing the performance verification of this method. Results showed that the route planner using PSO had better results when going a shorter distance, it was more stable and it could easily find the route. As per the execution time, results showed that the route planner required a shorter execution time using the CK optimizer, 89% less than the same execution using the PSO. Besides, the route planner was implemented on three embedded systems based on micro controllers, with ARM Cortex M0+, M4 and ColdFire V1 (Flexis 32-bits) architectures. Results prove that the proposed method is versatile enough and has low computational demand when planning the route on the microcontrollers. For the ARM Cortex M4, it delivers new advance points on the route in a time shorter than 6 seconds. This work shows the viability of implementing this type of methodology, allowing unmanned aerial vehicles to safely navigate on unknown areas, respecting commercial airways and obtaining maximum benefit from the aerial space..

Keywords: Metaheuristic, Particle Swarm Optimization, Cuckoo search algorithm, Functions Optimization, Unmanned Aerial Vehicle (UAV), real-time path planning.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

1.Introducción

Un sistema aéreo es una red de sistemas interconectados que abarcan el recurso humano, los procedimientos y certificaciones de operación, las rutas aéreas, las áreas protegidas con exclusión aérea y todos los equipos que intervienen para proporcionar servicios seguros de aeronavegación.Las rutas comúnmente denominadas como aerovía, son rutas nominales que tienen un área de protección que permite a una aeronavevolar con seguridad tanto en trayecto como en altura. Además, la autoridad aérea establece el espacio aéreo controlado y no controlado según la extensión de los sistemas. Es decir los servicios de tránsito aéreo pueden prohibir total o parcialmente los vuelos VFR (Visual Flight Rules), las cualesestablecen las condiciones suficientes para que el piloto pueda dirigir su aeronave, navegar y mantener la separación de seguridad con cualquier obstáculo con la única ayuda de la observación (AIP COLOMBIA ENR, 2014) (Civil, 2009).

La Organización Internacional de Aeronáutica Civil, (por sus siglas en inglés, International Civil Aviation Organization o ICAO), es la agencia encargada de promover normas y regulaciones para la aviación civil (International Civil Aviation Organization, 2006). Dichas normas y regulaciones son apropiadas por entidades gubernamentales que se encargan de establecer la normativa local y establecer las aerovías que tienen un área de protección que permite a una aeronave volar con seguridad tanto en trayecto como en altura. Para el caso de Colombia, la Aeronáutica Civil es la encargada de apropiar las recomendaciones de la ICAO (Aeronáutica Civil Colombiana, 2015).

La autoridad aeronáutica también define la operación de aeronaves remotamente tripuladas, cito textualmente: “En áreas diferentes a las zonas restringidas, se autoriza la operación de Aeronaves no tripuladas o Aeronaves Remotamente Tripuladas (ART) solamente propiedad del Estado Colombiano, para que operen en misiones de seguridad Nacional. Cualquiera sea la entidad u operador de estas aeronaves, la misma deberá contar con el aval y AIS COLOMBIA AIP COLOMBIA coordinación de la autoridad de Aviación de Estado, Fuerza Aérea Colombiana cuando se efectúen operaciones en

cualquier categoría de espacio aéreo. Con el fin de garantizar el cumplimiento de las correspondientes medidas de seguridad las aeronaves deberán tener la debida autorización y coordinación de la Autoridad de aviación de Estado y la Autoridad de aviación Civil, Aero civil*"* (Aeronáutica Civil Colombiana, 2015).

Las aeronaves remotamente tripuladas (ART), también conocidas como vehículo aéreo no tripulado (UAV, por sus siglas en inglés), es una aeronave pilotada de forma remota,de la cual,su uso se ha incrementado notablemente en los últimos años. Inicialmente de un carácter estrictamente militar, los UAVs han incursionado en el mundo civil en innumerables aplicaciones (Darío, Tarazona, & Lopera, 2014). En un futuro cercano cientos o miles de UAVs compartirán el espacio aéreo con la aviación convencional, lo cual los convierte en un nuevo factor de riesgo a la seguridad aérea (Valavanis, Oh, & Piegl, 2009). Esta nueva problemática justifica la necesidad de investigar y desarrollar sistemas inteligentes,capaces de optimizar y actualizar la ruta de vuelo con el fin de evitar colisiones,no solo con edificaciones y montañas, sino también con otros vehículos aéreos, sean o no tripulados.

Una ruta de vuelo es un conjunto de coordenadas geográficas que debe seguir una aeronave para ir de un punto de partida a un punto destino. El algoritmo de planeamiento de ruta es el encargado de encontrar dichos puntos entre cada coordenadaconsiderando una serie de restricciones impuestas por el entorno o por la misión (Peng , Li, 2012) (Meng & Xin, 2010). Los algoritmos para el planeamiento de ruta se dividen en dos grupos, los que realizan el planeamiento en línea y los que lo realizan fuera de línea (Xue, Cheng, & Cheng, 2014). El planeamiento de la ruta fuera de línea es una técnica propia de la aviación convencional, en la cual ya existen rutas preestablecidas (Aeronáutica Civil Colombiana, 2015; International Civil Aviation Organization, 2006), donde la corrección de rumbo depende directamente de los controladores de vuelo en tierra y se transmite a las aeronaves(Jardin, 2003). El planeamiento de ruta en la aviación convencional se realiza en función de optimizar el consumo de combustible, las emisiones de CO_2 y la contaminación por ruido, mejorando el costo de operación de la aeronave en general, así como los costos de mantenimiento. En este sentido se encuentran trabajos que buscan generar nuevas técnicas y algoritmos de simulación que

permiten evaluar el comportamiento de una aeronave mediante el levantamiento de un perfil de vuelo en crucero (Berton & Guynn, 2012; Colmenares Quintero et al., 2010).

El planeamiento de ruta en línea viene de la necesidad de dotar a los UAVs con la capacidad de corregir su trayectoria en función de los obstáculos detectados. Es decir, los UAVs deben tener la capacidad de determinar la trayectoria y corregir su rumbo a lo largo de una ruta de referencia (john tisdale, zuwhan kim, 2009). Es por eso que el problema general del planeamiento de ruta en línea son los ambientes dinámicos, donde el entorno es parcial o totalmente desconocido.

1.1 Objeto de estudio

Los vehículos aéreos no tripulados (UAV) más comúnmente conocidos como drones, son sistemas controlados por operadores en tierra, los cuales son totalmente dependientes desistemas de comunicaciones que les sirvan de enlace con el controlador en tierra. Dentro de los UAVs, encontramos plataformas de vuelo para aficionados, las cuales necesitan línea de visión entre el operador y el UAV para su control. También encontramos sistemas más robustos como los drones usados por las fuerzas militares de algunos países cuyo uso se centra en la vigilancia de fronteras, monitoreo de desastres naturales y, en algunos casos, con capacidad de portar armamento para ser usado en labores ofensivas en zonas de guerra.

El uso de drones es nuevo, mostrando un gran desarrollo en los últimos años. En el caso de Colombia su desarrollo ha estado ligado a situaciones excepcionales de seguridad que hace necesaria la implementación de estos equipos; un ejemplo, es la vigilancia de oleoductos que, según afirmala viceministra de defensa en entrevista realizada por el portal web (Diario Crítico Colombia, 2012), “Colombia en la actualidad posee unos 10.000 km de oleoductos, los cuales se encuentra bajo constantes ataques terroristas” y para lo cual hay designados, según la misma publicación,unos cinco mil hombres del ejército de Colombia para su vigilancia.

El ejemplo citado anteriormente demuestra la necesidad de desarrollar plataformas aéreas autónomas o semiautónomas que permitan cubrir de forma eficiente el espacio a

proteger, por lo que se hace necesario dotar a los UAVs con sistemas inteligentes que les permita realizar el planeamiento de la ruta en tiempo real, sin depender totalmente de un controlador en tierra, y que tenga la capacidad de planear la ruta por si solo si se pierde la comunicación con el operador. A su vez, el planeamiento y optimización de rutas en línea esboza inconvenientes para su implementación en UAVs debido a la robustez que se requiere en los equipos de procesamiento.

1.2 Descripción del problema

El dotar a un UAV con la capacidad de realizar el planeamiento de ruta de forma automática está directamente relacionado con el costo computacional y la energía requerida por el algoritmo para el planeamiento de ruta en tiempo real. Por esta razón, un factor relevante en el planeamiento de rutaes la necesidad de poseer un hardware que soporte procesamiento en tiempo real, es decir, que pueda entregar una solución en un tiempo de respuesta adecuado para que la aeronave pueda corregir la ruta dependiendo de la trayectoria que esté siguiendo. Tomando el planeamiento de rutas como un proceso demandante en tiempo de ejecución y energía (john tisdale, zuwhan kim, 2009).

El problema que se abordará en esta tesis, gira alrededor de los inconvenientes que se tienen que superar para dotar a un vehículo aéreo no tripulado con la capacidad de realizar el planeamiento de trayectorias en tiempo real usando técnicas de inteligencia artificial. La hipótesis de este trabajo, planteó que el uso de algoritmos de optimización bio-inspirados qué realicenel planeamiento automático de rutas, los cuales deben ser lo suficientemente livianos para ser implementados en un sistema embebido con capacidad de computo reducida y bajo consumo de energía.

1.3 Objetivo

Objetivo general

Proponer una metodología para el planeamiento de rutas en tiempo real, para vehículos aéreos no tripulados, mediante el uso de algoritmos de optimización bio-inspirados ejecutados sobre hardware con prestaciones reducidas.

Objetivos específicos

1. Caracterizar las técnicas de optimización basadas en algoritmos bio-inspirados, para el planeamiento de trayectorias en vehículos aéreos no tripulados, con el fin de proponer un algoritmo con baja demanda computacional que se ejecute sobre hardware con recursos limitados.

2. Establecer la demanda computacional y de consumo energético del algoritmo propuesto para el planeamiento de trayectorias, con el fin de proveer una solución basada en sistemas embebidos, que permita su ejecución con restricciones de tiempo real y conserve dimensiones reducidas.

3. Validación del algoritmo de planeamiento de trayectorias en línea propuesto, ejecutado sobre el sistema embebido.

1.4 Metodología propuesta

Con el fin de presentar una perspectiva general de la metodología que se usó para desarrollar el trabajo de investigación se diseñó el diagrama de bloques presentado en la Figura 1.

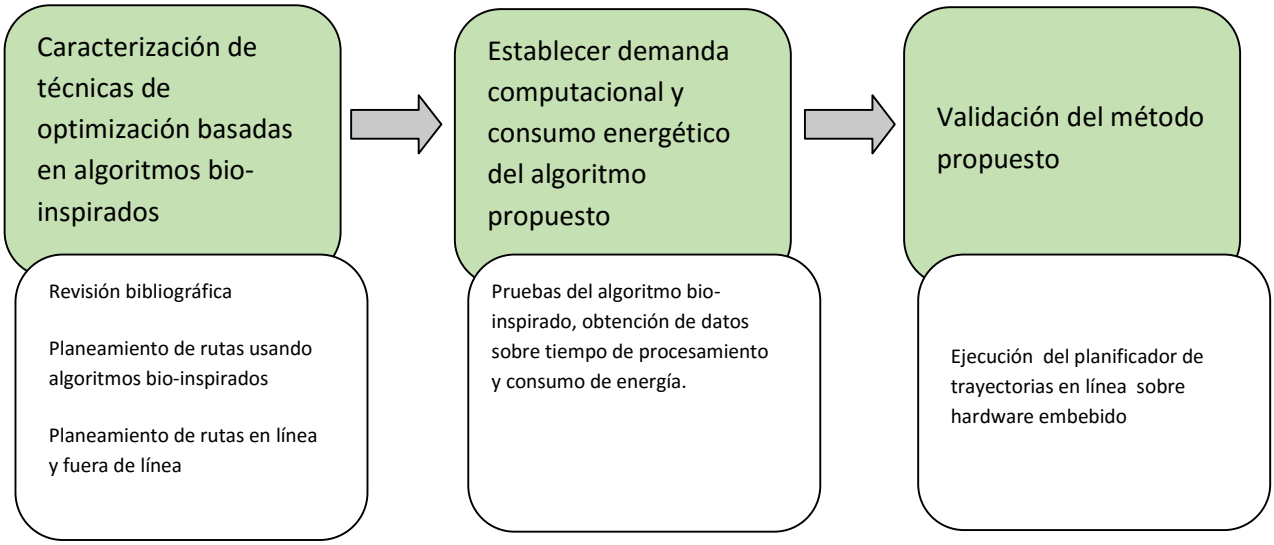

Figura 1: Metodología para el desarrollo del trabajo de investigación

Los bloques de izquierda aderecha representan la secuencia que se siguió para desarrollar los objetivos propuestos en la tesis, además establecen la ruta que permitió llevar a término el presente trabajo de investigación.

El proceso experimental que soporta la selección del algoritmo de optimización bio-inspirado, consistió en medir tiempo de ejecución, número de iteraciones del método y la menor distancia recorrida al ejecutar el planificador de ruta propuesto usando como base los algoritmos de optimización seleccionados. La selección del hardware embebido basado en microcontroladores, consistió en ejecutar el planificador de ruta con el algoritmo de optimización seleccionado previamente sobre los sistemas embebidos.

1.5 Estructura del documento

El presente proyecto se compone de cuatro capítulos: el segundo capítulo detalla el marco teórico, el estado del arte, la descripción de las técnicas de optimización bio-inspiradas yla revisión del hardware embebido susceptible a ser implementado sobre UAVs. El tercer capítulo presenta el método propuesto para el planeamiento de rutas. El cuarto capítulo describe el proceso experimental que se siguió y los resultados obtenidos. El trabajo finaliza con las conclusiones detalladas del trabajo.

2. Marco teórico y estado del arte

Con el fin de presentar un marco general del planeamiento de rutas para vehículos aéreos no tripulados, este capítulo se inicia con la descripción de las técnicas usadas para el planeamiento de rutas en UAVs. A continuación, se enfoca el contenido en las técnicas para el planeamiento de rutas basadas en algoritmos bio-inspirados encontradas en la revisión bibliográfica; a partir de allí, se ahonda en la temática propia de los algoritmos de optimización bio-inspirados, además de su implementación sobre hardware embebido.

2.1 Planeamiento de rutas, en línea y fuera de línea

Una ruta de vuelo es un conjunto de coordenadas geográficas que debe seguir una aeronave para ir de un punto de partida a un punto destino. Donde, el algoritmo de planeamiento de ruta es el encargado de encontrar dichos puntosconsiderando una serie de restricciones impuestas por el entorno o por la misión (Peng & Li, 2012). Los algoritmos para el planeamiento de ruta se dividen en dos grupos, los que realizan el planeamiento en línea y los que lo realizan fuera de línea (Cheng, Wang, Tang, & Wang, 2014)-(Xue et al., 2014).

El planeamiento de la ruta fuera de línea es una técnica en la cual ya existen rutas preestablecidas para vuelo directo y una ruta auxiliar en caso de emergencias. El planeamiento de ruta en línea viene de la necesidad de dotar a una aeronave con la capacidad de corregir su trayectoria en función de los obstáculos detectados, es decir, los UAVs deben tener la capacidad de determinar la trayectoria y corregir su rumbo a lo largo de una ruta de referencia (john tisdale, zuwhan kim, 2009). Es por eso que el problema general del planeamiento de ruta en línea son los ambientes dinámicos, donde el entorno es parcial o totalmente desconocido (Cheng et al., 2014).

En Chen, Chang, & Agate, (2013), los autores presentan un algoritmo para el planeamiento de ruta basado en el seguimiento de objetivos en tierra. La técnica propuesta en este trabajo se basa en un sistema de guía vectorial, compuesto por un

vector tangente al círculo equivalente al radio de la amenaza, y otro definido como guía de campo, los cuales, según los autores, permiten que el algoritmo explore la red de carreteras, la cual le sirve como guía mientras traza un camino más corto a través de los obstáculos detectados. Otra técnica para el planeamiento de ruta se encuentra en Wu, Zhang, & Pei, (2014). En este trabajo se presentauna técnica para el planeamiento de ruta en 3D bajo la consideración del conocimiento preliminar de las posibles amenazas, lo que permite establecer un modelo de probabilidad de todas las amenazas. Seguido al modelo de probabilidad, se ejecuta el algoritmo “Dijkstra”, el cual es un algoritmo ampliamente reconocido en el planeamiento de rutas(Wei, 2014).

En Xue et al., (2014) se presenta un algoritmo de planificación de trayectoria para UAVs denominado como “*D* lite IPRM*”. Este algoritmo consta de dos pasos: el primero es un método probabilístico llamado (*Roadmap*), usado para la planificación local y el segundo paso es el uso del algoritmo D*, el cual puede planificar la ruta desde el punto actual hasta el punto destino.

Como se puede ver en las técnicas descritas anteriormente para el planeamiento de rutas para UAVs, todos los algoritmos propuestos necesitaban cuantificar los niveles de amenazas en la ruta, esto significa un pre-procesamiento de la información, conocimiento previo de las amenazas en la ruta de vuelo y la necesidad de conocer el entorno topográfico sobre el cual se va a desarrollar el vuelo. Estas consideraciones presentan retos difíciles de subsanar a la hora de implementar los algoritmos propuestos en estos trabajos sobre sistemas embebidos con baja capacidad de cómputo y dimensiones reducidas.

2.2 Técnicas para el planeamiento de rutas que usan algoritmos de optimización bio-inspirados

Dentro de los métodos de optimización inspirados en la naturaleza se encuentran algunas técnicas sobresalientes, con características definidas por el comportamiento biológico que las inspira. Las características de un algoritmo bio-inspirado permiten abordar diferentes problemas qué, por su concepción, no responden a un modelo matemático conocido o de fácil solución. Cada algoritmo bio-inspirado presenta

diferentes soluciones para el mismo problema, siendo fuertes en unos casos y débiles en otros.

En la literatura se encuentran trabajos relacionados con el planeamiento de rutas para UAVs usando técnicas de optimización bio-inspiradas, a continuación se escriben las más relevantes halladas en la literatura.

2.2.1 Planeamiento de rutas usando algoritmos genéticos (AG)

La optimización usando algoritmos genéticos se inspira en el concepto de evolución biológica. Siendo la evolución biológica, la capacidad de mutación de una población en busca de la supervivencia de los más actos. Es decir, los individuos de una población que me mejor se adapten al entorno tendrán más posibilidad de sobrevivir y reproducirse(Digalakis, Margaritis., 2000). En este sentido, la solución de problemas de búsqueda y optimización que aplican el concepto de algoritmos genéticos, solo requieren conocer la función objetivo de la población, tamaño de la población y la probabilidad de cruce(Roberge, Tarbouchi, Labonté., 2013).Un algoritmo genético se compone con los siguientes elementos:

- Cromosoma: representación de una posible solución, ejemplo representación binaria
- Función de evaluación, que indica si un individuo es apto y la solución entregada es optima
- Operadores básicos: reproducción, cruce y mutación.
- Parámetros de un algoritmo genético: probabilidad de cruce, probabilidad de mutación, tamaño de la población, número de generaciones

A continuación se describe la secuencia lógica del algoritmo genéticosolución(Rodríguez-piñero, 2003):

- Evaluar cada uno de los cromosomas generados
- Permitir la reproducción de los cromosomas siendo los más aptos los que tengan más probabilidad de reproducirse

- Probabilidad de mutación de los genes (Indica la frecuencia con la que los genes de un cromosoma son mutados, si la probabilidad de mutación es del 100%, la totalidad del cromosoma se cambia. Es decir, cuantos bits de los que conforman el cromosoma se cambian)
- Organizar la nueva población

Estos pasos se repiten hasta que se dé una condición de parada, ya sea por número máximo de iteraciones o con una función que determine convergencia del algoritmo en el mínimo de la función objetivo.

En Meng y Xin (2010), se puede decir que el planeamiento de ruta que se vale de un algoritmo genético para suavizar la superficie de la ruta y luego realizan el planeamiento con el algoritmo recocido simulado (simulated annealing). Se observa que para hacer el planeamiento de ruta seejecuta un pre procesamiento del mapa de elevacionesen buscade suavizar la superficie. Luego de suavizar la superficie lanzan la técnica para hacer el planeamiento de ruta. Un análisis más a fondo de este trabajo indica que difícilmente se podría implementar sobre un UAV; este concepto se desprende del hecho que en un vuelo real, a medida que se recibe información del entorno sería computacionalmente costoso pre procesar toda la información recibida antes de poder lanzar el planificador de ruta. En segundo lugar, en la literatura es conocido de los problemas que presenta el algoritmo (simulated annealing), al converger en mínimos locales y no necesariamente en el mínimo global, posiblemente esta es la razón por la que necesitan pre procesar el mapa de alturas, por tanto, desde nuestro punto de vista el trabajo aún se encuentra incompleto y debe subsanar falencias graves. En la literatura se encontraron otros trabajos que realizan el planeamiento de ruta usando algoritmos genéticos (Roberge et al., 2013).

2.2.2 Planeamiento de rutas usando la optimización colonia de hormigas (ACO)

Un comportamiento importante e interesante de una colonia de hormigas es su proceso de búsqueda de alimento de cooperación indirecta. Al caminar por las fuentes de alimento al nido y viceversa, las hormigas depositan una sustancia llamada rastro de feromona. Las hormigas pueden oler feromonas y, al elegir su camino, tienden a elegir,

con alta probabilidad, senderos marcados por una fuerte concentración de feromona (ruta más corta) (Pei, Wang, & Zhang, 2012).

El algoritmo ACO se inspira directamente en el comportamiento de las colonias de hormigas para solucionar problemas de optimización. Su comportamiento se basa en agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales. En cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo de construcción. Cada arista del grafo representa posibles pasos que la hormiga puede dar y en consecuencia, después de una fase inicial donde se producen diversos caminos, las hormigas convergen en un solo camino (Duan, Yu, Zou, & Feng, 2010).

En Zaza & Richards, (2014) se propone el planeamiento de rutas para múltiples UAVs con prevención de colisiones.Otro trabajo que usa (ACO) se encuentra en (Shudao, Jun, Yongqi., (2012) donde los autores proponen un algoritmo basado en el comportamiento de las hormigas y los denominados diagramas de Voronoi, los cuales proporcionan una ruta entre la coordenada de salida y la coordenada destino. La técnica propuesta consiste en distribuir las amenazas encontradas en la ruta del avión con los diagramas de Voronoi, clasificándolas y asignándoles un costo, para luego utilizar la optimización por colonia de hormigas para encontrar la mejor ruta a través de las amenazas.

2.2.3 Planeamiento de rutas usando la optimización enjambre de partículas (PSO)

Otro algoritmo de optimización basado en la naturaleza, y que resulta ser muy usado en la investigación en el planeamiento de rutas, es la optimización por enjambre de partículas (PSO). Su uso tan extenso se debe a que PSO es un algoritmo evolutivo estocástico que no incorpora la supervivencia de los más aptos, todos los individuos se mantienen como miembros de la población a través del vuelo. Cada partícula ajusta su vuelo en el espacio de búsqueda en términos de su propia experiencia de vuelo y vuelo de sus compañeros (Civicioglu & Besdok, 2013).

La optimización por enjambre de partículas se refiere a una serie de métodos y algoritmos de optimización heurísticos que evocan el comportamiento de los enjambres de abejas en la naturaleza, los movimientos de grandes grupos de insectos o de bandadas de peces. En estos grupos se estudian los movimientos de cada individuo en busca de un objetivo, con la ventaja de que existe siempre alguna forma de comunicación entre los individuos. De esta forma, cada uno de ellos se mueve impulsado por tres variables: en primer lugar, la experiencia de los mejores lugares en los que ha estado hasta el momento; en segundo lugar, el mejor lugar descubierto por algún otro miembro del enjambre; por último, una componente de inercia del movimiento anterior que amortigua las velocidades de las partículas (Lee & Kim, 2013).

La ecuación (1) expresa la velocidad de la partícula mientras que la posición de la i-enésima partícula se expresa en la ecuación (2) (Clerc, 2005; Leonard & Engelbrecht, 2013).

$$v_j^{(n)} = \varphi_1 v_j^{(n-1)} + \alpha_2 \varphi_2 \left(\psi_g - \boldsymbol{x}_j^{(n-1)}\right) + \alpha_2 \varphi_2 (\psi_j - \boldsymbol{x}_j^{(n-1)}) \qquad (1)$$

Posición de la i-ésima partícula:

$$\boldsymbol{x}_j^{(n)} = \boldsymbol{x}_j^{(n-1)} + v_j^{(n)} \qquad (2)$$

Los parámetros de PSO son tomados de (Clerc, 2012):

- $v_j^{(n)}$, Velocidad ajustada
- $v_j^{(n-1)}$, Inercia del propio movimiento
- φ_1, Coeficiente de velocidad de la partícula [0-1], para este trabajo se tomó como $\frac{1}{2ln(2)}$
- φ_2, Coeficiente de confianza en la experiencia propia y del enjambre, $\frac{1}{2} + ln(2)$

- ψ_j, Mejor posición previa de la partícula
- ψ_g, Mejor posición previa encontrada por el grupo
- $\boldsymbol{x}_j^{(n)}$, Posición actual de la partícula
- $\alpha_2 \sim \mathcal{N}(0\,,1)$, Es una variable aleatoria extraída de una distribución gaussiana con media cero y varianza uno

En este algoritmo, cada solución se asocia con las coordenadas de un punto y, mediante la función objetivo, se calculará la bondad de cada solución o punto. El algoritmo comenzará con una nube de puntos que irán moviéndose en busca de mejores soluciones hasta llegar a la solución óptima (Fu & Member, 2012; P.K. Rout, Acharya, & Panda, 2010).

La **Figura** 2, ilustra el comportamiento social y personal de las partículas.

Figura 2: Comportamiento social del algoritmo PSO.Fuente, el autor.

En la **Figura 2**, se observa que las diferentes soluciones llamadas partículas recorren el espacio de búsqueda guiadas por la partícula que ha encontrado la mejor solución hasta el momento, lo que la convierte en líder del enjambre. Cada partícula evoluciona teniendo en cuenta su propia experiencia y la mejor solución encontrada por el enjambre. El resultado de este comportamiento es que las partículasconvergen en la mejor posición encontrada por el enjambre hasta que se cumpla la condición de parada, ya sea por número de iteraciones o por encontrar el error de referencia.

En Peng & Li, (2012)se propone una técnica denominada como modelo predictivo de control para el planeamiento de rutas. En síntesis,es una técnica que necesita conocer parcialmente el terreno para realizar un pre planeamiento de ruta. El planeamiento previo

le sirve como ruta de referencia, incorporando las posibles amenazas relacionadas con objetos desconocidos o factores climáticos durante la fase de vuelo en tiempo real, **Figura 3**. El resultado de este trabajo no es concluyente; primero, porque necesita una ruta de referencia, por lo que no es recomendable para ser usado en entornos desconocidosy segundo las simulaciones que soportan el trabajo se realizaron sobre una función matemática para levantar el mapa de alturas, en ningún momento se confronto contra una base de datos que represente una topología real.

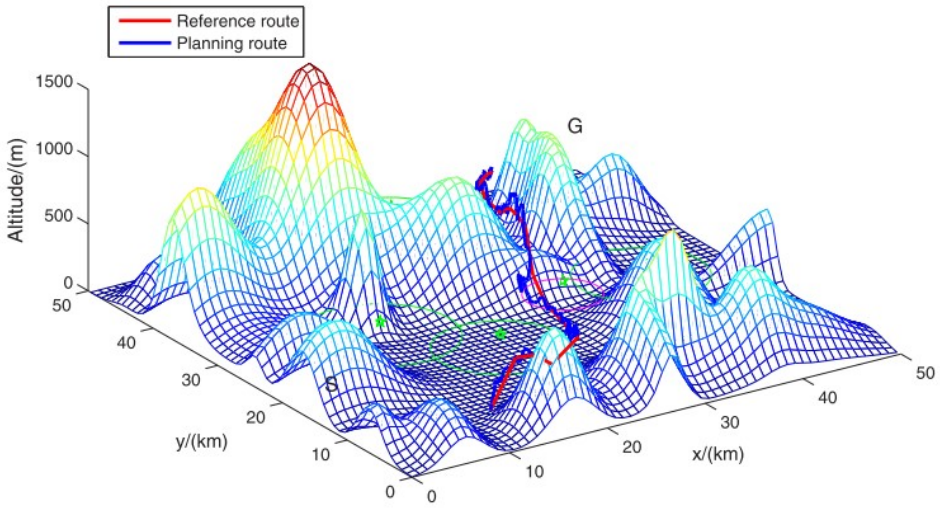

Figura 3: Planeamiento de ruta usando un modelo predictivo de control y PSO (Peng & Li, 2012)

Como se mencionó anteriormente, la optimización por enjambre de partículas ha sido ampliamente usada en la formulación de técnicas para el planeamiento de rutas. En Lamont, Slear, & Melendez, (2007) y Roberge et al., (2013) se presentan trabajos para el planeamiento de rutas que usan como base de optimización PSO. Y aunque son aporte al estado del arte, se observa que la evaluación se realiza sobre mapas ficticios levantados con funciones matemáticas que no responden a topologías reales. Además, es evidente que los trabajos son a nivel de simulación sobre plataformas robustas, ejemplo Matlab sobre una maquina Pentium 2.6GHz. En ningún trabajo se observa que se halla llevado estas técnicas a nivel de hardware embebido, el cual sería un paso necesario en miras de implementar el planeamiento de ruta en tiempo real sobre un UAV.Otros trabajos querealizan el planeamiento de ruta usando PSO (Yangguang Fu, Mingyue Ding, Chengping Zhou, 2013) (Roberge et al., 2013)(Lamont et al., 2007).

2.2.4 Optimización usando el algoritmo por búsqueda Cuckoo

Recientemente se han propuesto nuevos algoritmos de optimización los cuales no se han probado en el planeamiento de rutas. Dentro de estos se destaca la búsqueda Cuckoo el cual es un algoritmo meta heurístico que actúa esparciendo de manera aleatoria una población a lo largo de un espacio de búsqueda calificando las soluciones y seleccionando las mejores, descartando una parte de las peores soluciones sustituyéndolas por nuevas soluciones. En Civicioglu & Besdok, (2013). Los autores realizan la evaluación de este algoritmo de optimización comparándolo con la optimización por enjambre de partículas. Los resultados, después de ejecutar diferente funciones, arrojaron que la búsqueda Cuckoo presenta en muchos casos mejores resultados que PSO en un menor tiempo de ejecución. En la literatura se encuentran otros trabajos relacionados con la búsqueda Cuckoo, donde se presentan variantes del algoritmo con el fin de mejorar su convergencia y precisión de la solución(Li & Yin, 2015; Nguyen & Vo, 2015; Yang & Deb, 2014; Zhou & Zheng, 2013).

La optimización por búsqueda Cuckoo (CK) es un método basado en el comportamiento parásito de los Cuckoos a la hora de depositar sus huevos en nidos ajenos; se asocia cada posible solución del problema a las coordenadas de un nido que se ocupa (Civicioglu & Besdok, 2013). De esta forma, se parte de un conjunto inicial de nidos en los que se ha depositado un huevo, de todos ellos, los mejores huevos pasarán desapercibidos y sobrevivirán, pero los peores huevos tendrán una probabilidadde ser descubiertos por el dueño del nido y serán desechados. Para la selección de nuevos nidos, y completar el espacio de búsqueda, se imita el comportamiento de algunas moscas de la fruta y aves que muestran un vuelo aleatorio que se ajusta a los denominados vuelos de Lévy (Lévy Flights). Al generar nuevas soluciones$x_g + 1$para un cuckoo i-enésimo se genera un vuelo Lévy dado por la ecuación (**3**):

$$x_g + 1\,; i = x_g\,; i + \propto * Levy\,(\mathrm{S}) \qquad (3)$$

Este tipo de vuelo se caracteriza por largos recorridos en línea recta seguidos de bruscos cambios de dirección. La dirección del giro se distribuye uniformemente y el tamaño del

incremento S (paso del vuelo de Levy) y, se establece de acuerdo a la siguiente distribución de Lévy (Yang & Press, 2010):

$$S = \frac{\partial}{|\vartheta|^{\frac{1}{\beta}}} \qquad (4)$$

Donde ∂ y ϑes una variable aleatoria extraída de una distribución normal con media cero, ecuación (**5**) y un $\beta = \frac{3}{2}$

$$\partial \sim \mathcal{N}(0, \sigma_\partial^2) \quad , \quad \vartheta \sim \mathcal{N}(0, \sigma_\vartheta^2) \qquad (5)$$

Donde σ_∂ y σ_ϑ se obtienen en la ecuación (6).

$$\sigma_\partial = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}}\right\} \qquad \sigma_\vartheta = 1 \qquad (6)$$

Función Gamma $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$

2.3 Algoritmos de optimización bio-inspirados sobre hardware embebido

La capacidad de cómputo de los sistemas que pueden trabajar bajo el concepto de computación paralela ha sido aprovechada para procesar grandes volúmenes de datos, así como para experimentar con nuevas técnicas y algoritmosque permiten optimizar ciertas tareas con el fin de realizar un trabajo de forma más eficiente. Un ejemplo de esto se encuentra en Ginan, (2014), donde los autores proponen una técnica para el planeamiento de ruta para UAVs, usando un algoritmo de optimización inspirado en el comportamiento social de las hormigas al buscar comida.También es común encontrar trabajos que se enfocan en analizar el rendimiento en tiempo de ejecución y consumo energético al ejecutar algoritmossobre CPU, GPU y FPGAs, para lo cual, también se encuentran trabajos que estudian la arquitectura de hardware para trabajar con dichos

algoritmos, en especial los que requieren de operar con números con punto flotante y generadores de números aleatorios (Arboleda, Llanos, Coelho, & Ayala-Rincón, 2009). Esto permite a los investigadores dimensionar cuál sería el mejor sistema para ejecutar un proceso experimentalque requiera de grandes prestaciones computacionales (Jones, Powell, Bouganis, & Cheung, 2010; Kalarot & Morris, 2010; Kestur, Davis, & Williams, 2010; Matsuo, Hamada, Miyoshi, Shibata, & Oguri, 2009; Papadonikolakis, Bouganis, & Constantinides, 2009; Spurzem et al., 2009).

A continuación se describen los sistemas más usados a nivel de computación científica:

- **Un procesador multi núcleo** es un sistema que combina dos o más microprocesadores independientes en un solo circuito integrado. Por su estructura posibilitan de cierta forma procesamiento en paralelo.

- **Unidad de procesamiento grafico (GPU)**, es un circuito diseñado para acelerar la salida de la imagen en una memoria intermedia destinada a la salida en una pantalla. Aunque su función original es de alivianar al procesador central del manejo de imágenes, por su velocidad de procesamiento, se le está dando uso en el cálculo numérico(Matsuo et al., 2009).

- **Field Progammable Gate Array (FPGA)**, es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada mediante un lenguaje de descripción, ya sea VHDL o por circuitos esquemáticos. Su lógica puede producirfunciones básicas que, combinadas, generan aplicaciones robustas, fácilmente reconfigurables. Además, su versatilidad se encuentra en el poder tener procesamiento en paralelo a costos computacionales reducidos (P.K. Rout et al., 2010; Prakash Kumar Rout, Acharya, & Panda, 2010).

Un sistema embebido basado en un sistema micro procesado,es un sistema electrónico que cuenta con una unidad de procesamiento,teniendo capacidad de llevar a cabo una o varias tareas específicas. Típicamente los sistemas embebidos son reactivos, es decir,

responden a eventos; su función, a diferencia de una computadora, es la de dar versatilidad al sistema ya que se puede tener acceso a periféricos fácilmente, sean dispositivos entrada-salida, o se puedeproporcionar comunicación con diferentes dispositivos. Los sistemas embebidos se caracterizan por ser arreglos de tamaño reducido en comparación con una computadora personal, pero con recursos de hardware adecuados para la aplicación que deba realizar.

Un sistema embebido de tiempo real es aquel donde existe un máximo tiempo de respuesta que debe ser respetado ante la presencia de un evento o perturbación externa que amerite ejecutar de nuevo el proceso. Por esto, se hace necesario instalar sistemas que realicen el procesamiento en el sitio, no necesariamente dependientes de un sistema de comunicación. A este nivel se destacan los sistemas basados en microcontrolador, los cuales tienen un tamaño reducido y bajo consumo energético (Baynes et al., 2005), pero con la desventaja de tener capacidad y velocidad de procesamiento reducida.

En la actualidad se adelantan investigaciones que buscan evaluar la capacidad de los sistemas embebidos basados en microcontroladores ejecutando algoritmos inspirados en sistemas biológicos.Como ejemplo se puede mencionar la solución que se observa en Munoz, Llanos, Coelho, & Ayala-Rincon, (2010), donde los autores proponen una técnica para implementar el algoritmo PSO sobre un microcontrolador. Los resultados que ellos obtienen muestran que es posible implementar dicho algoritmo sobre sistemas microcontrolados. Otro trabajo que demostró que es posible implementar un algoritmo bio-inspirado en un microcontrolador se observa en Abdullah-ai-nahid, Alam, & Plabon,(2012). Aquí, los autores presentan los resultados obtenidos al implementar el algoritmo PSO para realizar el seguimiento del sol, obtener su máxima potencia y generar la mayor energía derivada de la luz solar. Otro trabajo que evalúa el rendimiento de la optimización usando los enjambres de partículas se encuentra en Tewolde, Hanna, & Haskell, (2009). Aquí los autores implementan el algoritmo PSO sobre un microcontrolador de 16 bits, además los resultados de la evaluación del rendimiento fueron comparados con la ejecución del mismo algoritmo sobre una FPGA (Field Programmable Gate Array), y como es de esperarse la aceleración obtenida en la FPGA fue muy superior a la obtenida con el microcontrolador de 16 bits. Pero también como es de esperarse, el consumo de energía y las dimensiones de la FPGA son mayores que las obtenidas en un microcontrolador.

El gran problema que se afronta, es la implementación de algoritmos de optimización bio-inspirados sobre hardware embebido con dimensiones reducidas y baja capacidad computacional. Con relación a esto, se observan trabajos que proponen variantes e híbridos de algoritmos de optimización con el fin de obtener una mejor convergencia, sin el problema que la solución se quede en mínimos locales, llegando a una solución global. Este es el caso que se presenta en Idoumghar, Melkemi, Schott, & Aouad, (2011). Aquí, los autores proponen un algoritmo denominado como “HPSO-SA”, el cual se caracteriza por combinar la capacidad exploratoria del PSO con la capacidad de encontrar mínimos locales del algoritmo recosido simulado. Como resultado de esta combinación de algoritmos de optimización se obtuvo una reducción en el consumo de energía de 76% y un ahorro de tiempo de 73%.

2.4 Síntesis del estado del arte

El estado del arte sobre el planeamiento de rutas muestra que hay dos ramas bien definas: la primera en el planeamiento de ruta fuera de línea, técnica que se basa en el conocimiento del terreno y rutas previamente establecidas, lo que es común en la aviación convencional, cualquier cambio depende de un controlador en tierra. La segunda rama se define como el planeamiento de ruta en línea, en este caso el planificador de ruta se ejecuta en tiempo real, permitiendo tener en cuenta amenazas desconocidas que se presentan en la ruta de vuelo.

Este trabajo se centra en el planeamiento de ruta en línea, en ese sentido los trabajos que investigan y desarrollan metodologías para planeamiento de rutas en línea usando algoritmos bio-inspirados, combinan técnicas que les permiten mejorar la precisión de la solución. Es decir, tener rutas que recorren una menor distancia entre el punto de salida y el punto de llegada. El aporte de estos trabajos es importante en ambientes donde se conoce la ruta de vuelo, y por ende la topografía sobre la cual se va a volar.Aun que se menciona que el planeamiento es en línea, se observan falencias graves como el no determinar el costo computacional de los algoritmos propuestos. En la totalidad de los casos observados las simulaciones se realizan sobre equipos de cómputo robustos que por su tamaño no es factible su implementación sobre un UAV. Además el criterio de

evaluación solamente se centra en la distancia mínima recorrida, en ningún momento tiempo de ejecución. Algunos de los equipos empleados en las simulaciones observadas en el estado del arte son: Matlab sobre una maquina Pentium 2.6 GHz, Matlab sobre una maquina Core2 Duo E7200, Matlab (módulo paralelo) Dell PE 2900 quad-core 1.6GHz.

Sobre las simulaciones se puede mencionar que en la mayoría de casos se usaron funciones matemáticas para generar un mapa de alturas. En otros casos realizan pre-procesamiento del terreno con el fin de suavizar la superficie yen pocos casos hacen uso de topografías reales para realizar las simulaciones. El no uso de bases de datos reales con topografías del terreno es una gran falencia, en especial porque los desniveles en un mapa topográfico real no responden a una función o modelo conocido. Por esta razón las funciones matemáticas seleccionadas para generartopografías no presentaban mayores retos al planificador de ruta, por lo que el algoritmo de optimización difícilmente se podría quedar en una solución local.

Sobre la ejecución de planificadores de ruta sobre microcontroladores no se hallaron evidencias o mención alguna sobre su implementación. Pero en la literatura se hallaron algunos trabajos que han evaluado el rendimiento de microcontrolares al ejecutar algoritmos de optimización bio-inspirados, ejemplo PSO ejecutado sobre una maquina freescale de 16 bits, cuyos resultados fueron satisfactorios.

3. Método propuesto para el planeamiento de ruta

En estasección se presenta el método propuesto para el planeamiento de rutas en tiempo real, para la navegación en ambientes desconocidos usando algoritmos bio-inspirados. El método propuesto basa su funcionamiento en la información recibida de los sistemas de sensores, la cual sirve de entrada al algoritmo de optimización que evalúa una función objetivo, retornando el costo mínimo de las trayectorias evaluadas. Para evaluar el método y minimizar la función objetivo se seleccionaron dos algoritmos de optimización basados en poblaciones. El primero es el enjambre de partículas (PSO). Su selección se dio a partir de los buenos resultados que se han obtenido con este algoritmo en la solución de problemas de optimización multi objetivo. Además, se halló un trabajo en el cual se implementó el algoritmo PSO sobre un microcontrolador de 16 bits (Tewolde, Hanna, &Haskell, 2009).

En cuanto al algoritmo Cuckoo, la literatura indica que es un algoritmo relativamente nuevo, el cual está siendo evaluado con diferentes funciones (Li & Yin, 2015; Nguyen&Vo, 2015; Yang &Deb, 2014; Zhou&Zheng, 2013). Los resultados observados en estos trabajos muestran que en muchos de los casos alcanzaba asuperar al algoritmo PSO en un menor tiempo de ejecución y solución óptima. Por tal motivo y en aras de verificar si el algoritmo CK tiene este mismo comportamiento al ejecutar el planificador de ruta se decidió implementarlo en la simulación y comparar sus resultados contra PSO.

3.1 Planeamiento de ruta en tiempo real

El método para la planificación de la ruta consiste en evaluar cada posible trayectoria generada por los individuos. La evaluación de las trayectorias permite obtener la coordenada del mejor punto encontrado por la población, donde la nueva coordenada obtenida sirve como vector de dirección y se utiliza como base para determinar el nuevo punto de avance en función de la velocidad crucero. El conjunto de todas las coordenadas encontradas se almacena en una matriz $\boldsymbol{P}$, donde la i-ésima fila

corresponde a la coordenada $\boldsymbol{\rho}^{(i)}$ encontrada en la i-ésima ejecución del algoritmo y se calcula como la suma entre la coordenada actual y la coordenada entregada por el algoritmo de optimización$\boldsymbol{x}^*$, multiplicada por el espacio recorrido a velocidad crucerov_{cr}en un tiempo $\boldsymbol{t}^{(i)}$, ecuación (**7**).

$$\boldsymbol{\rho}^{(i)} = \boldsymbol{\rho}^{(i-1)} + v_{cr} * \boldsymbol{t}^{(i)} * \frac{(\boldsymbol{x}^* - \boldsymbol{\rho}^{(i-1)})}{\left|\left|\boldsymbol{x}^* - \boldsymbol{\rho}^{(i-1)}\right|\right|} \qquad (7)$$

Donde:

- $\boldsymbol{\rho}^{(i)}$, es la coordenada calculada en la iteracióni
- $\boldsymbol{t}^{(i)}$, es el tiempo en segundos que tarda el algoritmo de optimización en entregar una nueva coordenada de la iteración i
- $\boldsymbol{v}_{cr}$, velocidad del avión en crucero (cruising speed)
- δ, es la visión en el horizonte el cual determina el espacio de búsqueda

En la **Figura** 4, se describe el método para la planeación de ruta en tiempo real.

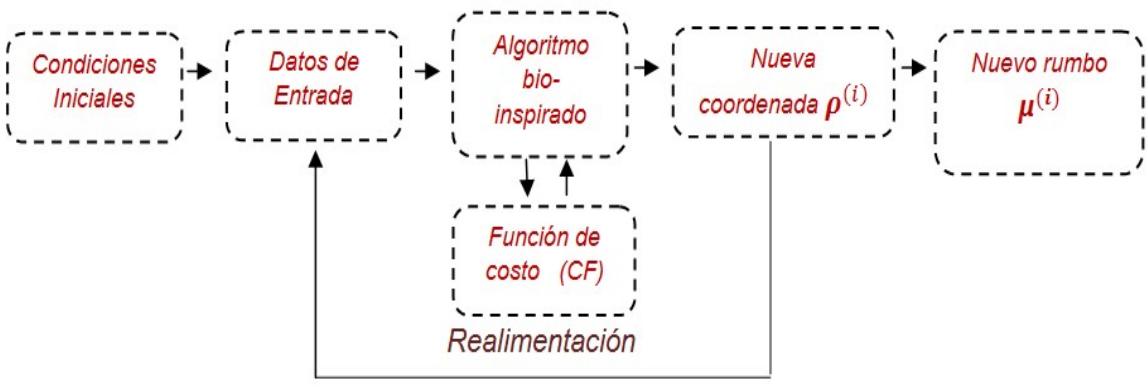

Figura 4: Metodo propuesto para el planeamiento de rutas on-line

Las condiciones iniciales proporcionan límitesal planificador de ruta que no permiten violar las propiedades dinámicas delavión mientras se esté realizando un vuelo. La planificación de ruta en línea que no cumplacon las restricciones mínimas soportadas por

la aeronave (constraint handling (C_h)), dará como resultado la pena de muerte para la partícula o el cuckoo, lo que implica que solo las más aptas serán evaluadas.

Las entradas (inputs) corresponden a la información que se suministra al algoritmo de optimización. A continuación se relacionan los parámetros que componen las entradas del sistema:

- Base de datos que corresponde a una matriz de alturas para cada punto coordenado,$\boldsymbol{H}$.
- Coordenada objetivo en latitud y longitud, $\boldsymbol{\rho}_{tar}$
- Coordenada de salida inicial en latitud y longitud $\boldsymbol{\rho}_{ini}$,, actualizable en cadanueva coordenada entregada por el algoritmo de optimización de ruta
- Velocidad de viraje = ratio de giro θ_g y se da en grados/segundo

El espacio de búsqueda está dado por la distancia de visión (δ), donde lasposiciones iniciales de las partículas o los Cuckoos están dispersas en el espacio de búsqueda con distribución normal en función del espacio de visión. El espacio de búsqueda queda expresado en la ecuación (**8**):

$$\boldsymbol{x}_j = \alpha 1\delta + \boldsymbol{\rho}_{ini} \qquad (8)$$

Donde:

- $\boldsymbol{x}_j$,es la j-ésima partícula de la población inicial
- $\alpha 1 \sim \mathcal{N}(0\ ,1)$, es una variable aleatoria extraída de una distribución gaussiana con media cero y varianza uno
- $\boldsymbol{\rho}_{ini}$, es la posición inicial que sirve de referencia para centrar el espacio de búsqueda

3.2 Función de costo

Los sistemas de planificación de ruta asocian la mejor ruta con la búsqueda del camino más corto. Este es el caso del problema del vendedor viajero (TSP), que consiste en encontrar el camino más corto para visitar las ciudades una sola vez. En este trabajo, la función de costo evalúa tanto la distancia más corta hacia una coordenada destino, como las características de aeronavegación propias de un vehículo aéreo. Es decir, ratio máximo de giro, velocidad real (TAS), y altura de vuelo. Además, se tienen en cuenta restricciones que vienen dadas por la imposibilidad de cruzar por ciertos lugares, ya sea por cambios en el clima o cualquier otro obstáculo que amerite determinar un nuevo cambio en la trayectoria y, en consecuencia, una nueva ruta.

Para este caso, la función de costo se compone de sub costos que tienen diferente grado de importancia en la evaluación de cada una de las partículas. Este grado de importancia se denomina como w_x y sus valores están entre [0,1].

La función de costoqueda expresada de la siguiente forma:

- Costo de la distancia C_{dist} :

El costo de la distancia C_{dist} ecuación **(9)**, indica qué tan cerca se encuentra la partícula de la coordenada destino, permitiendo evaluar y seleccionar las partículas que se acerquen más al destino.Su grado de importancia w_1 dentro de la función de costo es igual a1.

$$C_{dist} = \frac{(\bar{u}_1 + \bar{u}_2) - \bar{u}_3}{\bar{u}_3} \qquad (9)$$

$$\bar{u}_1 = ||\boldsymbol{\rho}_{ini} - x_j\ || \qquad (10)$$

$$\bar{u}_2 = ||x_j - \boldsymbol{\rho}_{tar}\ || \qquad (11)$$

$$\bar{u}_3 = ||\boldsymbol{\rho}_{ini} - \boldsymbol{\rho}_{tar}\ || \qquad (12)$$

- Costo del ratio de giroC_θ:

Este componente permite seleccionar la partícula que proporcione cambios de dirección suaves y que no exceda la tasa de giro del avión por segundo. El costo de ratio de giro queda expresado de la siguiente manera:

$$C_\theta = \theta_g \qquad (13)$$

El ángulo θ_g, ecuación (**14**) corresponde al ratio de giro de la aeronave, donde el radio de viraje que se utiliza para los puntos de referencia en vuelo se basa en un ángulo de inclinación lateral $Roll(\emptyset)$ a la velocidad real TAS dada $\frac{km}{h}$ (Civil, 2009).

$$\theta_g \frac{6355 \tan(\emptyset)}{\pi v_{cr}} \qquad (14)$$

Donde:

- v_{cr}= $(TAS + velocidad\ del\ viento)$ en $\frac{km}{h}$
- $\emptyset$ = ángulo de inclinación $(Roll)$
- TAS= velocidad real, en $\frac{km}{h}$

El peso w_2 que tiene C_θ en la función de costo depende de θ_μ, siendo $w_2 = \begin{cases} 0.1 * \theta_g & si \quad \theta_\mu \geq \theta_g \\ 0.00001 * \theta_g & si \quad \theta_\mu < \theta_g \end{cases}$, donde θ_μ, ecuación (15), es el ángulo formado por la magnitud del vector que está definido entre la coordenada $\boldsymbol{\rho}^{(i-1)}$ que es la coordenada de inicio; la coordenada indicada por el vector dirección $\boldsymbol{\mu}^{(\boldsymbol{i-1})}$ que sigue el avión en el instante i y la coordenada $\boldsymbol{x_j}$ propuesta por la partícula en la i-ésima ejecución del algoritmo de optimización, **Figura 5**.

$$\theta_\mu = arcos\left(\frac{(x_j - \boldsymbol{\rho}^{(i-1)})\ (\boldsymbol{x}^* - \boldsymbol{\rho}^{(i-1)})}{\left|\left|x_j - \boldsymbol{\rho}^{(i-1)}\right|\right| \left|\left|\boldsymbol{x}^* - \boldsymbol{\rho}^{(i-1)}\right|\right|}\right) \qquad (15)$$

Siendo $$\boldsymbol{\mu}^{(i-1)} = \frac{(x^* - \boldsymbol{\rho}^{(i-1)})}{\left\|x^* - \boldsymbol{\rho}^{(i-1)}\right\|} \quad (16)$$

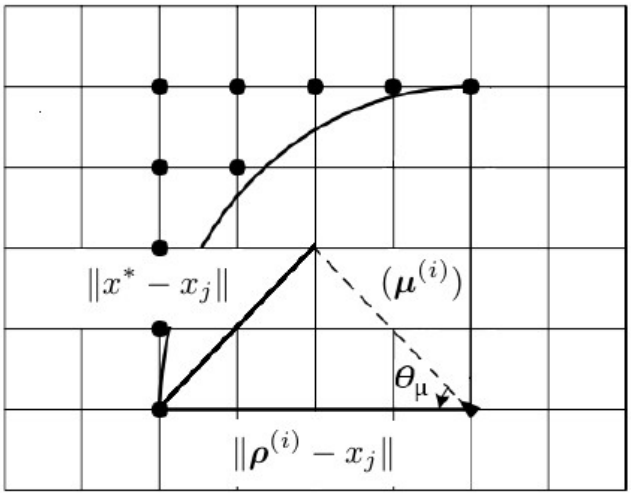

Figura 5: Dirección en el instante $\boldsymbol{i}$

- Costo de obstáculos imprevistos en la ruta de vuelo C_ϖ

El costo C_ϖ es la evaluación de los obstáculos imprevistos en la ruta de vuelo, asociados a fenómenos atmosféricos u otras aeronaves que se desplazan en la misma ruta o tienen peligro de colisión. Este componente evalúa las partículas que superan la altura mínima de vuelo sobre las montañas y en su horizonte de visión presentan algún tipo de obstáculo, teniendo un grado de importancia w_3 igual a 1. El costo por obstáculos imprevistos queda expresado de la siguiente manera, ecuación (**17**):

$$C_\varpi = \begin{cases} 0 & si R_g > R_{min} \\ \infty & si R_g \leq R_{min} \end{cases} \quad (17)$$

Donde:

- R_g, es la distancia entre el UAV y la posible amenaza.
- R_{min}, es la distancia de seguridad que se debe respetar entre el UAV y la posible amenaza para que el vuelo sea seguro.

El costo total del rumbo C_{rumbo}, ecuación (**18**) queda expresado de la siguiente manera:

$$C_{rumbo} = \begin{cases} w_1 C_{dits} + w_2 C_\theta + w_3 C_\varpi & si \quad h_{min} \leq h < h_{max} \\ \infty & si \quad h_{min} > h \quad o \quad h > h_{max} \end{cases} \tag{18}$$

Donde h es el elemento de la matriz de alturas H correspondiente a la posición de la partícula analizada.

4. Experimentos y resultados

Con el fin de validar el método propuesto en este trabajo, se seleccionaron los algoritmos enjambre de partículas (PSO) y la búsqueda Cuckoo. En la **Figura 6** se muestra el proceso se siguió con el fin de validar el método para el planeamiento de rutas ejecutado sobre hardware embebido.

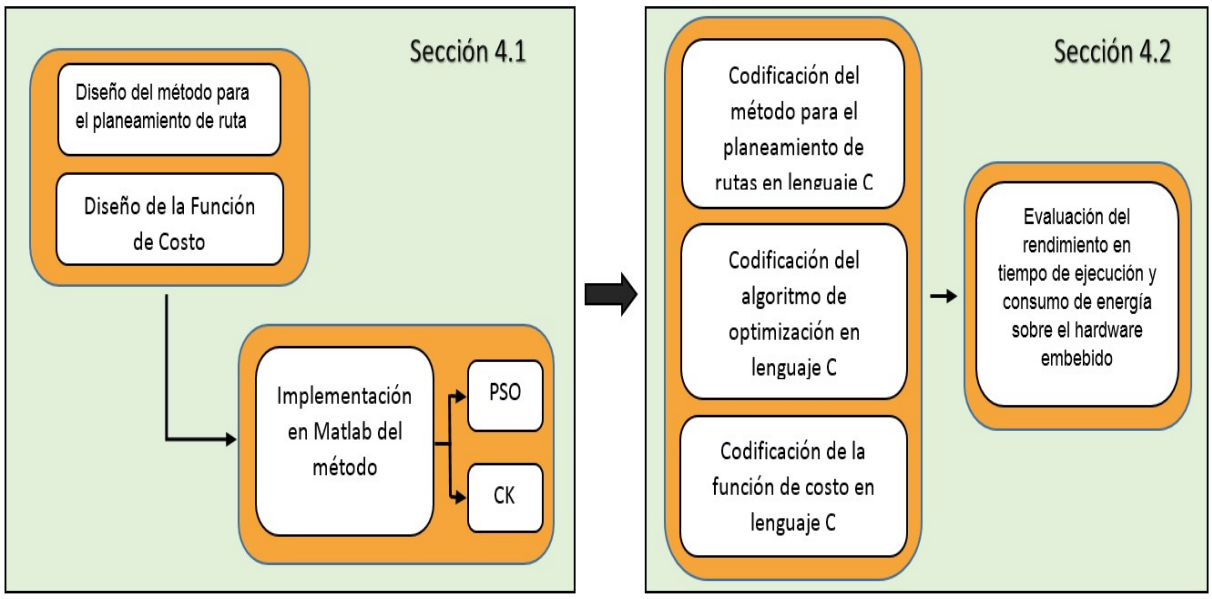

Figura 6: Validación del planificador de ruta sobre hardware embebido

Se inicia con la evaluacióndel método para el planeamiento de rutas usando como base los algoritmos de optimización PSO y el CK. Como resultado de la evaluación, se defineel algoritmo de optimización que presenta un mejor rendimiento al ser ejecutado con el método propuesto para el planeamiento de ruta.

En segundo lugar se implementa el planificador de rutas sobre los sistemas embebidos basados en sistemas microcontrolados con el fin de evaluar el rendimiento de este tipo de hardware al realizar el planeamiento de ruta en tiempo real. Los parámetros que se tuvieron en cuenta para evaluar el rendimiento fueron tiempo de ejecución y consumo de energía al realizar el planeamiento de la ruta.

4.1 Evaluación del planificador de ruta usando los algoritmos de optimización

Para los experimentos se usó una base de datos comprendida entre las latitudes 4.4029 grados N a 5.8860 grados N y las longitudes -76.3969 grados a -74.9069 grados. La conformación geográfica del área está constituida por valles y montañas de diferentes alturas y esta ubica en los departamentos de Caldas, Risaralda, Tolima, Antioquia y Choco. Su principal elevación se encuentra en el volcán Nevado del Ruiz a 5300 metros sobre el nivel del mar y su altura más baja se encuentra a 650 metros sobre el nivel del mar. La base de datos se obtuvo con la ayuda de **Google Earth** y el software de información geográfica **Global Maper**.

Para todos los experimentos se asume un horizonte de visión (δ) de 10 km. Los parámetros de simulación del método y de los algoritmos de optimizaciónse muestran en la **Tabla 1**.

Tabla 1: Parámetros de simulación

Descripción	Parámetro PSO	Parámetro CK
Miembros Población	20	20
Iteraciones	400	1000
Velocidad crucero	500km/h	500km/h
Roll	30	30
Ratio de giro	2,33 grados/segundo	2,33 grados/segundo
Altura de vuelo	Variable para cada experimento	Variable para cada experimento

Para el vuelo simulado se asume un entorno desconocido ya que la única información disponible para el espacio de búsqueda es el horizonte de visión, lo que constituye su dominio de predicción como δ_i para cada iteración. Por tanto, cada nuevo segmento descubierto implica un nuevo punto de avance $(i+1)$ que depende de la dirección seleccionada como mejor opción por la velocidad crucero determinada previamente para el avión. El algoritmo de planeamiento de ruta se ejecuta por separado, usando como base los algoritmos de optimización enjambre de partículas PSO y búsqueda Cuckoo.

Los experimentos se realizaron mediante la ejecución de cinco rutas diferentes. Para cada ruta se lanzó el algoritmo cinco veces. En la **Tabla 2**, se relación las coordenadas salida y llegada de las cinco rutas de prueba.

Tabla 2: Coordenadas de las rutas de prueba

Ruta	Salida		Llegada	
	Latitud N	Longitud	Latitud N	Longitud
1	5,7899	-75,6105	4,5334	-75,0242
2	4,5677	-75,2173	5,1307	-75,6588
3	5,7075	-74,6931	4,554	-75,2863
4	4,8218	-76,2382	4,9797	-74,5482
5	5,6389	-75,2311	4,4784	-75,3001

En la **Tabla 3**, se presentan los resultados de las simulaciones para cada una de las ejecuciones de las cinco rutas.

Tabla 3: Simulación de las cinco rutas

Ruta	Enjambre de partículas		Búsqueda Cuckoo	
	Distancia total Km	Tiempo Total sg	Distancia total Km	Tiempo Total sg
1	203.28	212.88	263.12	31.67
	215.43	223.27	259.29	30.82
	203.32	212.94	243.77	28.37
	202.46	225.90	304.59	36.32
	203.19	240.88	242.78	28.64
2	80.00	80.15	94.10	10.51
	80.00	79.04	93.56	10.81
	80.00	79.38	97,36	11.12
	80.00	79.74	98.46	17.68
	80.00	79.78	95.10	10.07
3	153.09	158.03	328.76	39.50
	155.77	158.00	211.20	25.38
	160.34	175.27	219.68	26.00
	167.37	191.10	266.00	31.87
	190.50	214.70	1467.41	237.95
4	182.40	327.77	248.81	43.97
	183.74	337.57	756.44	124.24
	225.09	395.30	183.43	33.31
	197.06	352.05	861.89	141.06
	199.73	356.09	187.10	34.18
5	118.34	293.02	231.24	44.67
	112.39	219.88	126.28	27.89
	112.35	222.44	226.41	42.70
	112.83	224.83	229.69	46.59
	114.91	220.71	156.66	30.20

Las proyecciones horizontales para las rutas de prueba ejecutando el p usando la optimización PSO se muestran en la **Figura** 7 y la figura en 3d en el **anexo A**.

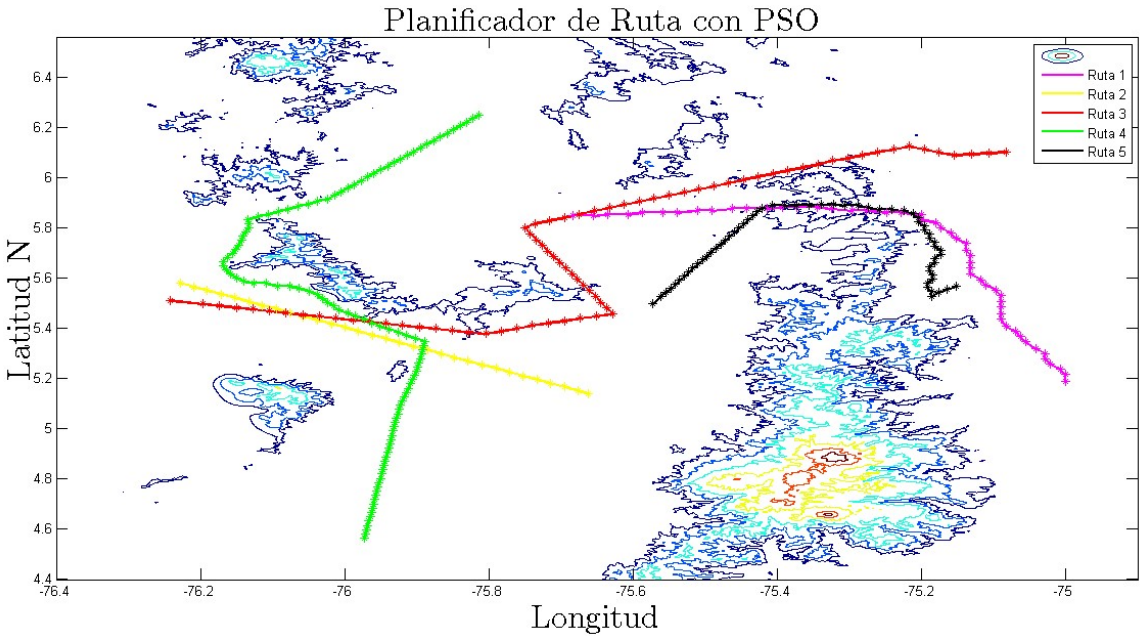

Figura 7:Método para el planeamiento de la ruta usando PSO

Las proyecciones horizontales para las rutas de prueba ejecutando el método usando la optimización CK se muestran en la **Figura 8** y la figura en 3d en el **anexo A**.

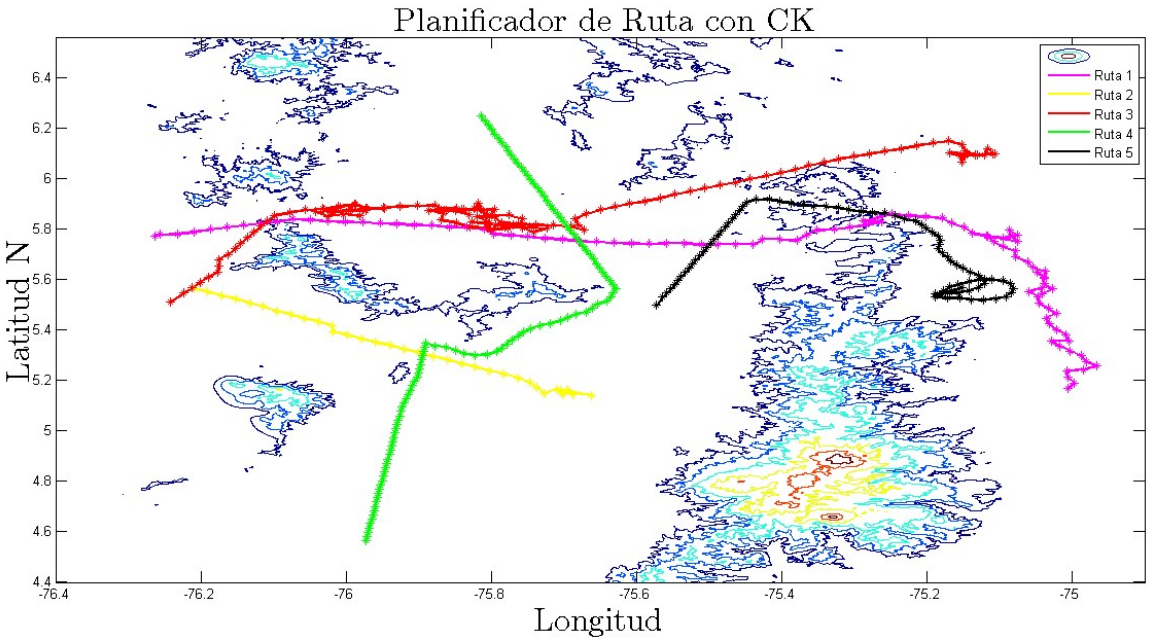

Figura 8:Método para el planeamiento de la ruta usando CK

Las **figuras 7 y 8** muestran que el método propuesto para el planeamiento de ruta encuentra todos los segmentos de ruta entre la coordenada de salida y la coordenada destino, sin importar que haya sido ejecutado con PSO o CK. El método propuesto para planeamiento de ruta usando como base en PSO mostró mayor estabilidad a la hora de encontrar la ruta. Esto es evidente en zonas que presentan múltiples accidentes geográficos como los valles profundos y montañas muy altas. La razón es la distribución de las partículas en el espacio de búsqueda, que al ser una distribución gaussiana le permite recorrer el espacio de búsqueda de forma simétrica garantizando que en la mayoría de los casos encuentre el mínimo global.

Con respecto al método para el planeamiento de ruta usando la optimización por búsqueda Cuckoo, **Figura 8**, se evidencia que encontró la ruta al igual que PSO, que son las rutas de la **Figura 7**. Al analizar de forma visual las dos imágenes con las cinco rutas, se observa que hay una mayor fluctuación en la ruta con la búsqueda Cuckoo. Esto es consistente con lo hallado en la literatura, donde se encontró que la búsqueda Cuckoo es un algoritmo que fomenta la exploración, previniendo converger de forma prematura hacia mínimos locales, pero, a su vez implicó mayores cambios de dirección y, en consecuencia mayor distancia recorrida. La razón de este comportamiento es la forma como se descubren los nuevos nidos, ya que se usa la técnica de vuelos de Levy mostrando más disposición para encontrar posibles nuevas rutas, lo que genero mayores cambios de dirección y en consecuencia mayor distancia recorrida.

En las siguientes tablas se presenta el resumen de los datos de la simulación de la ruta para el algoritmo basado en PSO y CK:

Tabla4: Media de la distancia por ruta y número de puntos

Ruta	**Planeamiento con PSO**	Número de puntos por ruta con PSO	Planeamiento con CK	Número de puntos por ruta con CK
1	**205.54 ±5.54**	76.1 ± 1.94	262.71 ± 25.10	97.6 ± 10.06
2	**80.00± 0.01**	31 ± 0	95.72 ± 2.11	35.2 ± 1.30
3	**165.41 ± 15.03**	66.4 ± 5.22	498.61 ± 543.58	236.6 ± 303.21
4	**197.61 ±17.20**	114.8 ± 7.39	447.53 ± 333.23	246.4 ± 171.05
5	**114.16± 2.56**	63 ± 8.39	194.06 ± 49.22	103.4 ± 21.38

La distancia total está compuesta por diferentes segmentos de ruta que corresponden a las coordenadas que entrega el planificador de ruta. En la **Tabla4**, se muestra la media de la distancia recorrida en las cinco rutas más o menos la desviación estándar ejecutadas cinco veces por ruta. En todos los casos, PSO recorrió una menor distancia, mostrando que a menor cambio de dirección, menor distancia recorrida. Los resultados muestran que el planeamiento de ruta usando como base PSO tiene una desviación estándar menor que CK, lo que significa que las rutas recorridas por PSO son consistentes, sin tantos cambios de dirección. La desviación estándar en la ruta cuatro muestra una diferencia de 333 km aproximadamente, esto es un indicativo que CK, en algunas de las pruebas de las cinco realizadas para la ruta cuatro, realizó demasiados cambios de dirección, lo que se evidencia en el mayor número de puntos encontrados durante el planeamiento de la ruta.

El tiempo que necesitó el planificador en cada iteración al realizar la optimización con PSO y CK por separado se muestra en la **Tabla 5**.

Tabla 5: Tiempo por iteración con PSO y CK

	Tiempo por iteración	
Ruta	PSO	**CK**
1	2,80 +- 0,33	**0,31 +- 0,032**
2	2,78 +- 0,34	**0,32 +- 0,033**
3	2,80 +- 0,32	**0,31 +- 0,034**
4	3,01 +- 0,47	**0,32 +- 0,034**
5	3,16 +- 0,45	**0,31 +- 0,034**

En la **Tabla 5**, se muestran los resultados de la media del tiempo que necesitó el planificador de ruta para encontrar el rumbo en cada uno de los segmentos de la ruta (iteración i) que componen la ruta total; es decir, desde la coordenada de salida hasta la coordenada destino. La media del tiempo en cada iteración del método usando por separado la optimización por PSO y CK.

Tabla 6: media del tiempoque necesita una iteración usando PSO y CK

Promedio del tiempo	Enjambre de partículas PSO	**Búsqueda Cuckoo CK**
Tiempo del algoritmo en segundos	2.91 ± 0.38	**0.31 ± 0.033**

En la **Tabla 6**, se observa la media del tiempo en segundos que necesitó el algoritmo de planeamiento de ruta cuando se ejecutó teniendo como base el optimizador por enjambre de partículas y por la búsqueda Cuckoo. Los resultados muestran que la búsqueda Cuckoo necesito un 10% menos en tiempo de ejecución que el algoritmo por enjambre de partículas. Eso es indicativo de que el costo computacional es mucho menor respecto al PSO. Este tiempo es la media respecto a lo que necesitó cada uno de los algoritmos para determinar un nuevo cambio en la dirección en busca de llegar a un destino predeterminado. En la **Tabla 7**, se presentan los resultados de la media del tiempo total que necesitó cada algoritmo para alcanzar el destino indicado. Es decir, la suma del tiempo de todas las iteraciones que necesitó cada algoritmo para determinar un nuevo rumbo.

Tabla 7: Resumen del tiempo del algoritmo por ruta

Tiempo del algoritmo por ruta segundos	PSO	**CK**
1	223.17± 11.53	**31.16± 3.20**
2	79.62± 0.42	**12.04± 3.17**
3	179.42± 23.57	**72.14±92.86**
4	353.76± 25.84	**75.36± 52.80**
5	236.18± 31.83	**38.41± 8.69**

Los resultados de la **Tabla 7** muestran que en cada uno de los recorridos CK necesitó un menor tiempo para determinar un nuevo rumbo, mientras que la desviación estándar del tiempo total de cada una de las rutas, muestra variaciones altas en el tiempo total por ruta. Es decir, cada vez que se ejecutó el optimizador para cada una de las rutas requirió un tiempo diferente, indicativo que en algunas ocasiones necesitó de más iteraciones para encontrar una ruta total. Nuevamente se evidencia que en la tercera ruta no hubo diferencias estadísticamente significativas, la razón es que esta ruta específica se dio

sobre un trazado en línea recta sin obstáculos, lo cual es un indicador que a medida que aumenta la dificultad, cada uno de los algoritmos toma características propias para la exploración.

4.1.1 Análisis de los resultados del método para planificación de ruta

Este trabajo propone un método para la planificación de ruta para vehículos aéreos tripulados y no tripulados que considera las propiedades básicas de vuelo propias de estos dispositivos. Las simulaciones se realizaron usando una base de datos real, la cual proporcionó las características propias del terreno como si se estuviera realizando un vuelo real. El planificador de ruta se probó usando dos algoritmos de optimización, el primero el denominado enjambre de partículas y el segundo la búsqueda Cuckoo. Los resultados mostraron el potencial del método propuesto, además de la versatilidad para ser usado con diferentes algoritmos basados en poblaciones.

Los resultados de las simulaciones mostraron que los algoritmos de optimización PSO y CK, tienen características diferentes a la hora de planear la ruta on-line. El planeamiento de ruta basado en PSO mostro más suavidad en los trayectos, evidenciado en la desviación estándar de las distancias recorridas en las pruebas. Es decir, realizó muchos menos cambios de dirección con respecto al algoritmo basado en CK. En consecuencia, PSO recorrió mucha menor distancia que CK. Otra conclusión importante es el tiempo que necesitó PSO y CK para poder entregar un nuevo rumbo. Para este caso CK fue un 89.8% más rápido que PSO. Es decir, en promedio PSO requirió 3:67 segundos para poder entregar un nuevo rumbo, mientras que CK requirió en promedio 0:37 segundos para entregar un nuevo rumbo.

Además, los algoritmos presentan características importantes en estabilidad de ruta y tiempo de ejecución. Mientras que CK presenta una mayor exploración de posibles nuevas rutas o caminos por donde seguir, PSO fue más estable y cuando localizó una ruta hizo lo posible por seguir en ella. Esto, complementado con el tiempo, nos muestra que los dos algoritmos son de buenas prestaciones en el planeamiento de ruta, lo que se

debe decidir es si requiere mayor exploración o se si se requiere una ruta más decidida con un mayor tiempo de computo.

4.2 Evaluación del rendimiento del hardware embebido ejecutando el planificador de ruta

Con el propósito de extender la funcionalidad del método presentado e integrarlo a los vehículos aéreos no tripulados esta sección determina la viabilidad de implementación del método para el planeamiento de ruta en tiempo real, analizando el desempeño al ejecutar el planificador de ruta en dispositivos con recursos limitados en memoria y capacidad computacional. Para el análisis se emplean tres plataformas de hardware aptas para ser implementadas sobre UAVs. Además, se mide el consumo de energía por cada iteración del método y se mide la energía requerida en toda la ruta.

4.2.1 Plataformas de hardware adoptadas

Para la evaluación se han empleado tres plataformas de hardware, basadas en ARM Cortex-M0+, M4(Semiconductor, 2013, 2014a) y ColdFire V1Flexis de 32-bit(Semiconductor, 2008). La familia de procesadores ARM Cortex-M, son sistemas de bajoconsumo de energía, pensados para procesos de desarrollo rápido y eficiente, además, se encuentran en múltiples plataformas, de fabricantes diferentes con la licencia ARM.

Para la codificación del método para el planeamiento de ruta se ha empleado el lenguaje de programación C, además, se realizó algunas modificaciones a los algoritmos y a la base de datos con el fin de ser implementados en dispositivos de bajo desempeño, principalmente para aquellos que están limitados en memoria de programa. A continuación, se realiza una breve descripción de los sistemas de desarrollo empleados para la evaluación:

- ColdFireFlexis-JM128-HHB

El sistema de desarrollo Flexis-JM128-HHB se basa en un microcontrolador MCF51JM128 de 32 bits. El microcontrolador MCF51JM128 tiene un núcleo ColdFire

V1Flexis 32-bit que proporciona el funcionamiento a ultra baja potencia, con un set de instrucciones reducido propietario ColdFire V1 denominado como (ISA_C). La velocidad del núcleo alcanza los 50MHz y 25 MHZ de velocidad de BUS, tiene una capacidad de 256 KB de flash y 32 KB de SRAM(Semiconductor, 2008).

- Freescale Freedom Kinetis-M KL25Z

Este sistema de desarrollo Freedom Kinetis-M KL25Z, se basa en el núcleo Cortex-M0+ de ARM, desarrollado principalmente para bajo consumo de potencia. La arquitectura Cortex-M0+ es una arquitectura tipo RISC, Von Newman, la cual cuenta con 16 registros de 32-bits, en su mayoría de propósito general, varios modos de operación: modo ejecución, modo excepción y modo depuración, manejo de un espacio de memoria de hasta 4GB, entre otras características. La frecuencia máxima soportada por la CPU es de 48MHz. En aspectos de desempeño, la arquitectura Cortex-M0+ ofrece mayor rendimiento en comparación con la arquitectura ColdFire v1 también de 32-bits (Semiconductor, 2014b).

- Freescale Freedom FRDM-K64F

Este sistema de desarrollo de Freescale FRDM-K64F, se basa en el núcleo Cortex-M4 de ARM, desarrollado principalmente para bajo consumo de potencia. El sistema incluye el MCU MK64FN1M0VLL12 (32-bits), con una frecuencia máxima soportada por la CPU de 120MHz y 60 MHz en la frecuencia del BUS. La arquitectura Cortex-M4 es una arquitectura tipo RISC que en aspectos de desempeño ofrece mayor rendimiento que ARM Cortex-M0+ ya que cuenta con arquitectura Harvard lo que le permite acceder a las instrucciones de programa y datos en el mismo ciclo de reloj(Semiconductor, 2014a, 2014b).

En la **Tabla 8**, se muestran las características relevantes de los sistemas micros procesados usados en los experimentos.

Tabla 8: Características de los sistemas micro procesados usados en los experimentos

	Sistema micro-procesado 1	**Sistema micro-procesado 2**	**Sistema micro-procesado 3**
Sistema de desarrollo	FLEXIS-JM128-HHB	Freedom Kinetis-M KL25Z	Freedom FRDM-K64F
MCU	MCF51JM128	Kinetis MKL25Z128VLK3	MK64FN1M0VLL12
CPU	ColdFire V1 Flexis 32-bit	ARM Cortex M0+ 32 bits	ARM Cortex-M4
Arquitectura	Von Newman	Von Newman	Harvard
CPU Clock	48 MHz	48 MHz	120 MHz
Bus Clock	24 MHZ	24 MHZ	60 MHz
Memoria FLASH	128 KB	128 KB	1 MB
Memoria RAM	16 KB SRAM	16 KB SRAM	256 KB SRAM

4.2.2 Diseño del experimento para evaluar los sistemas embebidos

En los experimentos se usó una base de datos comprendida entre las latitudes 4.4029 grados N a 5.8860 grados N y las longitudes -76.3969 grados a -74.9069 grados, que corresponde a una matriz de alturas de dimensiones 217*314. Para todos los experimentos se asume un horizonte de visión δ de 10 km. Los parámetros de vuelo para la simulación se muestran en la **Tabla 9**.

Tabla 9: Parámetros de simulación sobre sistema embebido

Descripción	**Parámetro**
Velocidad crucero	500km/h
Roll	45
Ratio de giro	2,33 grados/segundo
Altitud	Variable para cada experimento

Los parámetros del algoritmo PSO que es la base del método se muestran a continuación:

Tabla 10: Parámetros algoritmo de optimización

Descripción	**Parámetro PSO**
Miembros de la población	40
Horizonte de visión	10 Km
Número máximo de iteraciones del PSO	380

Los experimentos consistieron en implementar el método propuesto para el planeamiento de rutas en sistemas de desarrollo basados en procesadores ARM cortex Mx. Para esto se codificó en lenguaje C, teniendo en cuenta los requerimientos especiales propios de cada sistema de desarrollo. Con respecto al algoritmo PSO, la codificación en lenguaje C para microcontroladores requirió de diversas pruebas con el fin de verificar la generación de números aleatorios, además se realizó un escalamiento de los valores de la matriz de alturas con el fin que la longitud del dato fuera de ocho bits y así disminuir el tamaño de la base de datos.

La metodología usada se describe en las siguientes tareas:

- Compilar los códigos en todos los MCUs y verificar la correcta ejecución de los algoritmos.
- Ejecutar los algoritmos con el fin de estimar el tiempo de ejecución por cada iteración.
- Ejecutar los algoritmos con el fin de estimar el tiempo de ejecución para la ruta completa.
- Recolectar los datos obtenidos empleando una de las interfaces de entrada/salida de cada sistema de desarrollo. Los resultados muestran las coordenadas encontradas para cada segmento de ruta, tiempo por iteración y tiempo total de ruta.
- Recolectar los datos de consumo de corriente usando un multímetro de banco, referencia HM8112-3, con Resolución: 100nV, 100pA, 100μΩ, 0,01 ° C / F

La **Figura** 9, se representa de forma gráfica el algoritmo de planeamiento de rutas.

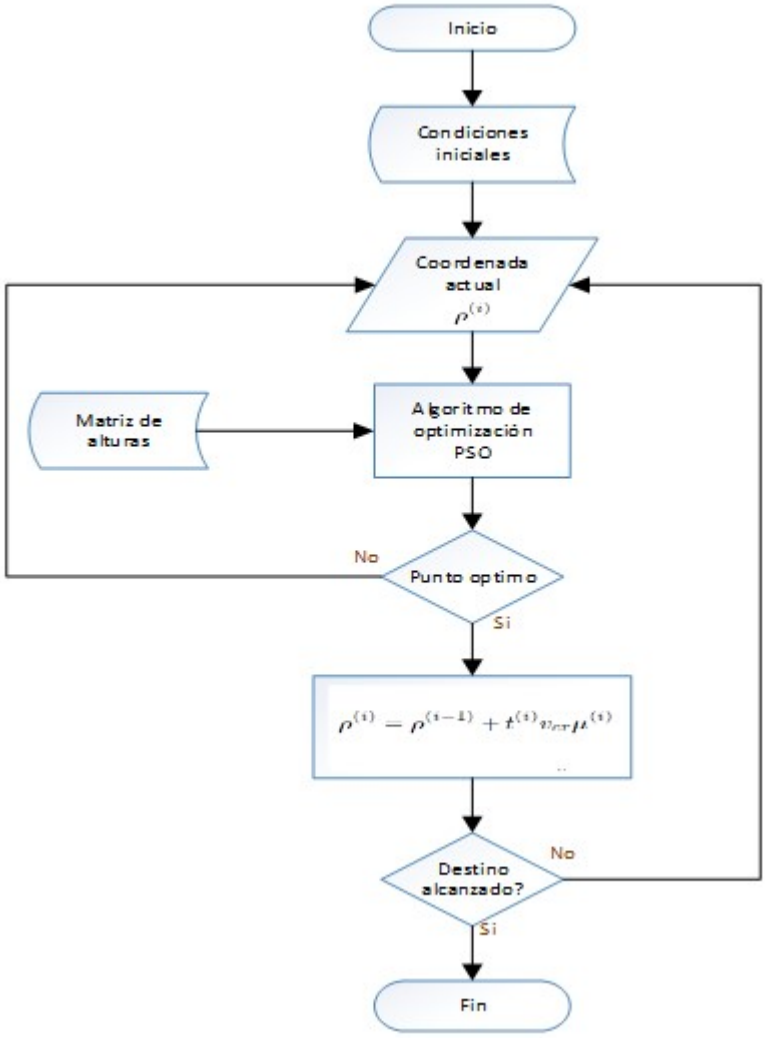

Figura 9: Codificación del algoritmo de planeamiento de ruta

En los sistemas de desarrollo Freedom Kinetis-M KL25Z, con procesador MKL25Z128 cortex M0+ y FRDM-K64F, con procesador MK64FN1M0 cortex M4, la codificación del algoritmo se realizó usando la plataforma Eclipse 10.6 IDE para microcontroladores Freescale. Para el sistema de desarrollo basado en un MCF51JM128 de 32 bits, la codificación en lenguaje C se realizó en la plataforma Code warrior 6.3. Los sistemas de desarrollo se configuraron a una velocidad de CPU de 48 MHz y velocidad de Bus de 24MHz. Todos los dispositivos con un voltaje de alimentación de 5 VDC. Además se incluye como referencia la simulación de las mismas rutas en Matlab, ejecutado sobre una CPU hp CORE i7 a 2.4GHz, con 4G en RAM (x86-64).

4.2.3 Resultado de la evaluación de los sistemas embebidos

Los experimentos consistieron en seleccionar tres coordenadas de salida y tres coordenadas de destino diferentes, lo que en consecuencia nos da tres rutas para prueba. Para cada ruta se ejecutó el algoritmo de planeamiento de ruta cinco veces.

En la **Figura** y **Figura** , se muestran las tres rutas de referencia ejecutadas en Matlab sobre el x86-64.

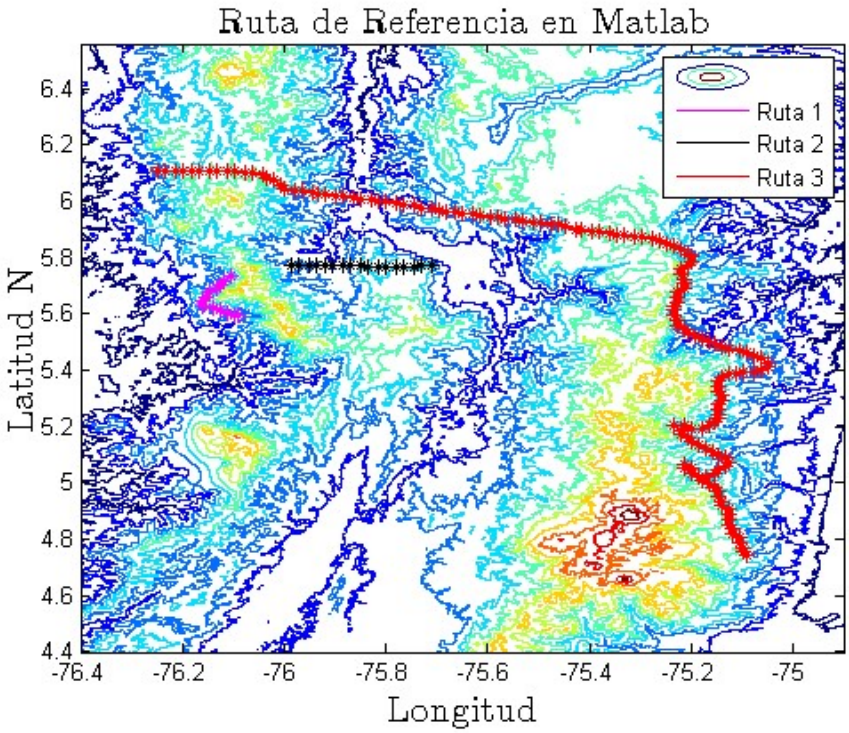

Figura 10: Contorno de la rutas de referencia

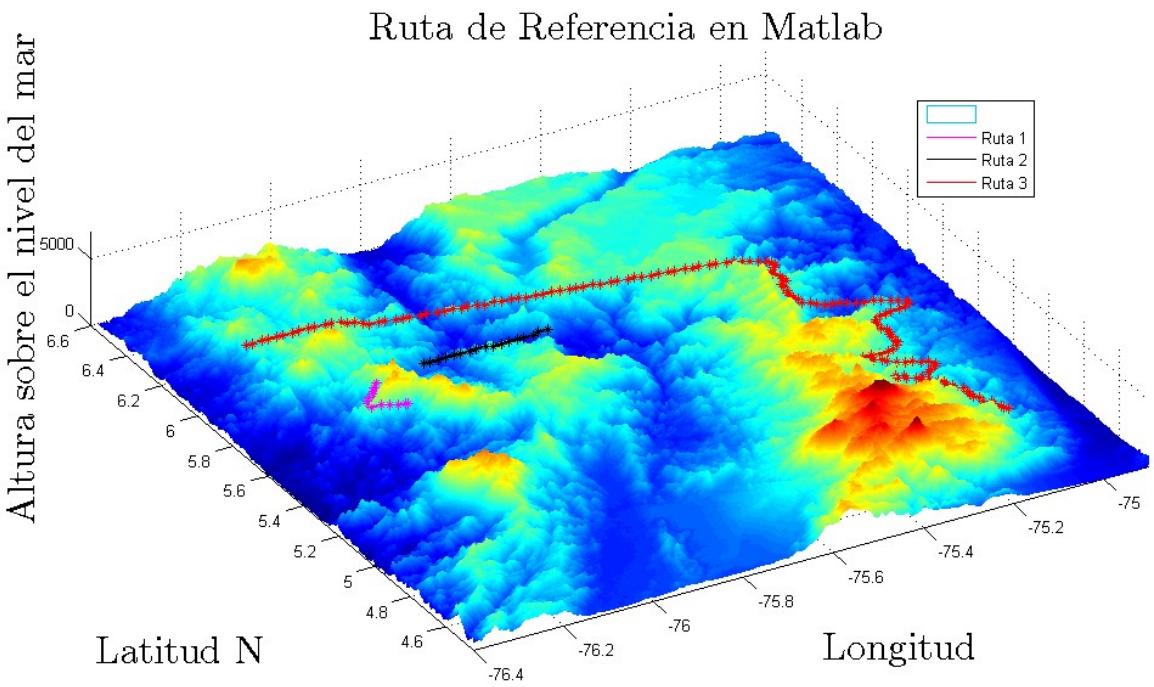

Figura 11: Topografía de las rutas de referencia

- **Tiempo por iteración del algoritmo**

La **Tabla 11**, indica los tiempos que requiere cada sistema de desarrollo para ejecutar una iteración del algoritmo de planeamiento de ruta. Es decir el tiempo que tarda cada sistema en encontrar una nueva coordenada de avance que le permita acercarse a la coordenada destino. Para lo cual, se seleccionaron siete iteraciones de los mismos segmentos de ruta para cada sistema.

Tabla 11: Tiempo por interacción del método en segundos

Iteración	MCF51JM128	MKL25Z128	MK64FN1M0	x86-64Matlab
1	54.155	26.368	**6.644**	1.85
2	53.68	26.461	**6.644**	1.86
3	54.325	26.392	**6,644**	1.84
4	54.14	26.367	**6.644**	1.85
5	54.958	25.976	**6.601**	1.88
6	53.532	26.369	**6.688**	1.88
7	54.317	26.355	**6.601**	1.91

En la **Tabla 12** se muestra el resumen de los resultados de la media del tiempo de cada una de las iteraciones.

Tabla 12: Media del tiempo por interacción del método en segundos

MCF51JM128	MKL25Z128	MK64FN1M0	x86-64Matlab
54.15 +- 0.46	26.32 +- 0.15	**6.63 +- 0.02**	1.87 +- 0.02

En la **Tabla 12** se indica el tiempo que necesitaron los sistemas microcontrolados para ejecutar cada una de las siete iteraciones del método seleccionadas. Los resultados muestran que a la misma velocidad de CPU y de BUS, el MK64FN1M0 Cortex M4 fue un 74.8% más rápido que el sistema MKL25Z128 Cortex M0+ y un 87% más rápido que el sistema MCF51JM128. La desviación estándar del tiempo para cada iteración es baja en todos los sistemas, lo que indica la estabilidad de los sistemas y el rendimiento de los mismos al ejecutar el método propuesto para el planeamiento de rutas. Además, se evidencia la capacidad del procesador MK64FN1M0 Cortex M4 para ejecutar algoritmos de optimización bio-inspirados, qué comparado con la ejecución del algoritmo sobre el x86-64, fue un 71.7% más lento, ejecutando cada iteración en $6.63s$ contra $1.85s$ de la plataforma de referencia, mostrando ser el sistema de mejor rendimiento en tiempo de ejecución.

- **Tiempo total por ruta**

En la **Tabla 13**, se muestra la media más o menos la desviación estándar, del tiempo total que necesito cada uno de los sistemas de desarrollo para ejecutar el algoritmo de planeamiento de ruta y alcanzar el destino indicado.

Tabla 13: Tiempo total por ruta en minutos para cada uno de los sistemas de desarrollo

Tiempo Total de Ruta	**MCF51JM128**	**MKL25Z128** Freedom K25z	**MK64FN1M0** Freedom K64F	**x86-64 Matlab**
1	10.94 +- 0.97	5.86 +- 1.14	**1.47 +- 0.28**	0.38 +- 0.24
2	17.51 +- 0.41	8.67 +- 0.18	**2.18 +- 0.04**	0.5 +- 0.02
3	140.52 +- 8.54	80.23 +- 3.45	**15.26 +- 1.23**	4.66 +- 0.65

El resultado de los experimentos muestra que los tres sistemas embebidos evaluados tienen la capacidad de ejecutar el algoritmo para el planeamiento de ruta y planear una ruta completa. Las imágenes de la simulación del planeamiento de la ruta con cada uno de los sistemas embebidos se pueden ver en el **anexo A.**

Además, los resultados indican que de los tres sistemas, el MCF51JM128 tiene un menor rendimiento en comparación con los otros dos procesadores ARM, requiriendo un 95.5% más de tiempo respecto a la referencia, que es el planeamiento de las mismas rutas en Matlab sobre el x86-64. Respecto al sistema MKL25Z128, se observa que en tiempo de ejecución respecto a la referencia, necesitó un 93.5% más de tiempo para realizar un planeamiento completo de ruta. De los tres procesadores evaluados, el MK64FN1M0 sobre el sistema de desarrollo Freedom K64F, mostró un tener mejor rendimiento respecto a la referencia de las rutas simuladas en Matlab. Los resultados mostraron que necesito de un 74.1% más que la simulación en Matlab, un74.9% menos que la MKL25Z128 y un 86.5% menos que el MCF51JM128.

Los resultados de las simulaciones en tiempo de ejecución permiten determinar que el sistema MK64FN1M0 Cortex M4 fue el sistema que presentó un mejor rendimiento en tiempo de ejecución del método, necesitando $6{,}63s \pm 0{,}02s$ para ejecutar una iteración del método que, en síntesis, es una nueva coordenada de avance $\boldsymbol{\rho}^{(i)}$.

- **Consumo de potencia del algoritmo de planeamiento de ruta**

El enfoque consiste en estimar por separado el consumo de energía de los procesadores Cortex M0+, M4 y del ColdFire V1 Flexis de 32-bit ejecutando el algoritmo de planeamiento de ruta. Para este fin se midió la potencia total de cada uno de los sistemas de desarrollo, luego se midió la potencia en bajo consumo de los procesadores, entregando como resultado la potencia de trabajo de cada uno de los sistemas de desarrollo y de sus procesadores de forma independiente. La razón para esto es que los sistemas embebidos tienen diferentes componentes que comparten un suministro de energía común, por lo que se hace necesario realizar esta evaluación por separado (Dhouib, Senn, Diguet, Laurent, 2009).

Los resultados del consumo de energía en un segundo se muestran en la **Tabla 14**.

Tabla 14: Energía por unidad de Tiempo Joule

Prueba	JM128	KL25z	KL64F
	Joule	***Joule***	***Joule***
1	0.026 +- 17E-05	**0.005 +- 1.67E-07**	0.025 +- 3.52E-05
2	0.026 +- 5.67E-06	**0.005 +- 2.47E-07**	0.025 +- 8.72E-07
3	0.026 +- 1.21E-06	**0.005 +- 4.44E-07**	0.025 +- 2.15E06
4	0.026 +- 1.92E-07	**0.005 +- 5.15E-07**	0.025 +- 5,57E-06
5	0.026 +- 2.37E-07	**0.005 +- 1.74E-07**	0.025 +- 6.14E-06
6	0.026 +- 1.63E-06	**0.005 +- 1.05E-07**	0.025 +- 7.64E-06
7	0,026 +- 5.24E-07	**0.005 +- 1.31E-07**	0.025 +- 5.17E-06
8	0.026 +- 4.03E-07	**0.005 +- 4,56E-07**	0.025 +- 7.02E-06
9	0.026 +- 1.2E-04	**0.005 +- 6.71E-07**	0.025 +- 8.15e-06
10	0.026 +- 2.13E-05	**0.005 +- 7.9E-08**	0.025 +- 6.64E-06

Los resultados de la **Tabla 14**, muestran que el rendimiento energético evaluado en un segundo es mejor en el MKL25Z128, con un procesador ARM cortex M0+,siendo 0.005 joule, que es un 80.7% menor que el procesador MCF51JM128 ColdFire V1 Flexis de 32-bit y un 80% menor que el MK64FN1M0, con procesador cortex M4. En la **Tabla 15**, se muestran los resultados del consumo de energía del algoritmo de planeamiento de ruta al ejecutar siete iteraciones del método.Y como se explicó anteriormente cada iteración del método arroja una nueva coordenada de avance.

Tabla 15: Energía requerida por cada iteración

iteración	JM128		KL25z		K64F	
	Energía por iteración *(joules)*	Segundos	Energía por iteración *(joules)*	Segundos	Energía por iteración *(joules)*	Segundos
1	14.20	54.15 +- 0,46	**1,33**	26.32 +- 0,15	1.67	**6.63 +- 0.02**
2	14.34	54.15 +- 0,47	**1,34**	26.32 +- 0,16	1.67	**6.63 +- 0.03**
3	14.33	54.15 +- 0,48	**1,34**	26.32 +- 0,17	1.66	**6.63 +- 0.04**
4	14.33	54.15 +- 0,49	**1,33**	26.32 +- 0,18	1.66	**6.63 +- 0.05**
5	14.33	54.15 +- 0,50	**1,33**	26.32 +- 0,19	1.66	**6.63 +- 0.06**
6	14.33	54.15 +- 0,51	**1,33**	26.32 +- 0,20	1.66	**6.63 +- 0.07**
7	14.33	54.15 +- 0,52	**1,34**	26.32 +- 0,21	1.66	**6.63 +- 0.08**

La **Tabla 15** muestra el consumo de potencia (capacidad de batería) por cada iteración del método de planeamiento de ruta. Los resultados indican que al finalizar cada una de las iteraciones del método, el rendimiento energético del procesador MK64FN1M0 sobre el sistema de desarrollo Freedom K64F fue un 20.3% mayor que el rendimiento energético del sistema MKL25Z128. Además, se observa que en cada una de las iteraciones del método el sistema basado el procesador MCF51JM128 consume un 89% más que el sistema MKL25Z128 y un 88% más que el sistema MK64FN1M0.

Los resultados permiten determinar que en energía requerida, el sistema Freedom Kinetis-MKL25Z, basado en un procesador MKL25Z128 tuvo el mejor rendimiento energético de los sistemas bajo experimentación, pero necesito un 74.8% más de tiempo para realizar la ejecución de cada una de las iteraciones.

4.2.4 Análisis de resultados de los sistemas embebidos

Con el fin de evaluar y determinar el rendimiento de sistemas de desarrollo ColdFire V1 Flexis de 32-bit y Freedom basados en arquitecturas ARM cortex M0+ y M4, ejecutando el método de planeamiento de ruta propuesto en este trabajo, se codificó el algoritmo de planeamiento de ruta en lenguaje C.

Los resultados del tiempo de ejecución para cada iteración $\boldsymbol{\rho}^{(i)}$ del método, mostraron que a la misma velocidad del reloj de la CPU, el sistema MK64FN1M0 sobre el sistema de desarrollo Freedom K64F con un procesador ARM cortex M4 mostró un mejor rendimiento al ejecutar una iteración del algoritmo propuesto en un tiempo promedio de $6.63 \pm 0.02s$, siendo un 74.7% más rápida la ejecución que el sistema MKL25Z128. En cuanto al consumo de energía, se observa que el sistema Freedom KL25z con un procesador ARM cortex M0+ tiene un menor consumo de energía de $0.005 \pm 1.67E-07$ (joules) por cada segundo que se ejecuta el método contra $0.025 \pm 3.52E-05$ (joules) del sistema MK64FN1M0.

La observación de los resultados de la ejecución de las tres rutas, muestran que el sistema MK64FN1M0 tiene el mejor rendimiento de los tres sistemas evaluados. Para la primera ruta el sistema basado el procesador MCF51JM128 necesitó10.94 ± 0.97 minutos. El sistema MKL25Z128 Cortex M0+ necesitó 5.86 ± 1.14 minutos para la

misma ruta y el sistema MK64FN1M0 necesitó1.47 ± 0.28 minutos para la misma ruta. Mientras que la referencia de la misma ruta ejecutada en Matlab sobre una x86-64 necesitó0.38 ± 0.24 minutos.

Los experimentos realizados se configuraron para una velocidad de CPU de 48 MHz y velocidad de Bus de 24MHz. El sistema MCF51JM128 ColdFire V1 puede subir a una velocidad máxima de 50MHz de la CPU y 25MHz en velocidad de BUS, por lo que los cambios en el tiempo de ejecución no son significativos. En el sistema MKL25Z128 Cortex M0+ puede subir a una velocidad máxima de 50MHz de la CPU y 25MHz en velocidad de BUS, por lo que los cambios en el tiempo de ejecución respecto a los obtenidos a 48MHz no son significativos.

A diferencia de los sistemas anteriores, el sistema MK64FN1M0 puede trabajar a una velocidad máxima de CPU de 60MHz, con lo que el tiempo de ejecución por iteración se redujo un 18% llegando a $5.24 \pm 0.01\, s$ mostrando que el sistema Freedom K64F, con un procesador ARM cortex M4, es el que presenta mejores prestaciones para ejecutar el algoritmo para el planeamiento de rutas, conservando un tiempo de ejecución reducido frente a un sistema de cómputo de muchas más prestaciones como lo es una x86-64 Matlab, pero conservando un consumo de energía menor. Además, las dimensiones físicas y de peso son reducidas, por lo que puede ser implementado fácilmente en plataformas de vuelo que requieran de sistemas embebidos de dimensiones limitadas.

En la **Tabla 16**, se presenta la relación de distancia recorrida por un vehículo aéreo en una iteración del algoritmo de planeamiento de ruta según la velocidad de desplazamiento ejecuta sobre el sistema MK64FN1M0 a una velocidad de CPU de 60MHz, que es la velocidad del sistema micro procesado usando un reloj externo.

Tabla 16: Distancia recorrida por una iteración

Velocidad Km/h	Tiempo iteración	Distancia Recorrida mt/it
800	5.24	1164
700	5.24	1018
600	5.24	873
500	5.24	727
400	5,24	582
300	5.24	436
200	5.24	291
100	5.24	145
50	5.24	72
20	5.24	29
10	5.24	14
5	5.24	8

Cabe anotar que si la aeronave recorriera una distancia en proporción al tiempo de ejecución del algoritmoen la realidad, cuando llegue al límite de la distancia podría encontrarse con un objeto que está por fuera del rango de visión y, en consecuencia, no tener el tiempo de realizar la evasión del obstáculo. Para encontrar la velocidad máxima en función del ángulo de alabeose da en (Civil, 2009) **ecuación(14),** donde se puede obtener la tasa de giro por segundo en función del máximo ángulo de alabeo soportado por la aeronave. A partir de este concepto se obtiene el radio de viraje R_{vr}para una tasa de giro determinada, ecuación **(19),** (Internacional, 2009).

$$R_{vr} = \frac{v_{cr}}{(20 * \pi * \theta_g)} \qquad \textbf{(19)}$$

El radio de viraje permite encontrar la velocidad máxima a la que se debe desplazar la aeronave en función del espacio de seguridad que necesita la aeronave para girar, respetando el tiempo que necesita para realizar el planeamiento de la ruta en función de los obstáculos que se presentan en su entorno.

4.3 Aporte del método propuesto

Por medio de los resultados obtenidos se comprueba que el algoritmo para el planeamiento de rutas en tiempo real basado en técnicas de optimización bio-inspirados se puede implementar sobre hardware embebido basado en sistemas micros procesados. Además, el método propuesto muestra ser versátil y de fácil implementación, lo que permite que trabajos futuros puedan usar este método como referente para evaluar nuevas técnicas de optimización usando bases de datos reales.

Este trabajo es un aporte en la implementación de técnicas de inteligencia artificial con el fin de dotar a vehículos aéreos no tripulados con la capacidad de navegación y planeamiento de rutas en línea, en miras de mejorar la seguridad aérea, en especial con la proliferación de drones, no solo militares sino de uso civil.

5. Conclusiones

Este trabajo propone un método para la planificación de ruta para vehículos aéreos tripulados y no tripulados que considera las propiedades básicas de vuelo propias de estos dispositivos. Las simulaciones se realizaron usando una base de datos real, la cual proporcionó las características propias del terreno como si se estuviera realizando un vuelo real. El planificador de ruta se probó usando dos algoritmos de optimización, el primero el denominado enjambre de partículas y el segundo la búsqueda Cuckoo.

Los resultados mostraron el potencial del método propuesto, además de la versatilidad para ser usado con diferentes algoritmos basados en poblaciones.Los resultados de las simulaciones mostraron que los algoritmos de optimizaciónPSO y CK, tienen características diferentes a la hora de planear la ruta en línea.

El planeamiento de ruta basado en PSO mostró más suavidad en los trayectos, evidenciado en la desviación estándar de las distancias recorridas en las pruebas. Es decir, realizó muchos menos cambios de dirección con respecto al algoritmo basado en CK. En consecuencia, PSO recorrió mucha menor distancia que CK. Otra conclusión importante es el tiempo que necesitaron PSO y CK para poder entregar un nuevo rumbo. Para este caso CK fue un 89.8% más rápido que PSO, es decir, en promedio PSO requirió 2.91 segundos para poder entregar un nuevo rumbo, mientras que CK requirió en promedio 0.31 segundos para entregar un nuevo rumbo.

Como conclusión final se puede decir que los dos algoritmos presentan características importantes en estabilidad de ruta y tiempo de ejecución. Mientras que CK presenta una mayor exploración de posibles nuevas rutas o caminos por donde seguir, PSO fue más estable y cuando localizó una ruta hizo lo posible por seguir en ella. Esto, complementado con el tiempo, nos muestra que los dos algoritmos son de buenas prestaciones en el planeamiento de ruta, lo que se debe decidir es si se requiere mayor exploración o se si se requiere una ruta más decidida con un mayor tiempo de computo.

Una implementación inicial muestra que es posible incorporar el algoritmo para el planeamiento de rutas basados en técnicas de optimización bio-inspirados en

dispositivos con baja capacidad computacional sin realizar modificaciones considerables. El rendimiento obtenido varía entre los 5 y los 54 segundos, dependiendo de la frecuencia de reloj y de la arquitectura seleccionada.

En cuanto al consumo de potencia, se muestra que las arquitecturas ARM Cortex M4 tienen un consumo similar a la arquitectura ColdFire V1 de 32 bits y en el caso del MKL25Z128 Cortex M0+ tiene un consumo de potencia mucho menor. Esto permite emplear dispositivos con capacidades computacionales mayores y dimensiones reducidas sin incurrir en un elevado consumo de potencia. Los resultados de las simulaciones mostraron que, de los tres sistemas evaluados con arquitecturas diferentes, el sistema Freedom K64F, con un procesador MK64FN1M0, presentó las mejores prestaciones respecto a los otros sistemas de desarrollo, siendo una buena opción para dotar a sistemas aéreos autónomos con capacidad de navegación autónoma sin depender de un controlador remoto.

Debido a los buenos resultados obtenidos con el algoritmo para el planeamiento de ruta propuesto en este trabajo, se considera que, a futuro, se puede aplicar esta misma metodología al planeamiento de rutas en la aviación convencional con el fin de ser unsoporte al controlador en tierra y a los tripulantes de aeronaves comerciales en la planificación de la ruta en línea, considerando la ventana de vuelo que tiene la ruta virtual asignada en el plan de vuelo. Todo esto se realizaría en función de optimizar los costos de operación de las aeronaves, así como disminuir la contaminación auditiva y los gases con efecto invernadero generados durante el vuelo (Berton & Guynn, 2012; Carlos, Quintero, & Carlo, 2009; Colmenares Quintero et al., 2010).

Bibliografía

Abdullah-ai-nahid, S., Alam, M. M., & Plabon, B. A. (2012). Algorithm for Maximum Solar Power Tracking, 0–5.

Aeronáutica Civil Colombiana. (2015). Autoridad Aeronáutica. Retrieved from http://www.aerocivil.gov.co/AAeronautica/Paginas/Inicio.aspx

AIP COLOMBIA ENR. (2014). Reglas de vuelo visual, 4–5.

Arboleda, D. M. M., Llanos, C. H., Coelho, L. dos_S., & Ayala-Rincón, M. (2009). Hardware Architecture for Particle Swarm Optimization Using Floating-Point Arithmetic. *2009 Ninth International Conference on Intelligent Systems Design and Applications*, 243–248. doi:10.1109/ISDA.2009.107

Baynes, K., Collins, C., Fiterman, E., Ganesh, B., Kohout, P., Smit, C., & Member, S. (2005). The Performance and Energy Consumption of Embedded Real-Time Operating Systems, *52*(11), 1454–1469.

Berton, J. J., & Guynn, M. D. (2012). Multi-Objective Optimization of a Turbofan for an Advanced , Single-Aisle Transport, (April).

Carlos, J., Quintero, C., & Carlo, M. (2009). GT2009-59096, *C*, 1–15.

Chen, H., Chang, K., & Agate, C. (2013). UAV path planning with tangent-plus-Lyapunov vector field guidance and obstacle avoidance. *Aerospace and Electronic …*, *49*(2). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6494384

Cheng, Z., Wang, E., Tang, Y., & Wang, Y. (2014). Real-time Path Planning Strategy for UAV Based on Improved Particle Swarm Optimization. *Journal of Computers*, *9*(1), 209–214. doi:10.4304/jcp.9.1.209-214

Civicioglu, P., & Besdok, E. (2013). *A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms*. *Artificial Intelligence Review* (Vol. 39, pp. 315–346). doi:10.1007/s10462-011-9276-0

Civil, O. de A. (2009). *Manual de diseño de procedimientos de performance de navegación requerida con autorización obligatoria (RNP AR)* (2009th ed.). 2010.

Clerc, M. (2005). *Particle Swarm Optimization*.

Clerc, M. (2012). Standard Particle Swarm Optimisation, 1–15.

Colmenares Quintero, R. F., Brink, R., Ogaji, S., Pilidis, P., Colmenares Quintero, J. C., & Quintero, A. G. (2010). Application of the Geared Turbofan With Constant Volume Combustor on Short-Range Aircraft: A Feasibility Study. *Journal of Engineering for Gas Turbines and Power*, *132*(6), 061702. doi:10.1115/1.4000135

Darío, R., Tarazona, F., & Lopera, F. R. (2014). Anti-collision System for Navigation Inside an UAV Using Fuzzy Controllers and Range Sensors. *978-1-4799-7666-9/14/$31.00 ©2014 IEEE 2014 XIX Symposium on Image, Signal Processing and Artificial Vision (STSIVA*.

Diario Crítico Colombia. (2012). No Titleaviones-no-tripulados. *http://colombia.diariocritico.com/noticias/aviones-no-tripulados/293335*.

Digalakis, J. G., & Margaritis, K. G. (2000). ON BENCHMARKING FUNCTIONS FOR GENETIC ALGORITHMS, *00*.

Duan, H., Yu, Y., Zou, J., & Feng, X. (2010). Ant colony optimization-based bio-inspired hardware: survey and prospect. *Transactions of the Institute of Measurement and Control*, *34*(2-3), 318–333. doi:10.1177/0142331210366689

Fu, Y., & Member, S. (2012). Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Applied to Three-Dimensional Route Planning for UAV, *42*(2), 511–526.

Ginan, Z. S. I. (2014). Karınca Kolonisi Optimizasyonu ile ˙ IHA Rota Planlamasının GPU Üzerinde CUDA Yardımı ile Paralel Çözümü Parallel Solution for UAV Route Planning Problem using Ant Colony Optimisation on GPU with CUDA, (Siu).

Idoumghar, L., Melkemi, M., Schott, R., & Aouad, M. I. (2011). Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems. *Applied Computational Intelligence and Soft Computing*, *2011*, 1–12. doi:10.1155/2011/138078

International Civil Aviation Organization. (2006). Convention on International Civil Aviation. *DOC Series*. Retrieved from http://www.icao.int/publications/documents/7300_9ed.pdf

Jardin, M. (2003). Real-Time Conflict-Free Trajectory Optimization, (June).

john tisdale, zuwhan kim, and j. K. hedrick. (2009). Autonomous UAV Path Planning and Estimation, (June), 35–42.

Jones, D. H., Powell, A., Bouganis, C., & Cheung, P. Y. K. (2010). GPU versus FPGA for high productivity computing. doi:10.1109/FPL.2010.32

Kalarot, R., & Morris, J. (2010). Comparison of FPGA and GPU implementations of Real-time Stereo Vision.

Kestur, S., Davis, J. D., & Williams, O. (2010). BLAS Comparison on FPGA , CPU and GPU.

Lamont, G. B., Slear, J. N., & Melendez, K. (2007). UAV Swarm Mission Planning and Routing using Multi-Objective Evolutionary Algorithms. *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, (Mcdm), 10–20. doi:10.1109/MCDM.2007.369410

Lee, K.-B., & Kim, J.-H. (2013). Multiobjective Particle Swarm Optimization with Preference-based Sort and its Application to Path Following Footstep Optimization for Humanoid Robots. *IEEE Transactions on Evolutionary Computation*, (c), 1–1. doi:10.1109/TEVC.2013.2240688

Leonard, B. J., & Engelbrecht, A. P. (2013). On the Optimality of Particle Swarm Parameters in Dynamic Environments, 1564–1569.

Li, X., & Yin, M. (2015). Modified cuckoo search algorithm with self adaptive parameter method. *Information Sciences*, *298*, 80–97. doi:10.1016/j.ins.2014.11.042

Matsuo, K., Hamada, T., Miyoshi, M., Shibata, Y., & Oguri, K. (2009). Accelerating Phase Correlation functions using GPU and FPGA : A comparison study. doi:10.1109/AHS.2009.53

Meng, H., & Xin, G. (2010). UAV route planning based on the genetic simulated annealing algorithm. *2010 IEEE International Conference on Mechatronics and Automation*, 788–793. doi:10.1109/ICMA.2010.5589035

Munoz, D. M., Llanos, C. H., Coelho, L. D. S., & Ayala-Rincon, M. (2010). Hardware Particle Swarm Optimization Based on the Attractive-Repulsive Scheme for Embedded Applications. *2010 International Conference on Reconfigurable Computing and FPGAs*, 55–60. doi:10.1109/ReConFig.2010.73

Nguyen, T. T., & Vo, D. N. (2015). Modified cuckoo search algorithm for short-term hydrothermal scheduling. *International Journal of Electrical Power & Energy Systems*, *65*, 271–281. doi:10.1016/j.ijepes.2014.10.004

Papadonikolakis, M., Bouganis, C., & Constantinides, G. (2009). Performance Comparison of GPU and FPGA architectures for the SVM Training Problem.

Pei, Y., Wang, W., & Zhang, S. (2012). Basic Ant Colony Optimization. *2012 International Conference on Computer Science and Electronics Engineering*, 665–667. doi:10.1109/ICCSEE.2012.178

Peng, Z., & Li, B. (2012). Online Route Planning for UAV Based on Model Predictive Control and Particle Swarm Optimization Algorithm, (60925011), 397–401.

Roberge, V., Tarbouchi, M., & Labonté, G. (2013). Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning, *9*(1), 132–141.

Rodríguez-piñero, A. P. T. (2003). Introducción a los algoritmos genéticos y sus aplicaciones.

Rout, P.K., Acharya, D. P., & Panda, G. (2010). Digital circuit placement in FPGA based on efficient particle swarm optimization techniques. *2010 5th International Conference on Industrial and Information Systems*, 224–227. doi:10.1109/ICIINFS.2010.5578703

Rout, Prakash Kumar, Acharya, D. P., & Panda, G. (2010). Novel PSO based FPGA placement techniques. *2010 International Conference on Computer and Communication Technology (ICCCT)*, 630–634. doi:10.1109/ICCCT.2010.5640451

Semiconductor, F. (2008). MCF51JM128 ColdFire ® Integrated Microcontroller Reference Manual Devices Supported : Home Page :

Semiconductor, F. (2013). KINETIS L COURSE IDEAS :

Semiconductor, F. (2014a). FRDM-K64F Freedom Module User ' s Guide, 1–21.

Semiconductor, F. (2014b). Freescale Sensor Fusion Library for Kinetis.

Shudao, Z., Jun, W., & Yongqi, J. (2012). Route Planning for Unmanned Aircraft Based on Ant Colony Optimization and Voronoi Diagram, 732–735. doi:10.1109/ISdea.2011.191

Spurzem, R., Berczik, P., Marcus, G., Kugel, a., Lienhart, G., Berentzen, I., … Banerjee, R. (2009). Accelerating astrophysical particle simulations with programmable hardware (FPGA and GPU). *Computer Science - Research and Development*, *23*(3-4), 231–239. doi:10.1007/s00450-009-0081-9

Tewolde, G. S., Hanna, D. M., & Haskell, R. E. (2009). Accelerating the performance of particle swarm optimization for embedded applications. *2009 IEEE Congress on Evolutionary Computation*, 2294–2300. doi:10.1109/CEC.2009.4983226

Valavanis, K., Oh, P., & Piegl, L. (2009). *Unmanned Aircraft Systems: International Symposium on Unmanned Aerial Vehicles, UAV'08*. Retrieved from http://books.google.com/books?hl=en&lr=&id=Xqzv_wXqC5EC&oi=fnd&pg=PA1&dq=Unmanned+Aircraft+Systems+International+Symposium+on+Unmanned+Aerial+Vehicles,+UAV+%E2%80%9808&ots=PykUs9xc-U&sig=yacR1eoadCmkbKPc456eGkP9VKQ

Wei, M. (2014). Research On The Optimal Route Choice Based On Improved Dijkstra, 303–306.

Wu, J., Zhang, D., & Pei, D. (2014). Autonomous Route Planning for UAV When Threats Are Uncertain, 19–22.

Xue, Q., Cheng, P., & Cheng, N. (2014). Offline Path Planning and Online Replanning of UAVs in Complex Terrain *, 2287–2292.

Yang, X., & Deb, S. (2014). Cuckoo Search : Recent Advances and Applications The Essence of an Optimization Algorithm, 1–9.

Yang, X., & Press, L. (2010). *Nature-Inspired Metaheuristic Algorithms Second Edition*.

Yangguang Fu, Mingyue Ding, Chengping Zhou, and H. H. (2013). Route Planning for Unmanned Aerial Vehicle (UAV) on the Sea Using Hybrid Differential Evolution and, *43*(6), 1451–1465.

Zaza, T., & Richards, A. (2014). Ant Colony Optimization for Routing and Tasking Problems for Teams of UAVs, (July), 652–655.

Zhou, Y., & Zheng, H. (2013). A Novel Complex Valued Cuckoo Search Algorithm, *2013*(1).

A. Rutas de prueba

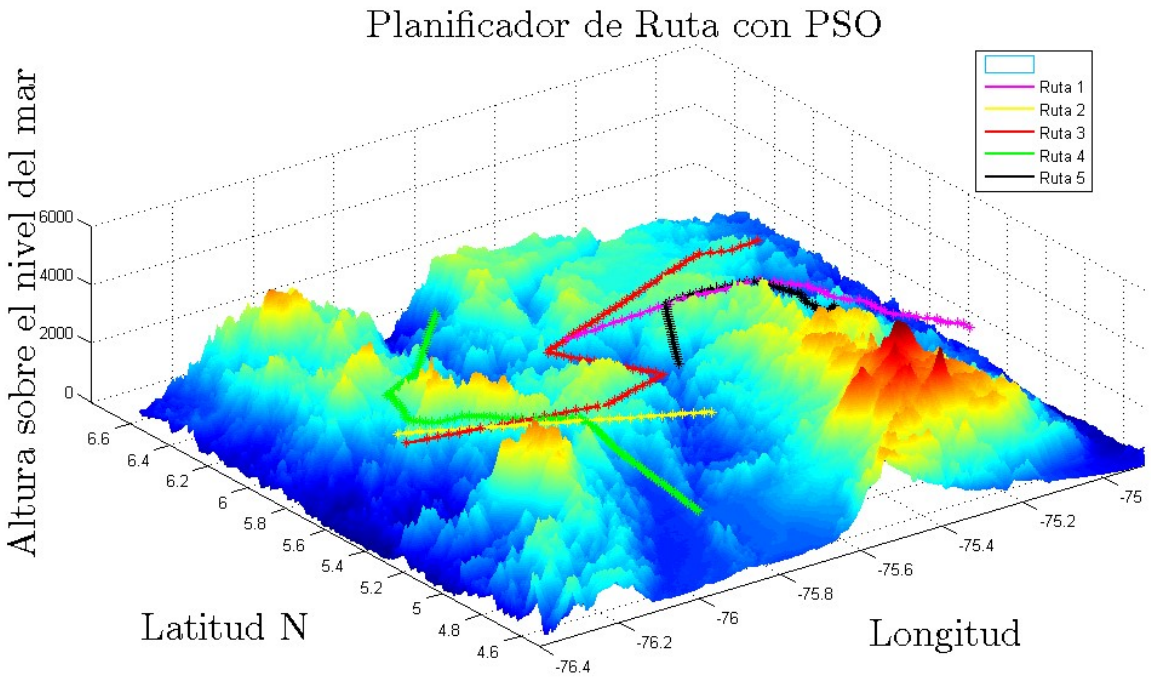

Figura 10: Método para el planeamiento de ruta usando PSO

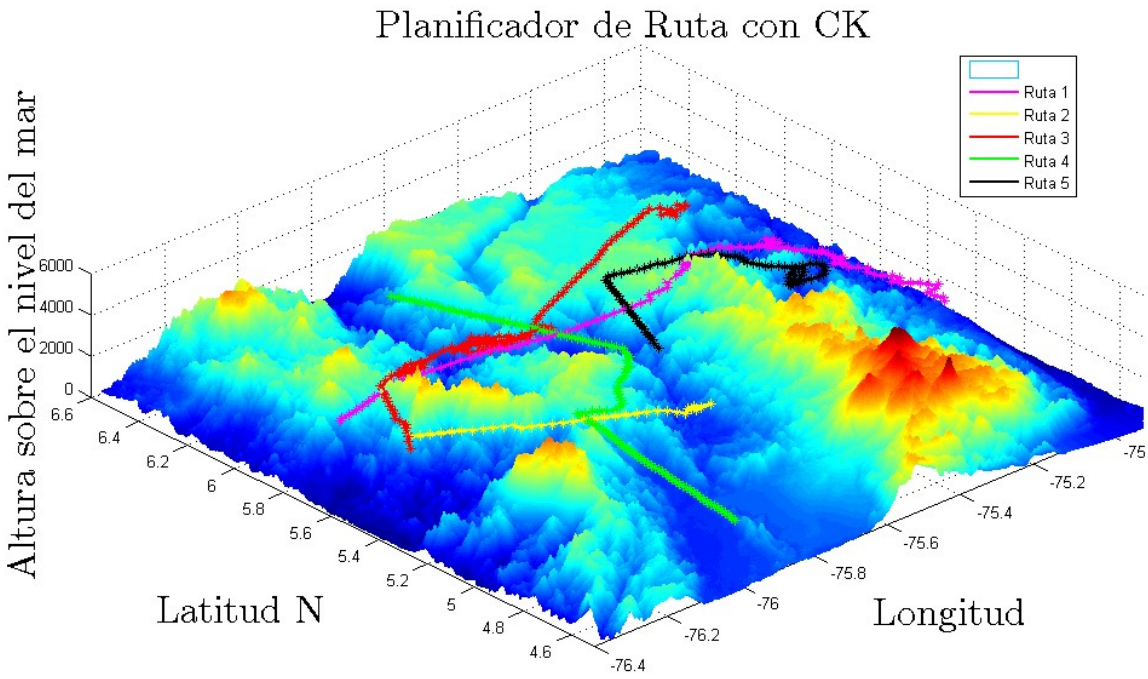

Figura 11: Método para el planeamiento de rutas usando CK

B. Simulación de las rutas sobre los sistemas microcontrolados

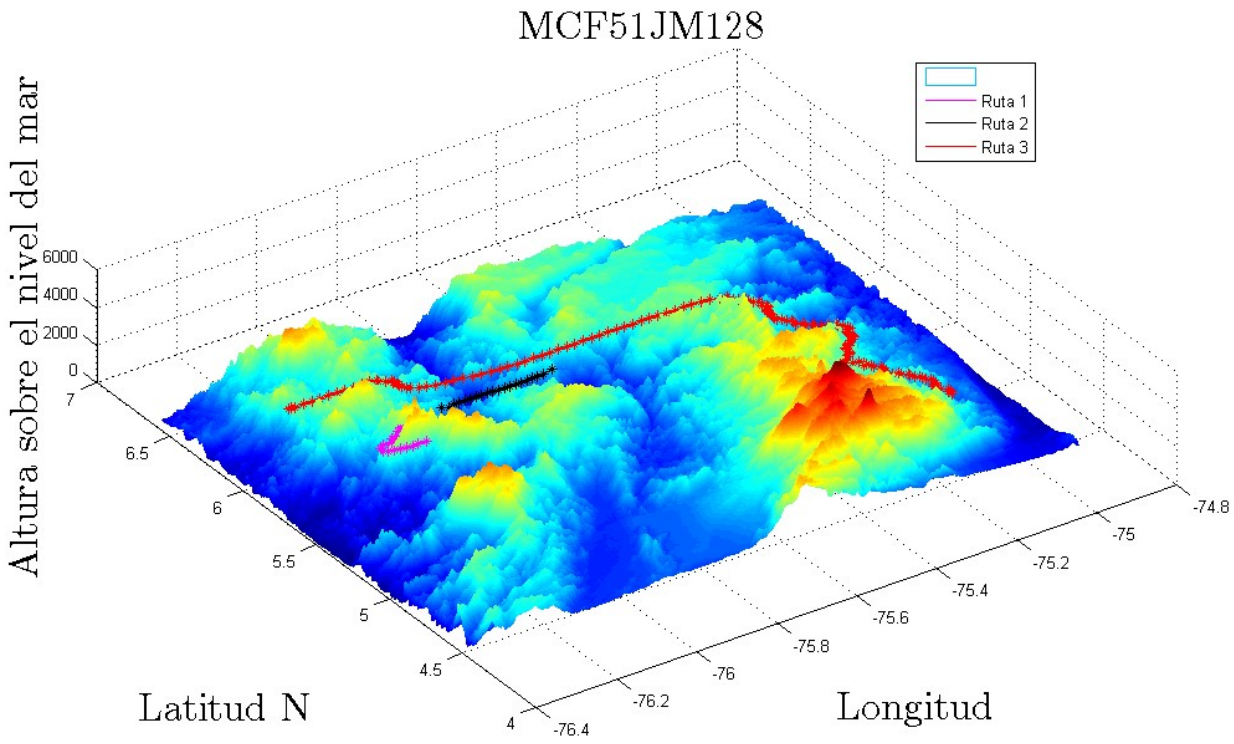

Figura 12: Método para el planeamiento de ruta sobre el MCF51JM128

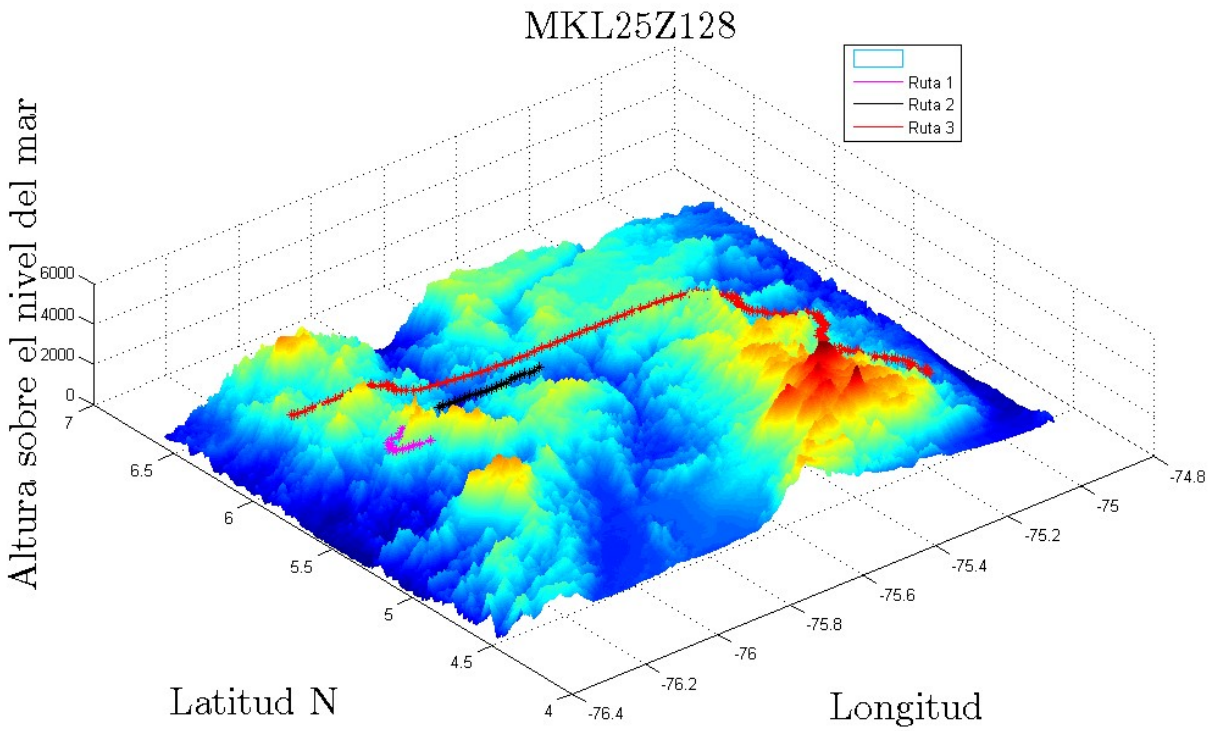

Figura 13: Método para el planeamiento de ruta sobre el MKL25z128

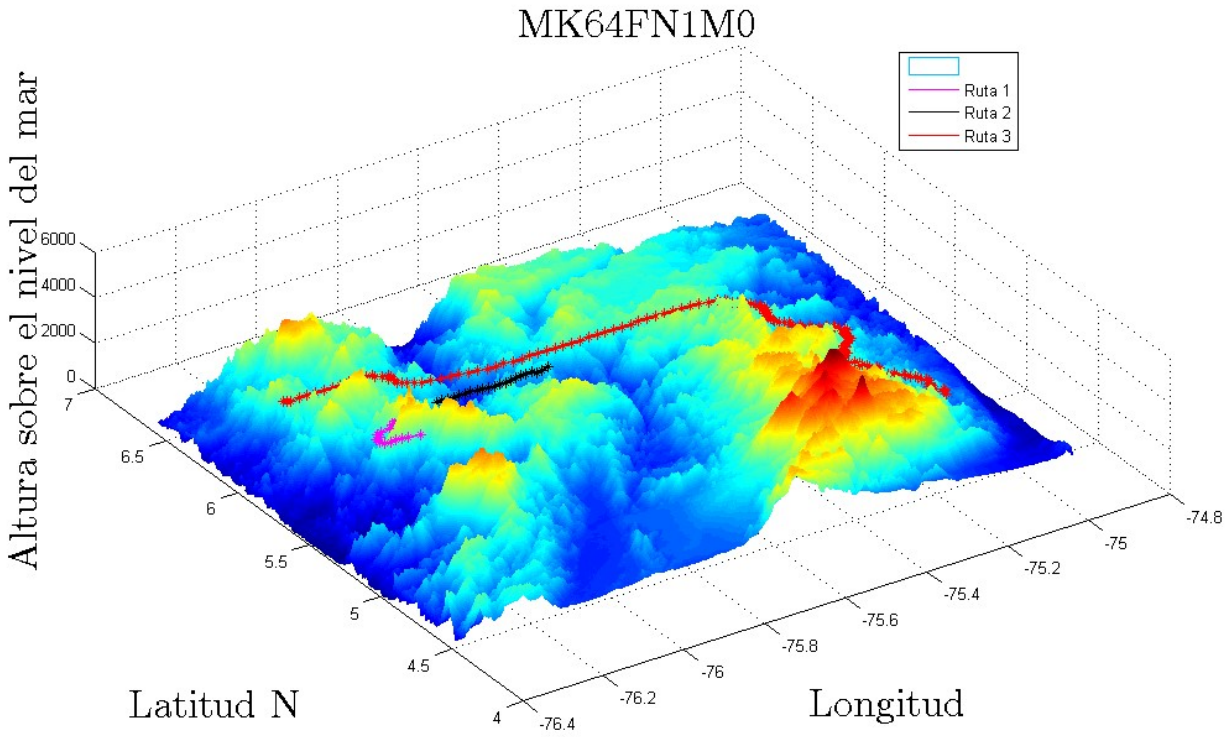

Figura 14: Método para el planeamiento de ruta sobre el MK64FNM0